Reliability-Aware Mapping Optimization of Multi-Core Systems with Mixed-Criticality

Shin-Haeng Kang*, Hoeseok Yang†, Sungchan Kim‡, Iuliana Bacivarov†, Soonhoi Ha*, and Lothar Thiele†
*School of EECS, Seoul National University, Seoul, Korea, {shkang,sha}@iris.snu.ac.kr
†TIK Laboratory, ETH Zurich, Zurich, Switzerland, {firstname.lastname}@tik.ee.ethz.ch
‡ Division of CSE, Chonbuk National University, Jeonju, Korea, sungchan.kim@chonbuk.ac.kr

***Abstract*—This paper presents a novel mapping optimization technique for mixed critical multi-core systems with different reliability requirements. For this scope, we derived a quantitative reliability metric and presented a scheduling analysis that certifies given mixed-criticality constraints. Our framework is capable of investigating re-execution, passive replication, and modular redundancy with optimized voter placement, while typical hardening approaches consider only one or two of these techniques. The proposed technique complies with existing safety standards and is power-efficient, as demonstrated by our experiments.**

I. Introduction

Today's multi-core systems with *mixed-criticality* come with a great performance potential, but also with strong requirements in terms of reliability [1]. An automotive or avionic system is a typical example, where control, signal processing, and multimedia applications co-exist with different safety and reliability requirement levels. Functional safety levels are defined by the International Electro-technical Commission in IEC 61508 [2], and specialized for the automotive domain in ISO 26262 [3]. This paper addresses the case of transient faults in mixed-criticality systems. *Re-execution* and *replication* are conventional hardening techniques [4]–[6], that come with respective overheads in terms of time and resources. Selecting one of these techniques is an optimization trade-off. When related to typical mapping/scheduling optimization trade-offs, the number of options in the design space is dramatically increasing. Several research efforts have been focusing on the fault-tolerant mapping optimization problem [4], [5], [7].

In terms of reliability model, Bolchini et al. [5] considered *fault-tolerance*, *fault-detection*, and *fault-ignorance* as three levels of mixed-criticality, but without any quantitative modeling. A probabilistic reliability model for a two-level mixed-criticality system was proposed by Axer et al. [6], where detection is achieved by a special case of replication, *dual modular redundancy* (DMR), and faults are tolerated by re-executions. The model derives the time to the first failure, but it is not clear how it can be applied for a certain failure rate per hour as specified in safety standards. Our technique has the advantage of considering a continuous range of mixed-criticality levels, enabled by a probabilistic reliability model.

In terms of system hardening, the proposed approach is generic, being able to incorporate any technique. Pop et al. [4], for instance, can only consider re-execution and replication. Passive replication, where replicated tasks are conditionally instantiated on request, is not possible there because the static scheduling cannot consider such conditional executions. In [7], passive replication can be taken into account as a fault-tolerance *pattern*, but without any worst-case performance guarantee. Grouped voter placement is proposed in [5] to reduce the overhead of voting procedures. However, they do not consider the side-effect of grouping, i.e., grouping may result in violation of reliability constraints if too many tasks are grouped without intermediate *checkings*. In the proposed approach, besides all the hardening techniques mentioned above, passive replication is also included and the voters are judiciously assigned in a grouped manner to satisfy reliability constraints.

The contribution of this paper is twofold: First, we propose a reliability-aware mapping optimization technique with general mixed-criticality levels enabled by a probabilistic reliability model, which is compatible with the existing standards. Second, the proposed technique is a generic framework that considers re-execution, active/passive replication, and the combination thereof, as well as judicious voter placement.

II. System Model

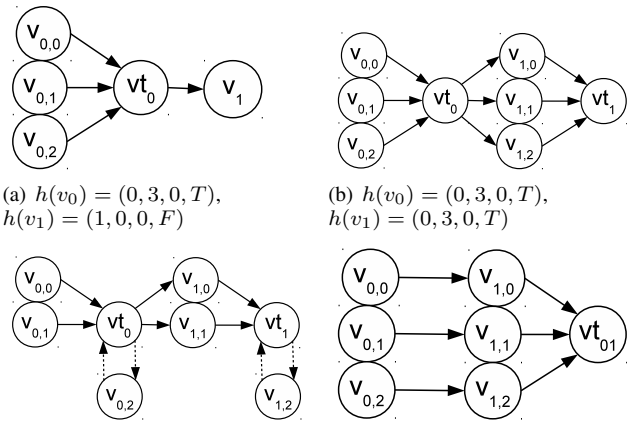
A. Target Architecture and Application

We assume a network-on-chip (NoC) based multi-core system $\mathcal{A} := (CL, nw)$, characterized by a set of clusters CL and an on-chip network nw. The on-chip network nw is characterized by its bandwidth limit bw_{nw}. A cluster $cl \in CL$ consists of a set of processors P_{cl} connected through an intra-cluster communication medium, such as a shared bus or a crossbar, characterized similarly to the on-chip network by an upper-bound on bandwidth bw_{cl}.[1] Each processor $p \in P_{cl}$ is characterized by its $type_p$, leakage power $stat_p$, dynamic power dyn_p, and a constant fault rate per time unit λ_p. Constant fault rate for a processor is popularly assumed as shown in [9], [10].

Multiple applications with different levels of criticality are sharing the system, each of them being described as a task graph such as a Kahn process network (KPN). An application set is defined as $\mathcal{T}$, whose elements are the task graphs $t := (V_t, E_t, pr_t, f_t) \in \mathcal{T}$. Each task graph t consists of a set of

[1] We do not consider faults in the communication links, since they are usually protected by low-level error-resilient techniques [8].

978-3-9815370-2-4/DATE14/©2014 EDAA



(a) $h(v_0) = (0, 3, 0, T)$, $h(v_1) = (1, 0, 0, F)$

(b) $h(v_0) = (0, 3, 0, T)$, $h(v_1) = (0, 3, 0, T)$

(c) $h(v_0) = (0, 2, 1, T)$, $h(v_1) = (0, 2, 1, T)$

(d) $h(v_0) = (0, 3, 0, F)$, $h(v_1) = (0, 3, 0, T)$

Fig. 1: Hardening examples: (a) triplication of v_0 and re-execution of v_1, (b) triplication for both, (c) passive replication, and (d) selective voter placement with grouping.

tasks V_t, a set of channels E_t, an invocation period $pr_t \in \mathbb{Z}$, and a reliability constraint $f_t \in (0, 1]$. The reliability constraint f_t denotes the number of maximum allowable failures per unit time. The lower the reliability constraint f_t is, the higher the criticality level is. An instance of the task graph is released every pr_t time units, and the probability of *unsafe execution* should be smaller than f_t.

Each task $v \in V_t$ of the task graph t is characterized by $(bcet_v, acet_v, wcet_v, ve_v, dt_v)$, i.e., best/average/worst-case execution time (BCET/ACET/WCET), voting overhead, and detection overhead. The voting overhead ve_v is related to replication, while detection dt_v overhead includes fault detection, storing/restoring the context, and rolling-back for re-execution. These will be explained in more detail in the following subsection. A channel $e := (src_e, dst_e) \in E_t$ represents a data dependency from task src_e to dst_e and each transmission causes a data transfer of size s_e.

B. Fault Management

Choosing an appropriate hardening technique for a task $v \in V$ can be defined as a function $h : V \to \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \{T, F\}$, where V is a set of all tasks in the system ($\bigcup_{t \in \mathcal{T}} V_t$), which can have different degrees of re-execution, active replication, passive replication, and voter existence, respectively (i.e., the three natural numbers and one boolean value).

Re-execution: In re-execution scheme, it is assumed that a fault is locally detected at the end of the task execution. Therefore, other than the overhead imposed by re-executing the task, the detection comes with additional overhead dt_v. In case of re-execution, all the stateful values are rolled-back to the initial state and the same task instance is executed again. In Figure 1(a) v_1 of a simple producer(v_0)-consumer(v_1) application is hardened by re-execution, that is, $h(v_1) = (1, 0, 0, F)$. In this case, the task graph topology remains unchanged, but the task $wcet_v$ is modified to

$$wcet'_v = (wcet_v + dt_v) \times (k + 1) \tag{1}$$

with k being the maximum number of re-executions.

Replication: Replication uses multiple instances of the hardened task mapped on different cores. The degree of replication is usually larger than two, to enable majority voting. For a task that is only duplicated, only detection is possible which is the use case of [6]. Contrary to re-execution, replication will modify the task graph topology. The replication is called *active* if all replicated tasks are always executed at runtime. Task v_0 is actively triplicated in Figure 1(a), that is, $h(v_0) = (0, 3, 0, T)$.

In *passive* replication, not all cloned tasks are proactively instantiated, but only on request of the voter. This is particularly beneficial when the system is to be optimized to minimize average utilization or average power dissipation. Let us take the example in Figure 1(c) where $h(v_0) = h(v_1) = (0, 2, 1, T)$. $v_{*,0}$ and $v_{*,1}$, are actively duplicated. Then, when a faulty situation is detected by the voter, a third replica $v_{*,2}$ is instantiated to break the tie (highlighted with dashed arrows in the figure).

Selective voter placement: Replication scheme can further be optimized by judicious voter placement. In some cases, two (or more) consecutive tasks can be clustered and replicated in a group, instead of individually. By doing so, less resources are needed for voting, and the overhead of the replicated communication channels is reduced. Suppose the example given in Figure 1(d), where v_0 and v_1 are grouped and replicated together, sharing the voter vt_{01}. Compared with a naive voter placement with a similar h function shown in Figure 1(b), three communication channels are reduced, by removing redundant copies before/after the voter. The voting overhead is also reduced by half.

However, the grouped voter placement may come with reliability loss with respect to the naive strategy, as the existence of faults is checked less frequently. For instance, supposing that two faults are happening in the same iteration on $v_{0,1}$ and $v_{1,0}$, respectively, the naive approach is safe, as it separately detects the two faults. In the case of a grouped voter placement, this cannot be tolerated since two (out of three) input values to the voter vt_{01} are defective. Another possible drawback is in terms of re-execution latency. If a fault is detected at vt_1 (Figure 1(c)), the execution of $v_{1,2}$ is enough. But if v_0 does not have a voter in Figure 1(c), we would have to restart from $v_{0,2}$ since it is not clear where exactly the fault happened.

C. Problem Definition

Inputs: Given a target architecture $\mathcal{A}$ and a set of applications $\mathcal{T}$ with mixed-criticality requirements,

Outputs: determine a hardening technique h that results in a modified application $\mathcal{T}'$, and for each v such that $v \in V_t$ and $t \in \mathcal{T}'$ determine a mapping map, where $map : V \to P$ is mapping tasks to processors, with $V = \bigcup_{t \in \mathcal{T}'} V_t$ and $P = \bigcup_{cl \in CL} P_{cl}$.

Constraints: The output values are valid as long as bandwidth, schedulability, and reliability constraints are all satisfied. Due to space limitation, only the intra-cluster communication bandwidth constraint is described: For all edges $e \in E = \bigcup_{t \in \mathcal{T}} E_t$ such that $map(src_e) \neq map(dst_e)$ and $map(dst_e) \cup map(src_e) \in cl$, the following equation holds:

$$\sum s_e / pr_{src_e} \leq bw_{cl}. \tag{2}$$

Objectives: The proposed technique is not specific to a certain objective, thus any formulatable objectives can be minimized/maximized. In this work, we minimize the average power consumption, i.e., $minimize \ \{\sum_{p \in P} (stat_p + dyn_p \cdot u_p)\}$, with $u_p = \sum_{\forall v,t,v \in V_t \wedge map(v)=p} acet_v / pr_t$ the average utilization of processor p.

III. Evaluation of Reliability and Schedulability

In this section, we explain how reliability and schedulability constraints are evaluated in the proposed mapping optimization framework. [2]

A. Reliability Quantification

The probability that a task v running on processor p experiences a failure $fr(v)$ is denoted as:

$$fr(v) = wcet_v \cdot \lambda_v \tag{3}$$

That is, the processor executing task v has a constant failure rate λ_v, and the longer the execution time is, the more vulnerable to faults the system is. When more than one tasks of a task graph run concurrently, the expected failure probability per unit time is the sum of individual probabilities. That is, supposing $S(v) \subset V_t$ be a task set that can be executed concurrently with $v \in V_t$, $\sum_{v_i \in S(v) \cup \{v\}} fr(v_i) \leq f_t$ must hold to satisfy the reliability constraint of task graph t. How to compute $fr(v)$ with hardening techniques is explained in what follows.

Re-execution: If task v is hardened by re-execution of degree k, that is, it can be re-executed on the same core at most k times, the failure probability of v becomes:

$$fr_{rx,k}(v) = ((wcet_v + dt_v) \cdot \lambda_v)^{k+1} \tag{4}$$

Additionally, the timing overhead dt_v is considered for detection, state storing/restoring, and rolling-back. Compared to (3), the system goes faulty only if all $k+1$ trials are faulty.

Replication: When task v is hardened by active replication of degree k, a system-level failure occurs when more than half of the replicas, i.e., $k' = \lceil k/2 \rceil$ replicas, are faulty. In other words, a set of faults of cardinality less than k' can be tolerated by this hardening technique. The failure rate $fr_{rp,k}$ can be calculated as follows:

$$fr_{rp,k}(v) \quad = \quad \sum_i \left\{ \prod_{j \in H_i} wcet_{v_j} \cdot \lambda_{v_j} \right\} + ve_v \cdot \lambda_{vt_v} \tag{5}$$

[2]For brevity, we use simplified terms $wcet_v$, dt_v, ve_v, and λ_v instead of $wcet_v(type_{map(v)})$, $dt_v(type_{map(v)})$, $ve_v(type_{map(v)})$, $\lambda_{map(v)}$, respectively, omitting the index of mapped processors in what follows.

with ve_v being the accumulated overhead for the voter vt_v and H_i is defined as a set of indices of k' or more faulty replicated instances of the original task v.

Two failure scenarios are possible in this case. In one scenario, k' or more replicas go out of order due to faults. The first term of the equation enumerates all the combinations of k' faulty replicas. If $k = 3$ and $k' = 2$, for instance, all possible cases are $H_1 = \{1, 2\}$, $H_2 = \{1, 3\}$, $H_3 = \{2, 3\}$, and $H_4 = \{1, 2, 3\}$. The second scenario is when the voter task itself goes defective, which is taken into consideration in the second term.

We also consider passive replication, where the failure rate is computed with (5), similar to active replication. Only that, in passive replication, a task v is hardened with $k1$ active replicas and $k2$ passive replicas. A failure happens if either half of the total active or passive replicas go out of order or there is a fault in the voter. Therefore, for passive replication, we use equation (5) with $k' = \lceil (k1 + k2)/2 \rceil$, which in fact gives a conservative bound of failure rate.

B. Schedulability Analysis

The fault-management technique chosen for each task to enhance the reliability may cause uncertain behaviors. If a task is passively replicated, for instance, not all replicas are always invoked but only on request of the voter. These conditional executions complicate the scheduling analysis. Moreover, due to the well-known *scheduling anomaly* [11], it cannot be naively assumed that all the replicas are invoked every time for the worst case. This issue can be addressed by using existing scheduling analysis techniques that support variable task execution times [11], [12]. The original WCET of a task v is modified to $wcet'_v$ as shown in Equation (1) in case of re-execution, while the behavior of passively replicated tasks can be modeled by allowing zero BCET. Though we adopt a holistic scheduling analysis technique proposed in [12], it can be replaced with any existing technique as long as the variable execution time is supported.

IV. Evaluation and Conclusion

We show the effectiveness of the proposed technique with two synthetic (*Synth-1,Synth-2*) and three real-life examples. A cruise control application, *Control* is taken from [13] and enriched with three additional synthetic applications, *Synth-a/b/c*, to increase the benchmark complexity. Two more control benchmarks, "medium/large distributed non-preemptive real-time CORBA application" (*DT-med/large*) are borrowed from [14]. In order to impose more complexity and uncertainty, the invocation period and execution time of constituent tasks are enlarged by 20 times for *DT-med/large*. The SIL standard, defined in IEC 61508 [2], is used to specify the criticality levels of the applications. It contains four levels with level 4 being the most dependable ($10^{-8} \sim 10^{-9}$ failures per hour) and level 1 being the least ($10^{-5} \sim 10^{-6}$ failures per hour). We assume that the target multi-core architecture has 4-16 processors on 2-8 clusters. The detection overhead dt for all tasks is set to 10 ms, while the voting overhead ve varies from

TABLE I: Power dissipation of different hardenings [W]

	Synth-1	*Synth-2*	*DT-med*	*DT-large*	*Cruise*
Base	3.39	3.64	4.62	5.41	9.56
BaseP	3.39	3.62	4.54	4.93	8.64
BasePV	**2.71**	**3.57**	**3.95**	**4.68**	**7.92**

1 to 2 ms excluding communication overhead. The fault rate λ of the processors is calculated for the 50 nm technology according to [10] as 868 FIT (Failure-in-Time, $i.e.$, expected failures for one billion device-hours of operation). The power dissipation of the processor is $stat_p = 0.5W$ and $dyn_p = 0.9W$. Since the defined problem is NP complete, we adopt an Evolutionary Algorithm (EA) based optimization.

a) Effects of Passive Replication and Selective Voter Placement: Effects of passive replication and selective voter placement are shown separately in the comparison. Previous approaches [4], [5], [7] that only consider re-execution, active replication, and the combination thereof are set as a baseline *Base* in Table I. Passive replication is additionally considered in *BaseP*. Finally, all possibilities including passive replication and selective voter placement of the proposed technique are included in *BasePV*. *BasePV* always outperforms the others, demonstrating once more that considering passive replication and selective voter placement at the same time is beneficial. In the table, it is also shown that *Synth-1* and *Synth-2* do not take advantage of the passive replication in *BaseP*, since it has very tight latency constraints. And for such a case, the passive replication and re-execution do not show much additional benefit.

b) Impact of Various Reliability Levels: As explained so far, the quantitative reliability model proposed in this paper leads to an efficient design, enabling the distinction between several mixed reliability levels. To illustrate this fact, different reliability requirements were assigned to the four applications in the *Cruise* benchmark. For this experiment, all of hardening techniques are deployed. When the optimization does not include any reliability concerns (i.e., all applications have the lowest reliability requirement level, SIL 1), the optimized average power dissipation was 1.96 W. On the contrary, when the system is maximally enhanced for reliability with all applications having SIL 4, the power dissipation was 9.56 W. In this case, the system hardening was done at the cost of about 7.6 W. Leveraging on the proposed quantification model, we assign different levels to the four applications, i.e., $(4, 3, 2, 1)$ for *Control*, *Synth-a*, *Synth-b*, and *Synth-c*, respectively. In such a mixed criticality configuration, the optimized power was 7.92W, the cost of system hardening being reduced by 1.64 W. In absence of faults, the WCRT of *Control* is 406 ms, while WCRT with faults is 576 ms when the deadline is given as 600 ms. Note that all of hardening techniques, including re-execution, active replication, and passive replication, are properly used with selective voter placements.

c) Conclusion: In this paper, we propose a reliability-aware mapping optimization technique for multi-core systems with mixed criticality. To enable a comparative and quantitative evaluation of the optimized mapping, a probability based reliability metric is proposed. In order to have more general and complete coverage of hardening techniques, passive replication and selective voter placement are judiciously adopted with respect to the given constraints. The proposed reliability model is verified with existing standards and real-life examples. Experimental results prove the effectiveness of the proposed approach.

V. Acknowledgements

This work was supported by EU FP7 projects EURETILE and CERTAINTY under grant numbers 247846 and 288175, the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2013-H0301-13-1011), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0023325).

References

[1] Y. Xiang *et al.*, "System-level reliability modeling for mpsocs," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, 2010, pp. 297 – 306.

[2] H. Gall, "Functional safety iec 61508 / iec 61511 the impact to certification and the user," in *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, ser. AICCSA '08, Washington, DC, USA: IEEE Computer Society, 2008, pp. 1027–1031.

[3] International Organization for Standardization, *International Standard 26262: Road vehicles Functional safety. International Standard*, First Edition. 2011.

[4] P. Pop *et al.*, "Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 389 –402, 2009.

[5] C. Bolchini *et al.*, "Reliability-driven system-level synthesis for mixed-critical embedded systems," *Computers, IEEE Transactions on*, vol. 62, no. 12, pp. 2489–2502, 2013.

[6] P. Axer *et al.*, "Reliability analysis for mpsocs with mixed-critical, hard real-time constraints," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2011 Proceedings of the 9th International Conference on*, 2011, pp. 149 –158.

[7] P. v. Stralen *et al.*, "A safe approach towards early design space exploration of fault-tolerant multimedia mpsocs," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, ser. CODES+ISSS '12, Tampere, Finland: ACM, 2012, pp. 393–402.

[8] H. Kopetz *et al.*, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[9] A. Sanyal *et al.*, "An improved soft-error rate measurement technique," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 4, pp. 596–600, 2009.

[10] P. Shivakumar *et al.*, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on*, 2002, pp. 389–398.

[11] S. Perathoner *et al.*, "Evaluation and comparison of performance analysis methods for distributed embedded systems," *Master's thesis, Swiss Federal Institute of Technology, Zürich*, 2006.

[12] J. Kim *et al.*, "A novel analytical method for worst case response time estimation of distributed embedded systems," in *Proceedings of the 50th Annual Design Automation Conference*, ser. DAC '13, Austin, Texas: ACM, 2013, 129:1–129:10.

[13] N. Kandasamy *et al.*, "Dependable communication synthesis for distributed embedded systems," in *Computer Safety, Reliability, and Security*, Springer, 2003, pp. 275–288.

[14] G. Madl *et al.*, "Tutorial for the open-source dream tool," *Univ. California, Irvine, CA, CECS Tech. Rep*, 2006.