Mitigating the Impact of Hardware Failures on Multimedia Applications – A Cross-Layer Approach

ABSTRACT
Incorporating reliability in resource-limited mobile devices poses significant challenges due to the high overheads of power and performance. Using the observation that error-awareness at one abstraction layer can be exploited to enhance end-to-end reliability at other abstraction layers, this paper proposes a cross-layer method exploiting error control schemes across layers. In particular, a cross-layer approach is presented to combat hardware defects at minimal costs of power and performance by using error control schemes in multimedia applications. We propose a cross-layer, error-aware method that exploits dual approach – a Drop and Forward Recovery (DFR) and an error-resilient video encoding – driven by intelligent middleware schemes for mitigating the impact of soft errors. Our cross-layer approach enables exploration of feasible solutions satisfying minimal reliability with minimal overheads of power and performance, and additionally opens up a new design space yielding solutions that can tradeoff QoS for reliability. Experimental evaluation demonstrates that our cross-layer, error-aware method for a video encoding with video sequences improves performance by 60% and the energy consumption by 58% with even better reliability at the cost of 3% quality degradation on average, as compared to an ECC-based hardware protection technique.

1. MOTIVATION
With advances in processor and wireless communication technologies, mobile devices such as PDA and smart phones have emerged as a main component for a range of multimedia applications such as video telephony and remote image sensing. Challenges to cope with error-prone transmission over wireless networks bring out the need for error-resilient techniques, and they must be implemented in an energy-efficient way to prolong the life of battery-powered mobile devices. Note that the main objective of error-resilient techniques, e.g., an error-resilient video encoding, is to recover the erroneous video data due to transmission errors for maintaining the video quality.

On the other hand, soft errors or transient faults are becoming a critical concern as technology scales [2, 11, 35]. The primary source of soft errors in digital circuits are cosmic radiation. The occurrence of soft errors is directly proportional to the exposed area of the logic [5]. Since caches are one of the largest area contributors in processors, they are most vulnerable to soft errors. Due to incessant technology scaling (voltage scaling and critical dimension scaling), soft error rate (SER) has exponentially increased [11, 35]. And in emerging pervasive computing environments, systems will be exposed to the environments with drastic increase of radiation intensity [10]. For example, SER in an airplane can be worse than that on the ground by at least a couple of orders of magnitude [20]. Now it has reached a point, where it has become a real threat to system reliability. Indeed, a single soft error may result in a catastrophic system failure.

Solutions to reduce soft errors have been proposed at all levels of design hierarchy from hardening devices [3, 27] to error control schemes such as TMR (Triple Modular Redundance) [26] and ECC (Error Correction Codes) [26]. However, these techniques incur high overheads in terms of power and performance [18, 19]. For example, TMR typically uses three functionally equivalent replicas of a logic circuit and a majority voter, but the overheads of hardware and power for conventional TMR exceed 200% [24]. Also, implementing an ECC-based scheme, the most popular one in memory systems, raises access time by up to 95% [18] and power consumption by up to 22% [25] in the caches. To reduce the overheads, several microarchitectural solutions [17, 19, 23] have been investigated but still incur overheads in terms of power and performance.

On the other hand, an EDC (Error Detection Codes) based technique such as parity codes [26] incurs much less overheads than an ECC-based technique such as a Hamming code [26]. For example, a parity code decreases the access delay by up to 47% and the power consumption by up to 73% compared to a Hamming code (38,6) according to [19]. However, an EDC scheme can only detect an error while an ECC can even correct it. Thus, to make a system tolerant against soft errors, an EDC scheme must be followed by an error recovery technique such as rolling-backward recovery with checkpoints [26]. Checkpoints are made every interval and the system states are saved at the reliable storage. Once an error is detected, the system rolls backward to the last saved checkpoint and re-processes the functionality after a recovery – backward error recovery (BER). However, recovery with checkpoints is inappropriate for real-time applications since they have in general poor predictability of the completion time – a checkpoint interval will be lost whenever an error is detected, and more intervals may be lost on the occasion of multiple error occurrences.

This paper studies low-cost approaches to mitigate the impacts of soft errors at data caches in mobile video encoding system. Figure 1 shows our system model – a mobile video encoding system. A mobile video encoding system consists of several abstraction layers such as application, middleware, OS, and hardware layer. And the mobile video encoding system transmits video data through the wireless network as shown in Figure 1. Due to intensive complexity of processing algorithms and
large amount of data transmission, it is a challenging task to satisfy multiple constraints such as energy consumption and QoS that mobile video systems demand on battery-operated mobile devices. One of promising approaches to balance these multiple constraints is a cross-layer method. With the global view of the whole system, the cross-layer methods obtain the maximal power reduction with the satisfactory QoS (GRACE) [9, 38], present a proxy-based middleware approach (DYNAMO) by trading off video QoS [7, 8, 21, 22], and recently study online timing-QoS verification at the proxy server (xTune) [14]. To the best of our knowledge, no efforts have studied the cross-layer interactions and potential cooperations among error control schemes to maintain multiple constraints in resource-constrained mobile devices.

In this work, we analyze different error models and error control schemes across abstraction layers, and observe that they have different impacts on reliability and QoS. Soft errors can degrade not only the video quality but also reliability while packet losses cause quality degradation, but not system failures such as system crash, infinite loop, memory violation, etc. Thus, we present a cross-layer, error-aware method in mobile devices so that it protects hardware components such as data caches from soft errors for satisfactory QoS and reliability with minimal costs of power and performance. To mitigate impacts of soft errors on data caches for low costs, we wisely exploit an error-resilient video encoding and a DFR (Drop and Forward Recovery) mechanism with an EDC protection in mobile video encoding systems. Since TMR and ECC incur overheads of power and performance, less expensive EDC is implemented in the previously proposed PPC (Partially Protected Caches) [17] at the hardware layer. A DFR mechanism is selected for the reliability improvement rather than a backward error recovery (BER) once an error is monitored at the middleware layer. Note that a DFR mechanism is named from a simple error concealment scheme in video decodings so that it drops a lossy frame due to transmission errors, and reconstructs it by making use of available data such as adjacent frames [33]. Thus, this DFR mechanism skips the intensive processing algorithms and moves forward to the next frame encoding. This moving-forward is the difference between a DFR and a BER. One potential problem of a DFR mechanism can degrade the video quality since it drops a frame once an error is detected. However, error-resilient techniques in video encodings are wisely exploited to mitigate the impacts of soft errors on the video quality by considering these errors as packet losses. In order to use schemes from other abstraction layers, there are needs for middleware, which allows cross-layer trade-offs. Middleware translates an error metric at the hardware layer to a different error metric, which is necessary for the use of an error control scheme at the application layer. For example, SER (Soft Error Rate) at the hardware layer is translated into FLR (Frame Loss Rate), which will be provided to an error-resilient encoding at the application layer. Also, middleware assists a DFR mechanism, and selects a policy based on available information for further improvements.

The contributions and results of our work are:

- We propose a cross-layer, error-aware method so that both performance and energy costs are minimized while obtaining high reliability at the minimal degradation of QoS.
- Our cross-layer method exploiting an error-resilient video encoding and a DFR mechanism with a PPC architecture does not incur overheads in terms of power and performance. Rather, our proposal reduces the access latency of memory subsystem by 36%, the energy consumption of memory subsystem by 47%, the failure rate by 250× at the cost of less than 1 dB of video quality, compared to a traditional video encoding running on a data cache without protection.
- Our cross-layer method extends the applicability of existing error control schemes at the application abstraction layer to mitigate the impact of hardware defects at the hardware abstraction layer.
- To assist our cross-layer methods, we present the middleware that triggers error control schemes with an appropriate error translation, and selects a recovery policy based on available information in a mobile multimedia system.

2. CROSS-LAYER, ERROR-AWARE METHOD

2.1 System Model and Problem Definition

Figure 1 shows our cross-layered system model of video encoding. Note that multimedia applications such as video streaming and conferencing applications require timeliness and content-correctness at the practical level. In general, they require soft real-time constraint (not hard real-time) and accept slight quality degradation. For example, missing deadlines in video streaming applications results in the service delay, and losing packets degrades the video quality. These practical level of QoS degradations are acceptable by end-users or unnoticeable by the human eye. Note that errors in networks may degrade the QoS, not system failures such as mobile system crash.

Figure 2 shows error-resilient multimedia applications robust against errors in the error-prone network. For example, video encodings can be error-resilient. Mainly, error-resilient techniques in multimedia applications recover the lost multimedia data for the quality improvement. Thus, they deal with multimedia data such as input or output data, which are vulnerable due to error-prone networks. We define this multimedia data as external data as shown in Figure 2 with the view of the mobile device. On the contrary, internal data is defined as data, program codes, etc. residing inside the mobile device during the process of the functionality. Note that external data can be internal data since input is the source for the processing and output resides temporarily inside the mobile device before being transmitted outside for the further usage. It is important that errors on internal data cause not only QoS degradation but...
also system failures. For example, errors on program variables may result in memory segmentation violation, which is a system failure. Hardware defects such as soft errors can occur on internal data, which has different characteristics from network errors on external data as summarized in Table 1.

Table 1 presents different error models and error control schemes at the application and hardware abstraction layers in a mobile multimedia system. Note that we do not consider any software errors such as bugs at application, middleware, and OS layers, nor permanent failures at the hardware layer in this paper. In general, data losses such as packet losses induced from the error-prone network only cause the quality degradation while defects induced at the hardware layer, e.g., at data caches or logic components, not only cause the quality degradation but also cause the failure of the system. These different impacts are mainly because errors at the hardware layer occurs on internal data, especially on control data, which can result in system crash, infinite loop, and memory segmentation fault. Thus, protection techniques at the application layer have been developed to mainly improve the delivered quality. For example, error-resilient schemes mitigate the effects of packet losses on the QoS [6, 15]. On the other hand, error control schemes for hardware defects such as soft errors must consider both reliability and QoS in mobile multimedia systems. Note that traditional protection techniques implemented at the hardware layer incurs significant overheads in terms of power, performance, and yield cost. There is a potential possibility that error-resilient schemes at the application layer is used for mitigating hardware defects on the QoS, but how? Table 1 shows that SER should be understandable by error-resilient applications to combat those soft errors. Further, it is not enough unless error control schemes can combat errors with respect to reliability. Therefore, exploiting error control schemes across abstraction layers is an interesting challenge since they must consider different types of errors, data, impacts, and error measures as shown in Table 1. By being aware of error specifics and error control schemes, our system can make use of them in a cross-layered manner for obtaining low-cost reliability while maintaining the QoS.

In this context, our main problem is how to compose error-protection schemes across layers, and objective is to develop cross-layer methods that are capable of: (i) minimizing the overheads of power and performance, (ii) satisfying the QoS requirement, and (iii) achieving the same level of fault tolerance as traditional error protection techniques. In particular, we investigate how to exploit an error-resilient video encoding and a DFR mechanism for mitigating soft errors in a cross-layered manner to obtain the low-cost reliability while satisfying the video quality.

## 2.2 Cross-Layer Method

Cross-layer methods have demonstrated the effectiveness compared to schemes isolated at a single abstraction layer [7, 8, 9, 14, 21, 22, 38]. Yuan et al. in [37] proposed an energy-efficient real-time scheduler (GRACE-OS) based on statistical distribution of application cycle demands, and presented a practical voltage scaling algorithm (PDVS) [38] to coordinate adaptation of multimedia applications and CPU speeds for mobile multimedia systems. Mopy et al. in [21] presented an integrated power management technique considering hardware-level power optimization and middleware-level adaptation to minimize the energy consumption while maintaining user experience of video quality in mobile video applications. Recently, Kim et al. in [14] proposed a unified framework that employs on-the-fly formal verification combined with iterative parameter tuning for continuous cross-layer adaptations for distributed real-time systems.

Cross-layer methods in the OSI (Open Systems Interconnection) reference model have been widely investigated as a promising optimization tools to efficiently reduce the energy consumption, especially transmission energy consumption, in wireless multimedia communications [1, 32, 31]. Vuran et al. in [32] presented a cross-layer methodology to analyze error control schemes with respect to transmission power and end-to-end latency, especially impacts of routing, medium access, and physical levels in wireless sensor networks. Schaar et al. in [31] proposed a joint cross-layer approach of application-layer packetization and MAC-layer retransmission strategy, and developed on-the-fly adaptive algorithms to improve the video quality under the bandwidth and delay constraint for wireless multimedia transmission. Bajic in [1] developed cross-layer error control schemes considering joint source rate selection and power management for wireless video multicast.

Our cross-layer, error-aware approach has two unique aspects compared to cross-layer error control schemes in the OSI reference model. First, our error-aware scheme is implemented at a different abstraction model in mobile multimedia devices as shown in Figure 2. Secondly, we exploit actively existing error control schemes at one abstraction layer to combat errors at a different abstraction layer, which has different specifics from originally considered errors.

To evaluate our cross-layer approach, we consider a simplified system consisting of a video encoding application and a data cache. A video encoding application can be error-prone or error-resilient, and similarly a data cache can be error-prone or error-protected as shown in Figure 2. Any composition of cross-products from them has pros and cons with respect to performance, power, QoS, and reliability. For example, an error-prone video encoding running on an error-protected data cache suffers from high failures due to no protection at data cache against soft errors. While an error-prone video encoding on an error-protected data cache improves the video quality as well as the reliability, it incurs high overheads in terms of power and performance. An error-resilient video encoding on an error-prone data cache may increase the video quality, but fail to increase the reliability. An error-resilient video encoding running on an error-protected data cache is possibly of over protection on the QoS since it incurs high overheads due to expensive protection. Approaches unaware of errors on different data at different abstraction layers may result in inefficiency in terms of power, performance, QoS, and/or reliability.

**Cross-Layer, Error-Aware Method (Our Proposal)** Our proposal is aware of different data, error control schemes, and impacts across layers. Thus, our cross-layer, error-aware method mitigates hardware defects such as soft errors using error-resilience and a DFR mechanism for maximal reliability with minimal overheads of power and performance at the cost of minimal quality degradation. Our approach exploits an error-resilient video encoding and a DFR mechanism to combat soft errors at a data cache, where a less expensive EDC scheme protects only non-multimedia data instead of using an ECC scheme. All these schemes are energy and performance efficient. The DFR mechanism is shown in Figure ??, and drops a current encoding frame, for example, and moves
forward to the next frame encoding once an error is detected in a mobile video encoding system. Note that a BER mechanism rolls backward and re-encode the current frame once an error is detected as shown in Figure 2.1. Thus, this DFR mechanism with an EDC scheme improves power and performance significantly in several directions and increases reliability as well. First, dropping an erroneous frame potentially improves performance and energy reduction due to skipping the expensive processing algorithms. Secondly, EDC is less expensive than ECC [19] and overheads due to checkpoints are negligible [36] while EDC can be as immune to soft errors as ECC with respect to reliability. The potential problem due to DFR is the quality degradation (it drops an erroneous frame), which is recovered by wisely exploiting error-resilience in multimedia applications. One of the main differences between errors on internal data and those on external data is the impact of internal errors on the failures. This reliability issue is managed by the less expensive technique such as an EDC scheme incorporated with a DFR, and the quality is managed by existing error-resilient techniques.

In summary, our cross-layer, error-aware method exploits existing techniques at one abstraction layer to solve problems at the other abstraction layer without incurring significant overheads of power and performance while satisfying the reliability and QoS requirements.

3. EXPERIMENTAL EVALUATION

3.1 Simulation Setup

![Figure 3: Experimental Setup](image)

To demonstrate the effectiveness of our cross-layer, error-aware method, a simulation framework has been built to evaluate several system compositions in terms of performance, power, reliability, and QoS as shown in Figure 3. We study a simplified video encoding system consisting of a video encoding at the application layer and a data cache at the hardware layer.

We study a H.263-based error-prone video encoding and error-resilient video encoding as shown in Figure 3. For an error-prone video encoding, we use a GOP (Group-Of-Picture) encoder [6]. For GOP, all frames encoded as P-frames except for the first frame (I-frame), and the quantization scale is set to 10. PBPAIR [15] is used as an error-resistant video encoding, and takes two parameters such as intra_threshold and PLR. In this study, we consider that PLR from the network is zero to isolate the impacts of the hardware defects such as soft errors. Intra_threshold is selected through the original method of PBPAIR to generate the similar impacts of the hardware defects such as soft errors. Intra_threshold is set to 10. PBPAIR [15] is used as an error-resilient video encoding system. Note that a BER mechanism rolls backward and re-encode the current frame once an error is detected as shown in Figure 2.1. Thus, this DFR mechanism with an EDC scheme improves power and performance significantly in several directions and increases reliability as well. First, dropping an erroneous frame potentially improves performance and energy reduction due to skipping the expensive processing algorithms. Secondly, EDC is less expensive than ECC [19] and overheads due to checkpoints are negligible [36] while EDC can be as immune to soft errors as ECC with respect to reliability. The potential problem due to DFR is the quality degradation (it drops an erroneous frame), which is recovered by wisely exploiting error-resilience in multimedia applications. One of the main differences between errors on internal data and those on external data is the impact of internal errors on the failures. This reliability issue is managed by the less expensive technique such as an EDC scheme incorporated with a DFR, and the quality is managed by existing error-resilient techniques.

In summary, our cross-layer, error-aware method exploits existing techniques at one abstraction layer to solve problems at the other abstraction layer without incurring significant overheads of power and performance while satisfying the reliability and QoS requirements.

3.2 System Composition

A video encoding can be GOP (error-prone) or PBPAIR (error-resilient). And a data cache can be a unprotected cache (error-prone) or a PPC (error-protected). Thus, our study investigates the following cross-product system compositions from them:

- Base Composition: GOP and a unprotected data cache
- HW-Protection Composition: GOP and a PPC
- App-Protection Composition: PBPAIR and a unprotected data cache
- All-Protection Composition: PBPAIR and a PPC
- Cross-Layer Composition (Our Proposal): PBPAIR and a PPC with cross-layer mechanisms

3.2.1 Base Composition - No Protection

Base composition is evaluated for comparison in our study. It does not incur overheads for protection in terms of power and performance while it causes high failure rate and low video quality due to no protection on internal data, especially on control data (non-multimedia data) in internal data, at a data cache from soft errors.

3.2.2 HW-Protection Composition

Since HW-Protection composition protects a data cache from soft errors, it presents the low failure rate and high QoS. However, it incurs overheads in terms of power and performance. We consider two types of protection for a protected data cache in a PPC: (i) an ECC scheme such as a Hamming Code (38,6) for FER recovery, and (ii) an EDC scheme such as a parity code for BER or DFR recovery.
3.2.3 App-Protection Composition

A video encoding PBPAIR improves the quality against transmission errors in the network but it is unaware of soft errors on a data cache. A data cache is of no protection for soft errors. This composition does not incur overheads in terms of power and performance for protecting internal data at a data cache. Interestingly, PBPAIR can improve the quality of service occasionally, e.g., the anticipated degree of error-resilience is more than the real degree of error-proneness in the network, and can be unintentionally used for recovering the lost data due to soft errors. However, PBPAIR and an unprotected data cache are disconnected, so it is possible to process video data without error-resilience in case of error-free networks, for instance. Further, the error-resilient algorithms in PBPAIR are developed to combat errors on external data for the quality improvement, thus they fail to protect data cache from failures due to soft errors onto control data in internal data such as loop variables and conditional variables.

3.2.4 All-Protection Composition

All-Protection composition protects both a data cache and a video encoding from its own errors. All-Protection composition consists of PBPAIR and a PPC with an ECC-based protection. Hardware protection techniques unaware of the feature of error-resilience in multimedia applications incur unnecessary overheads in terms of power and performance while decreasing the failure rate and the quality degradation. On the other hand, PBPAIR possibly improves the quality of video due to soft errors in a data cache, which still remain after the data cache protection (e.g., double-bit errors if it can protect only single-bit errors).

3.2.5 Cross-Layer Composition (Our Proposal)

Our cross-layer composition consists of PBPAIR and a PPC with an EDC-based scheme. Further, cross-layer error-aware methods exploits a DFR mechanism to improve the reliability once an error is monitored at the protected cache in a PPC. Thus, cross-layer composition achieves high reliability with minimal costs of power and performance at the slight degradation of QoS.

In order to enhance the video quality from frequent DFRs, we present Slack-Aware DFR/BER, Frame-Aware DFR/BER, and QoS-Aware DFR/BER. For evaluation, we consider naive DFR, naive BER, no DFR/BER, and random DFR/BER for our selective schemes. Naive DFR and naive BER always trigger a DFR mechanism and a BER mechanism, respectively, once an error is detected. No DFR/BER implies a mechanism that a system lets an application keep running even though a soft error is detected to observe the impacts of soft errors on failures. Random DFR/BER selects DFR or BER based on randomly generated number, i.e., if the generated number is less than 50 out of 100, it goes with BER. For Frame-Aware DFR/BER, Slack-Aware DFR/BER, and QoS-Aware DFR/BER, we profiled preliminary simulations, and selected parameters for each selective scheme in our simulations.

3.3 Evaluation Metrics

3.3.1 Performance

For performance comparison of each composition, we estimate the access latency to the memory subsystem using the statistics generated by the cache simulator as shown in Figure 3. Note that our performance model using a functional simulator sim-cache has been compared to runtime using a cycle accurate processor simulator sim-outorder, and the relative ratio between them maintains constant, but sim-cache is much faster. The access latency of a composition, \( L \), is estimated as

\[
L = L_{acache} \times L_{access} + M_{cache} \times L_{miss} + N_{policy} \times L_{policy}
\]

where \( L_{acache} \) is the number of accesses to a cache, \( L_{access} \) is the cache access penalty, \( M_{cache} \) is the number of misses to a cache, \( L_{miss} \) is the cache miss penalty, i.e., the access penalty to a bus and a memory, \( N_{policy} \) is the number of triggered policies such as DFR and BER, and \( L_{policy} \) is the latency penalty for a policy. Table 2 summarizes access and miss penalties of a data cache in cycles with or without a protection. The overhead of delay for ECC is estimated and synthesized using the CACTI [28] and the Synopsys Design Compiler [30] as in [17], and the overhead of delay for DFR is calculated using the ratio between delays of ECC and DFR from [17, 19]. Also, the delay overheads for DFR and BER are estimated through the simulations so that the overheads for context switch and checkpoints are added at the analysis stage in our simulation study as shown in Figure 3.

3.3.2 Power

For power comparison, we estimate the energy consumption of the memory subsystem. To estimate the energy consumption of the memory subsystem, we use the power models presented in [29]. The overheads of power for a Hamming code (32,8) and a parity code are similarly synthesized and estimated to those of delay as in Section 3.3.1. The power consumption penalty for a recovery mechanism such as DFR and BER is estimated through the simulations. Using the energy models in Table 2, the energy consumption of the memory subsystem is

\[
E = A_{cache} \times P_{access} + M_{cache} \times P_{miss} + N_{policy} \times P_{policy}
\]

where \( P_{access} \) is the power penalty per cache access, \( P_{miss} \) is the power penalty per cache miss, \( P_{policy} \) is the power penalty for a policy.

3.3.3 QoS

For reliability evaluation, a failure rate is used. Each execution is defined as a Success if it ends within twice of a normal execution time and returns the correct output opened by a decoder. Otherwise, it is a Failure such as a system crash, infinite loop, or segmentation fault. Note that the degradation of video data is not considered as a failure. The failure rate has been obtained through at least hundreds of executions for each composition by counting the number of failures out of a total number of executions based on the following binomial distribution analysis.

Assuming that each execution is an independent Success/Failure event, \( X \) is the number of Success in executions, and we perform \( n \) runs, then if the probability of a failure is \( p \), then \( X \) is a binomially distributed random variable, which follows the binomial distribution with parameters \( n \) and \( p \) (\( X \approx \mathcal{B}(n, p) \)). Therefore, the mean of \( X \) is \( \mu = np \), and the variance of \( X \) will be \( \sigma^2 = np(1-p) \). The error \( E \) is \( |X - \mu| \) is less than or equal to \( z_\alpha/\sqrt{\mu} \) with confidence 100(1-\( \alpha \))%, where \( \alpha \) is a confidence level. Therefore, the sample size to be able to state with 100(1-\( \alpha \))% confidence that the error \( |X - \mu| \) will not exceed a specified amount \( E \) is

\[
N = \left( \frac{z_\alpha/\sqrt{\mu}}{E} \right)^2
\]

For 95% confidence, \( z_{0.025} = 2 \), and confidence interval \( 0.01 \%), the sample size is \( N = 40,000 p(1-p) \). To estimate the probability of a failure, suppose the probability of a failure is 1%, then \( N = 40,000 \times 0.01 \times 0.99 = 396 \).

3.3.4 QoS

For QoS comparison in video encoding, the video quality is measured in PSNR (Peak Signal to Noise Ratio). PSNR is defined in dB as

\[
PSNR = 10\log_{10} \left( \frac{MAX^2}{MSE} \right)
\]

where \( MAX \) is the maximum pixel value

\begin{table}[h]
\centering
\caption{Power and Delay Penalties}
\begin{tabular}{|c|c|c|c|c|}
\hline
Cache (size) & Protection & \( L_{access} \) (cycles) & \( P_{access} \) (nJoules) & \( L_{miss} \) (cycles) & \( P_{miss} \) (nJoules) \\
\hline
Unprotected (2 KB) & NO & 2 & 1.06 & 25 & 41.96 \\
Protected (2 KB) & ECC & 2 & 1.19 & 25 & 42.16 \\
Protected (2 KB) & EDC & 1.5 & 0.91 & 25 & 42.06 \\
Unprotected (2 KB) & NO & 1 & 0.80 & 25 & 41.96 \\
\hline
\end{tabular}
\end{table}
and $MSE$ is the Mean Squared Error, which is the mean of the square of differences between the pixel values of the erroneous video output due to soft errors and frame drops, and the correctly reconstructed output without errors.

4. EXPERIMENTAL RESULTS

We present two sets of results. First, we demonstrate the effectiveness of our cross-layer methods aware of error control schemes in terms of low-cost reliability at the slight cost of QoS (Section 4.1). Second, we show the effectiveness of intelligent DFRBER selection schemes to improve the video quality (Section 4.2).

4.1 Effectiveness of Cross-Layer, Error-Aware Methods

Figure 4 clearly shows that our cross-layer and error-aware scheme increases the reliability with the minimal costs of performance and energy consumption at the minimal degradation of video quality.

Figure 4(a) clearly demonstrates that our Cross-Layer composition (composed of PBPAIR, a DFR mechanism, and PPC with an EDC protection) improves the failure rate by more than two orders of magnitude than that of Base composition (GOP and unprotected data cache). This reliability improvement mainly because of the EDC protection with a DFR mechanism in Cross-Layer. While HW-Protection and All-Protection have lower failure rates than that of Base composition, Cross-Layer has lower failure rate than those of HW-Protection and All-Protection. This is because it has less time to be exposed to soft errors thanks to a frame drop, and thanks to performance-efficiency of PBPAIR than GOP. It is important that App-Protection composed of PBPAIR and an unprotected data cache shows the close failure rate to that of Base since a failure results from errors on internal data, specifically non-multimedia data or control data in internal data. Thus, our Cross-Layer scheme can achieve the best reliability among all compositions.

Figure 4(b) shows that our Cross-Layer composition is the best in terms of performance. It reduces the memory subsystem access delay by 58%, compared to that of Base composition. It is very effective since our Cross-Layer composition reduces the failure rate by a couple of hundreds times and it reduces the access latency of memory subsystem compared to Base. All the other compositions incur the performance overhead but Cross-Layer improves the performance. This performance improvement is because of skipping intensive compression algorithms thanks to a DFR mechanism and because of the performance efficiency of PBPAIR algorithms. However, the performance efficiency of PBPAIR is not well exploited in case of App-Protection composition as it shows 5% overhead compared to Base because PBPAIR increases the compression efficiency rather than the performance efficiency at low PLR such as 0% PLR. With the same reason, All-Protection incurs about 4% overhead compared to Base. HW-Protection does not incur the performance overhead because a PPC achieves promising performance efficiency by protecting only non-multimedia data [17]. Indeed, HW-Protection and All-Protection compositions reduce the access time of memory subsystem compared to Base and App-Protection compositions, respectively.

With the perspective of energy consumption of memory subsystem, our Cross-Layer composition saves the energy consumption by 49%.
Table 3: Experimental Results (Normalized Result of Each Composition to that of Base Composition for Different Video Sequences)

<table>
<thead>
<tr>
<th>Video Sequence</th>
<th>System Composition</th>
<th>Application</th>
<th>Policy</th>
<th>Hardware</th>
<th>Access Time</th>
<th>Energy Consumption</th>
<th>Failure Rate</th>
<th>Video Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKIYO (low activity)</td>
<td>Base</td>
<td>GOP</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>HW-Protection</td>
<td>GOP</td>
<td>FER</td>
<td>PPC with ECC</td>
<td>0.99</td>
<td>1.14</td>
<td>1.0E-2</td>
<td>1.02</td>
</tr>
<tr>
<td></td>
<td>App-Protection</td>
<td>PBPAIR</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>0.87</td>
<td>0.89</td>
<td>91.7E-2</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>Cross-Layer Protection</td>
<td>PBPAIR</td>
<td>DFR</td>
<td>PPC with EDC</td>
<td>0.27</td>
<td>0.34</td>
<td>0.7E-2</td>
<td>1.02</td>
</tr>
<tr>
<td>FOREMAN (medium activity)</td>
<td>Base</td>
<td>GOP</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>HW-Protection</td>
<td>GOP</td>
<td>FER</td>
<td>PPC with ECC</td>
<td>1.00</td>
<td>1.15</td>
<td>0.7E-2</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td>App-Protection</td>
<td>PBPAIR</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>1.05</td>
<td>1.06</td>
<td>74.8E-2</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>Cross-Layer Protection</td>
<td>PBPAIR</td>
<td>DFR</td>
<td>PPC with EDC</td>
<td>0.42</td>
<td>0.51</td>
<td>0.4E-2</td>
<td>0.99</td>
</tr>
<tr>
<td>COASTGUARD (high activity)</td>
<td>Base</td>
<td>GOP</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>HW-Protection</td>
<td>GOP</td>
<td>FER</td>
<td>PPC with ECC</td>
<td>0.99</td>
<td>1.14</td>
<td>1.4E-2</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>App-Protection</td>
<td>PBPAIR</td>
<td>NO</td>
<td>Unprotected Cache</td>
<td>1.09</td>
<td>1.10</td>
<td>77.6E-2</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>Cross-Layer Protection</td>
<td>PBPAIR</td>
<td>DFR</td>
<td>PPC with EDC</td>
<td>0.49</td>
<td>0.58</td>
<td>0.4E-2</td>
<td>0.93</td>
</tr>
</tbody>
</table>

In summary, our cross-layer, error-aware methods exploit a DFR mechanism with an inexpensive EDC protection to decrease the failure rate by 200×, and an error-resilient video encoding technique to minimize the quality degradation by 2% while significantly saving the access time by 61% and energy consumption by 52% on average over multiple video sequences, as compared to Base composition. Also, our cross-layer approach achieves a better reliability than a previously proposed PPC architecture with an ECC protection at the cost of 3% QoS degradation while reducing the access time by 60% and the energy consumption by 58% on average.

4.2 Effectiveness of Intelligent DFR Schemes

Our Cross-Layer outperforms all possible compositions in terms of power, performance, and reliability while it degrades the video quality mainly due to frame drops once soft errors occur. Figure 5 demonstrates that all intelligent selective schemes improve the video quality without incurring performance and energy costs significantly (still mostly lower than costs of Base composition).

In Figure 5, x-axis represents selective mechanisms compared to Base composition. Note that they are all running PBPAIR on a PPC architecture with an EDC scheme except for Base (running GOP on a unprotected cache) and no DFRBER (running PBPAIR on a PPC without any protection) for comparison. And we parameterize a policy selection based on available information in a mobile embedded system. Naïve DFR always selects a DFR mechanism while naïve BER selects a BER once a soft error occurs. This naïve DFR scheme shows the worst video quality as shown in Figure 5(d) at the minimal costs in terms of power and performance as shown in Figure 5(b) and Figure 5(c). Note that naïve DFR in Figure 5 results from multiple soft errors on the protected data cache in a PPC, which degrades the video quality worse than that of Cross-Layer composition in Figure 4(d). On the other hand, naive BER scheme presents the best video quality while incurring the most expensive power and performance costs. In terms of reliability, naïve BER shows higher failure rate than that of naïve DFR as shown in Figure 5(a). This is mainly because naïve BER increases the execution time, causing the more time for a PPC to be exposed to soft errors. Clearly, No DFR/BER does not have a mechanism to protect a system from soft errors, causing very high failure rate as shown in Figure 5(a). Random DFR/BER provides the best video quality with inexpensive power and performance. For SA-DFR/BER, the results have been profiled with the knob S, the portion of ACET, from 0% to 100% in 10% increments, and SA-DFR/BER with S = 80% is compared in Figure 5 since it is the least value of the knob to recover the video quality better than that of Base composition based on profiled results. However, it is an expensive approach since it incurs high overheads in terms of power and perfor-
For QA-DFR/BER, 31.8 dB is considered as the QoS threshold value since it has lower costs with better QoS (failure rates are close). Experiments show that FA-DFR/BER scheme is more effective than SA-DFR/BER and are unable to drive the whole system’s reliability in power and performance. Moreover, error-awareness introduces a new design margin to further save the resources with high reliability at the cost of QoS.

Our future work includes the extended cross-layer approach considering end-to-end devices in distributed real-time systems, and varying error control schemes with different error models across system abstraction layers.

6. REFERENCES

Figure 5: Intelligent selective schemes between DFR and BER improve the video quality with minimal costs

(a) Reliability: Failure Rate

(b) Performance: Memory Subsystem Access Time

(c) Power: Energy Consumption of Memory Subsystem

(d) QoS: Video Quality

Figure 5: Intelligent selective schemes between DFR and BER improve the video quality with minimal costs

The figure shows the performance metrics for different error resilience techniques. The x-axis represents the error resilience techniques, and the y-axis represents the video quality. The graphs illustrate the trade-offs between video quality and reliability, highlighting the benefits of using selective schemes between DFR and BER.


