

# Reliability Requirements of Control, Address, and Data Operations in Error-Tolerant Applications

Darshan D. Thaker †, Diana Franklin ‡, Venkatesh Akella †  
Frederic T. Chong★

†*University of California, Davis*  
‡*California Polytechnic State University*  
★*University of California, Santa Barbara*  
contact: ddthaker@ucdavis.edu

## Abstract

*Rising chip densities and decreasing voltages increasingly expose future processors to soft errors. The reliability of computations has become an architectural resource, giving rise to a tradeoff between increased complexity and less reliable results. Error-tolerant applications, such as the MPEG decoder, are an excellent match for this tradeoff [CM01]. We propose the R-bit micro-architecture, based upon a simple reliability tag bit that allows dynamic scheduling of instructions to functional units of differing reliability. In particular, we discover that it is important to differentiate control, address, and remaining data in our computations. We present preliminary data which illustrates how error rates in these three categories affect the fidelity of our MPEG decoder. We further show that our initial R-bit design protects control data and increases quality of decoder output.*

## 1 Introduction

As chip technology continues to shrink, it will become more difficult to perform reliable computation. In order to continue the development of inexpensive, fast, power-efficient chips, we will need to view reliability as a resource. In order to maintain reliability, we must pay in cost, speed, or power. However, if we see reliability as a continuum, then our goal is maximize the tradeoff between reliability and the other variables. In this work, we explore the tradeoff between reliability and complexity.

This paradigm shift requires two pieces - software that can tolerate less reliability and hardware that can manage correctness guarantees. Error-tolerant applications, such as the MPEG decoder, are already abundant in the digital signal processing domain where networks are inherently unreliable. We are merely pushing this error tolerance into the processor requiring that errors occur only in the network.

The goal of our hardware design is to manage the errors without adding complexity. The challenge is that many existing error-tolerant applications are designed to detect and tolerate certain error rates in raw data from the network. In the processor domain, more critical errors can occur. For example, if the stack pointer becomes incorrect, the entire program could be compromised. We find that there are three classes of data that have varying importance in the overall correctness of the program: data, stack, and pointers.

We propose the R-bit micro-architecture that allows some functional units to be unreliable. It uses simple hardware and heuristics to reduce the chances that calculations affecting the stack use these unreliable functional units. This is accomplished by associating a reliability bit with each register in the register file that indicates whether the data in that register is allowed to be unreliable. The result of any operation involving possibly unreliable data is considered unreliable.

We present preliminary data which illustrates how error rates in data, stack, and pointers affect the fidelity of our MPEG decoder. We further show that our initial R-bit design protects control data and increases quality.

Our paper is organized as follows. We present the MPEG application in Section 2. In Section 3, we describe the R-bit micro-architecture. Section 4 shows our preliminary results. We conclude with related work and directions for future work in Section 6.

## 2 MPEG

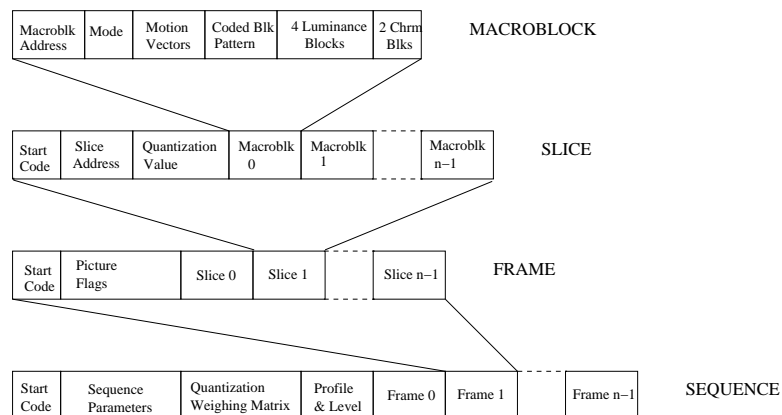
The MPEG standard, first proposed by the Moving Picture Experts Group in 1993, currently forms the most popular compression method for video and audio, and for streaming applications. In MPEG-2 digital video data is encoded as a series of code words in a compressed manner such that the average length of the code words is much smaller. The standard allows for encoding video over a wide range of resolutions.

### 2.1 Video Redundancy

A bit rate reduction system operates by removing redundant information from signal at the encoder, prior to transmission, and re-inserting it at the decoder. An encoder and decoder pair are referred to as a 'codec'. Given that all video is primarily composed of stationary picture frames, MPEG takes advantage of two distinct kinds of redundancy in video signals.

1. *Spatial and temporal redundancy*: Pixel values are not independent, but are correlated with their neighbors both within the same frame and across frames. This implies that the value of a pixel may be predicted, given the values of neighboring pixels.
2. *Psychovisual redundancy*: The human eye has a limited response to fine spatial detail [PSK04, GF99], and is less sensitive to detail near object edges or around shot-changes. Consequently, controlled impairments introduced into the decoded picture during the bit rate reduction process should not be visible to a human observer.

MPEG employs two primary techniques. The intra-frame Discrete Cosine Transform (DCT) coding and motion-compensated inter-frame prediction. A two-dimensional DCT is performed on small blocks (an array of 8x8 pixels) of each component of the picture to produce blocks of DCT co-efficients. The magnitude of each DCT coefficient indicates the contribution of a particular combination of horizontal and vertical spatial frequencies to the original picture block. Pictures called I frames are encoded without prediction, P frames may be encoded using prediction from previous pictures, while B frames may be encoded with prediction from both previous and subsequent frames.



**Figure 1. Data structures employed in an Mpeg-2 Sequence**

## 2.2 Encoding P and B frames

Data representing macroblocks of pixel values for a picture to be encoded are fed to both the subtracter and the motion estimator. The motion estimator compares each of these new macroblocks with macroblocks in a previously stored reference picture or pictures. It finds the macroblock in the reference picture that most closely matches the new macroblock. The motion estimator then calculates a motion vector which represents the horizontal and vertical displacement from the macroblock being encoded to the matching macroblock-sized area in the reference picture. Note that the motion vectors have 1/2 pixel resolution achieved by linear interpolation between adjacent pixels.

The motion estimator also reads this matching macroblock (known as a predicted macroblock) out of the reference picture memory and sends it to the subtracter which subtracts it, on a pixel by pixel basis, from the new macroblock entering the encoder. This forms an error prediction or residual signal that represents the difference between the predicted macroblock and the actual macroblock being encoded. This residual is often very small.

The residual is transformed from the spatial domain by a 2 dimensional DCT. (The two dimensional DCT consists of separable vertical and horizontal one-dimensional DCTs.) The DCT coefficients of the residual are then quantized in a process that reduces the number of bits needed to represent each coefficient. Usually many coefficients are effectively quantized to 0.

The quantized DCT coefficients are Huffman run/level coded which further reduces the average number of bits per coefficient. This is combined with motion vector data and other side information (including an indication of I, P or B picture) and sent to the decoder.

For the case of P pictures, the quantized DCT coefficients also go to an internal loop that represents the operation of the decoder (a decoder within the encoder). The residual is inverse quantized and inverse DCT transformed. The predicted macroblock read out of the reference picture memory is added back to the residual on a pixel by pixel basis and stored back into memory to serve as a reference for predicting subsequent pictures. The object is to have the data in the reference picture memory of the encoder match the data in the reference picture memory of the decoder. B pictures are not stored as reference pictures.

The encoding of I pictures follows the same procedure, however no motion estimation occurs. Here, the quantized DCT coefficients represent transformed pixel values rather than residual values as was the case for P and B pictures. As is the case for P pictures, decoded I pictures are stored as reference pictures.

The system can encode sequences of progressive or interlaced pictures. For interlaced sequences, pictures may be encoded as field pictures or as frame pictures. For progressive sequences, all pictures are frame pictures with frame DCT coding and frame prediction. Further explanation is found in Field DCT Coding and Frame DCT Coding.

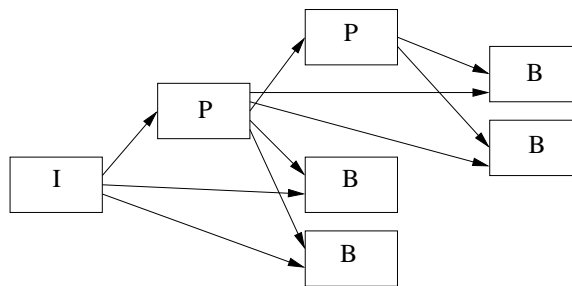


Figure 2. Dependencies between I,P and B frames

## 2.3 Decoding

The decoding process can be thought of as the reverse of the encoding process. The received encoded data is Huffman/run-level decoded. Motion vectors are parsed from the data stream and fed to the motion compensator. Quantized DCT coefficients are fed to the inverse quantizer and then to an Inverse DCT that transforms them back to the spatial domain. For P and

B pictures, motion vector data is translated to a memory address by the motion compensator to read a particular macroblock (predicted macroblock) out of a previously stored reference picture. The adder adds this prediction to the residual to form reconstructed picture data. For I pictures, there are no motion vectors and no reference picture, so the prediction is forced to zero. For I and P pictures, the adder output is fed back to be stored as a reference picture for future predictions.

In general, the loss of B and P frames can be compensated for by the decoder, while the loss of an I frame will result in very noticeable quality degradation.

### 3 Architecture

Since the streaming media protocols are designed to tolerate loss of packets over the network, we can exploit this inherent redundancy to address the issue of reliability in the micro-architecture. Basically, we allow the architecture to have components of low reliability and then use these components to compute data that is not critical.

We let some of the functional units operate at low reliability, and ensure that the data that reaches these functional units is not critical. To ensure this, we add an bit to each architectural register, termed the *R-bit*. If the R-bit of a register is high then the data in that register can only be scheduled to a high reliability functional unit. Similarly, an unset R-bit implies that the data can be sent functional units of lower reliability. In addition to the R-bits, we have a global flag, *Rzone*, which indicates when R-bits may be used for scheduling decisions. The Rzone flag is set to low the application is operating on a block of less critical data; and the R-bits help identify individual instructions within that block that could be performed by less reliable functional units. To either set or unset the Rzone flag, we require that the application provide a hint. This hint is in the form of a single instruction and, for the mpeg decoder, is detailed in the Experiments section .

Initially all R-bits are set to high, and the Rzone flag is also set to high. When the hint is received, the Rzone flag is unset and we change the R-bit values based on the following heuristic (see figure 3):

1. On load, if the target register is a general purpose, unset its R-bit.
2. On store, if R-bit is unset, reset it.
3. On integer computation, if the R-bit of both source registers is unset, the instruction can be scheduled to a less reliable functional unit. The R-bit of the destination register is also unset.

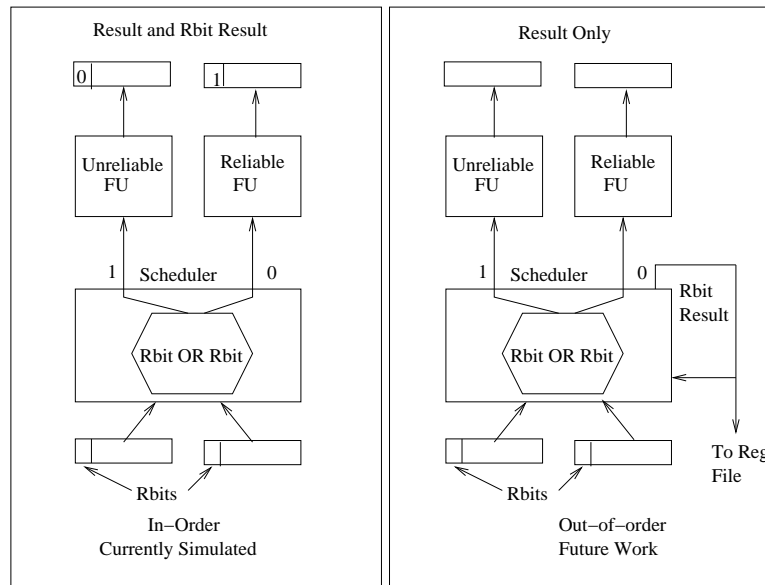


Figure 3. The R-bit micro-architecture

The simple heuristic makes sure that we do not corrupt control data (which in SimpleScalar is registers 30,31 and 29), but it does not guarantee that pointers do not become corrupted. We discuss pointer protection in future work (Section 6). In

order to verify that possibly corrupt data is not used for pointer calculations, we add a second bit, the *vulnerability bit (V-bit)*, to data that has passed through a less reliable functional unit. For the Mpeg decoder, we count how often data with the V-bit set is used as load address and measure the associated change in fidelity of decoder output.

Instruction scheduling is an issue, since R-bits need to be available at the time of scheduling. In order to facilitate scheduling, we assume a separate R-bit forwarding network and a fast path through functional units that propagates R-bits using our simple heuristics. Future work will measure the precise complexity and cost of these mechanisms.

## 4 Experimental results

### 4.1 Methodology

We conducted fault injection experiments on the mpeg2 decoder both with the R-bit scheme and without. These were performed using the SimpleScalar tool-set. Since we require a hint from the application that allows the architecture to start using the low reliability functional units, we instrumented the decoder by adding a single assembly instruction in the right places. This *hint* is placed where the decoder starts processing either a P frame or a B frame and hence sets the global reliability flag to low. A second hint is placed where the decoder is no longer processing frames that don't require high reliability.

We ran a short test clip through the decoder. To get the base value, against which all other runs would be compared, all decoded data was output to files. From this decoded data, the number of macroblocks that the decoder skipped per frame, can be determined. The next step was running the decoder in SimpleScalar and turning fault-injection on; this second set of run did not use Rbits. Finally, the third set of data made use of Rbits while running the decoder in SimpleScalar with fault-injection on. The lowest fault injection rate was approximately 1 fault every two seconds. The other rates were twice, five times and 10 times the lowest rate. With Rbits enabled, the faults were injected only into the low reliability functional unit. Without Rbits, they were distributed evenly.

The data from the second and third set was compared to the data from the unmodified decoder. If a frame in either the second or third set had greater than 10 percent difference, in number of skipped macroblocks, as compared to the unmodified decoder, the frame was considered corrupt. Our goal was to compare the number of corrupt frames when injecting faults with Rbits enabled, and without. Since the heuristic mainly protected against control errors, this difference shows how errors in data that manipulate control-flow increase the number of errors.

### 4.2 Results

We define the quality of the decoder output as the number of frames, where a corrupt frame is one that has greater than 10 percent difference in the number of skipped macroblocks (as compared to the reference output) Figure 4 shows the results that were obtained. It can be seen that enabling Rbits results in a significant reduction in the number of corrupt frames output by the decoder. In fact, as the fault injection rate increases, the savings attained by using Rbits dramatically increases.

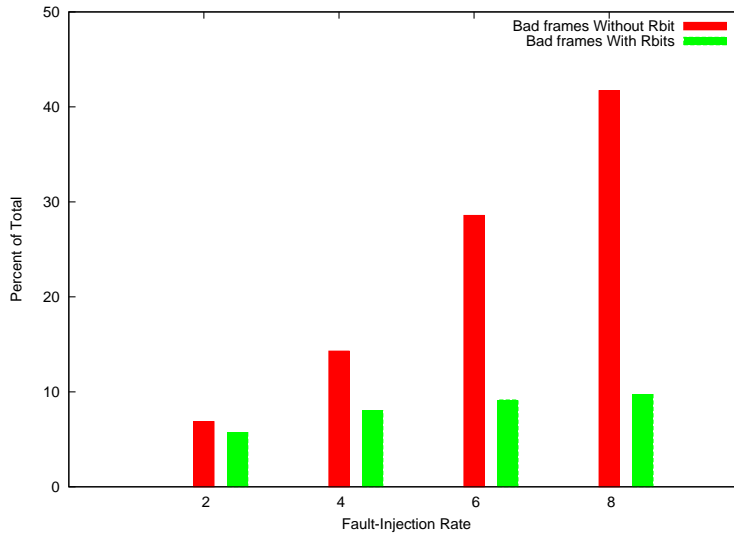
We also would like to know what effect an error in a pointer has on the overall fidelity of the picture, and how corruption in data spreads. Figure 5 takes us a step towards that information. For this data, the maximum errors have been capped at the values shown on the X-axis (10,50 etc). Using the *V-bits* we keep track of when an error is introduced into data, and how this corrupt data effects other operations. In 5 we show how increasing the number of errors inserted into the decoder, increases the amount of bad pointers and bad integer operations performed. This in-turn increases the number of skipped macroblocks.

The next step is to allow only errors in pointers or errors in data to propagate, and observe how that changes the quality of the decoder output. This would allow us to say that for a given MPEG stream, which is worse: an error in data or in pointer calculations?

## 5 Related Work

This R-bit architecture builds upon work on reliability in architecture and error-tolerance in digital signal processing applications.

The RAMP architecture[SABR04] dynamically adapts the reliability of the processor depending on the device properties through time. As different components fail, the architecture adjusts parameters to maintain the overall level of reliability desired. The most closely related work is from Weaver et al[WEMR04], who developed techniques to reduce the probability



**Figure 4. Number of corrupt frames with and without Rbits. Since the heuristic uses R-bits to protect control data, the graph shows how protecting control data leads to less degradation of the output.**

or errors and reducing the impact of errors. They analyzed which structures were less reliable and reduced the time spent in those structures. We, on the other hand, analyze which operations will be least affected by unreliability and send them to unreliable structures.

The theoretical foundations of this work are based in approximate signal processing [SJ97] and flexible computation [E.J87], which deal with the systematic tradeoff between accuracy of results and the utilization of resources. Video compression is a good application of approximate signal processing since human perception can inherently tolerate some inaccuracy. This can be exploited to create a hierarchy of importance in terms of the underlying data and its impact on distortion. This can now be used to make dynamic resource trade-offs, which in the case of video, is typically the bit rate. The novelty of our approach is the architectural support for approximate signal processing.

The Razor work by Austin et al [ABMF04] is the most relevant and complementary to our work. Razor uses “shadow latches” to check pipeline paths in the presence of voltage overscaling. Our work exploits application characteristics and could use Razor for our reliable operations while potentially operating beyond the error regime of Razor for our unreliable operations.

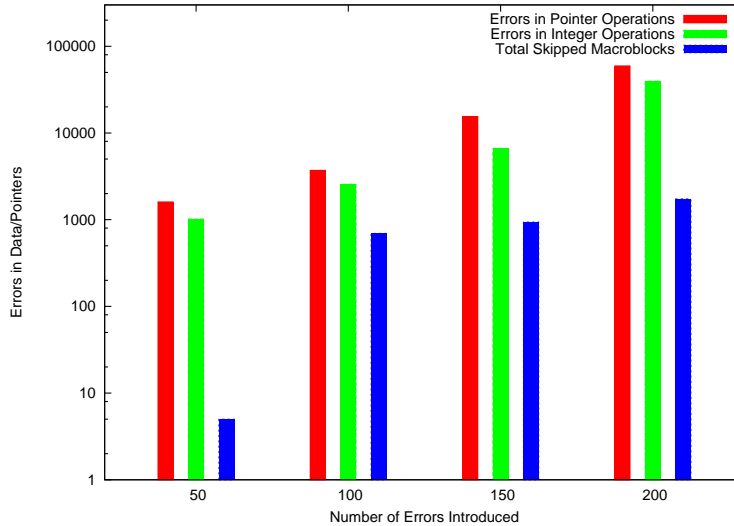
## 6 Future Work

The goal of this preliminary study has been to measure the potential gains of reliability management for error-tolerant applications such as MPEG. Our results suggest several major areas of future work.

Although our simple heuristics have demonstrated promising gains, there is substantial room for improvement. We have only protected a subset of control data in our application. Furthermore, we have not protected pointer data. Our next step is hand-code applications to add such protection through R-bits stored in memory. One R-bits are stored in memory, making the R-bits available for instruction scheduling becomes a challenge. R-bit caching and/or prediction will be necessary to allow instructions to be scheduled in a timely manner.

In addition, in this study, we simulated our heuristic on an in-order processor. We will adapt the design to an out-of-order processor. The challenge is that scheduling decisions must be made before the dependent instructions have completed, so the R-bits will need a separate, fast data-forwarding path. This is feasible since the R-bit calculation is only a single OR operation which is completed in the scheduling step of the instruction. If the R-bits are stored in memory, however, we do not have this advanced calculation. We need either prediction or a faster R-bit region of memory for loaded values so that subsequent instructions need not wait for the load to complete in order to be scheduled.

Once the potential of hand-coded schemes is measured, we plan to investigate compiler analysis to generate R-bits.



**Figure 5. Classification errors.** The X-axis shows the number of errors added. We track how each error propagates and causes pointer errors and errors in data. The graph also shows the total number of macroblocks skipped by the decoder for each error-injection campaign.

Since such analysis is a superset of pointer analysis and necessarily conservative, it will be important to augment such static methods with dynamic mechanisms. Our V-bit gives us a starting point for such mechanisms, allowing us to detect vulnerabilities in control, pointer, and remaining data. Further mechanisms might include iterative adaptation of R-bits – unsetting conservatively set R-bits over successive frames. Such adaptation might require keeping extra data dependence information in hardware as the code executes.

As we improve our reliability management methods, our architecture will be able to tolerate higher error rates. These higher error rates will allow us to move from the regime of soft errors caused by noise to the regime of soft errors due to voltage overscaling. Voltage overscaling puts us in a new landscape of tradeoffs between reliability and power.

Finally, while we have focused on the reliability of functional units in our micro-architecture, our architecture naturally (in fact, unavoidably) extends to the memory system. Tradeoffs in memory system design involving power, area and reliability will provide fertile ground for innovation and optimization.

## 7 Conclusion

Our preliminary results indicate that the explicit management of reliability at the microarchitectural level can be a powerful tool for efficient execution of error-tolerant applications. The potential of this approach reveals a large space of research issues. Integrating instruction scheduling with reliability management will be challenging, but we expect to leverage traditional ideas in caching and prediction. Determining reliability requirements will also be challenging and require substantial work in both static and dynamic program analysis. We expect the interaction between static analysis and dynamic hardware mechanisms to be an exciting area of investigation. Overall, we hope that this work will generate interesting discussion and help stimulate research in a promising area.

## References

- [ABMF04] Todd M. Austin, David Blaauw, Trevor N. Mudge, and Krisztián Flautner. Making typical silicon matter with razor. *IEEE Computer*, 37(3):57–65, 2004.
- [CM01] Philip A. Chou and Zhouong Miao. Rate-distortion optimized streaming of packetized media. *Microsoft Technical Report*, February 2001.

- [E.J87] E.J.Horvitz. Reasoning about beliefs and actions under computational resource constraints. *Third Workshop on Uncertainty in Artificial Intelligence*, pages 429–439, 1987.
- [GF99] Bernd Girod and Niko Farber. Wireless video. *Compressed Video Over Networks*, Chapter 12, 1999.
- [PSK04] Martin Reisslein Patrick Seeling and Beshan Kulapala. Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial. *IEEE Communications Surveys*, 6(3):58–78, Third Quarter 2004.
- [SABR04] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. The case for lifetime reliability-aware microprocessors. *Proceedings of the 31st Annual International Symposium on Computer Architecture*, December 2004.
- [SJ97] A.P.Chandrakasan S.H.Nawad, A.V.Oppenheim and J.Wingrad. Approximate signal processing. *Journal of VLSI Signal Processing*, 15:177–200, 1997.
- [WEMR04] Christopher T. Weaver, Joel Emer, Shubendu S. Mukherjee, and Steven K. Reinhardt. Reducing the soft-error rate of a high-performance microprocessor. *IEEE Micro*, 24(6):30–37, Nov 2004.