Design and Analysis of Low Power Image Filters Toward Defect-Resilient Embedded Memories for Multimedia SoCs

Kang Yi[1], Kyeong Hoon Jung[2], Shih-Yang Cheng[3], Young-Hwan Park[3], Fadi Kurdahi[3], and Ahmed Eltawil[3]

[1] School of Computer Sceince and Electronic Engineering, Handong Global University, Pohang, Korea
yk@handong.edu
[2] Department of Electrical Engineering, Kookmin University, Seoul, Korea
khjung@kookmin.ac.kr
[3] Department of EECS, University of California, Irvine, CA 92697-265
{shihyanc, younghwp, kurdahi, aeltawil}@uci.edu

Abstract. In the foreseeable future, System-on-Chip design will suffer from the problem of low yield especially in embedded memories. This can be a critical problem in a multimedia application like H.264 since it needs a huge amount of embedded memory. Existing approaches to solve this problem are not feasible given the higher memory defect density rates in technologies below 90 nm. In this paper, we present a new defect-resilience technique which employs the directional image filter in order to recover data from corrupted embedded memory. According to the analysis based on simulation the proposed filter can greatly improve the visual quality of the defected H.264 video streams with errors in data memory reaching up to 1.0% memory BER (Bit Error Rate) with lower power consumption relative to conventional median filter. Therefore, the proposed method can be a good solution to overcome the problem of low yield in multimedia SoC memory without suffering from additional redundant memory overhead.

Keywords: Low power image filter design, Embedded memory, Memory yield enhancement, Memory-error resilient design, BIST, BISR, H.264 codec.

1 Introduction

The ever-shrinking design geometries are allowing system designers to integrate larger memories on-chip. Integrating memories and processing core (random logic) into a single chip has the following benefits: (1) higher performance, (2) reduced power consumption, (3) lower parts cost, (4) less inter-chip communication complexity, and (5) smaller number of packages on a system board. Embedded memories are expected to account for most of the silicon die area in the near future SoC (System-on-a-Chip) design as the system-on-chips are moving from logic dominant to memory dominant to meet the application requirements [9]. According to the 2001 International Technology Roadmap for Semiconductor the embedded memories are going to occupy from 54% to 94% of silicon real estate by year 2014 [1,2]. In particular, multimedia application such as H.264 video decoders require significant amount of memory to store intermediate frame data. Thus, the rapid growth in multimedia content and the corresponding increase in demand for high performance and low-power terminals has exerted increased pressure to integrate these large memories into a System-on-Chip.

However, the lower yield of embedded memory is becoming a barrier to the widespread of very deep submicron technology adoption in SoC. Because memory design uses the most aggressive design rules, memory circuits are more susceptible to the manufacturing errors than random logic circuits. The situation is likely to get worse with the random dopant fluctuation (RDF) problem under 100 nm technology [7]. Therefore, the increasing embedded memory size on a chip may result in the lower SoC yield and higher manufacturing chip cost. Thus, the embedded memory yield is becoming a key issue for the overall SoC yield improvement.

In order to combat this memory yield problem memory repair is typically performed after manufacturing using redundancy. Spare columns and rows are used to replace the defective parts of the memory [3, 4]. These methods show a reasonable yield improvement for the current technology by repairing defective memories with up to 0.003% error rate. But, these existing techniques do not work properly with the very deep-submicron technologies because these methods require too much overheads in terms of the area for the higher defect density of future nano device technology. According to [4] the redundancies to repair memory with even 0.1% BER (Bit Error Rate) require at least 67% of area overhead.

In this paper, we present a new approach that handles embedded memories with higher defect densities than current technologies. Our approach essentially pushes the task of repair from the circuit level up to the application level. To illustrate this approach we focus in this paper on one representative application which is the H.264 video decoder. H.264 is a promising multimedia application which is also memory-hungry. To achieve 100% error free huge embedded memory is an almost impractical assumption especially for advanced processes. Instead of fixing the errors at the circuit (or bit-exact) level, we compensate for them at the application level. There exist a multitude of techniques to do so. One of these techniques is to perform simple spatial filtering on the individual video frames. In this paper, we focus on this approach and present a family of spatial filters that can recover the defect pixel values with simple and yet effective image processing algorithms as depicted in Fig 1. We also analyze the performance and power consumption of these filters in software implementation. With these filters, we can guarantee 100% application-level recovery from errors in embedded memories with up to 0.1% BER as well as current technology memory with BER of 0.001%.

The next chapter explains the background knowledge about defect memory problems in H.264 application and summarizes the related previous works. In Chapter 3, we define the problem for filter design and in Chapter 4 we design new filters for simple and enhanced image processing to recover defective pixel values. In Chapter 5, we show the experimental results and discuss the implication. Finally, in Chapter 6, we conclude our work with future research directions.

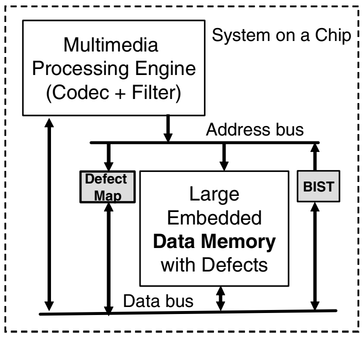

System on a Chip
Multimedia Processing Engine (Codec + Filter)
Address bus
Defect Map
Large Embedded Data Memory with Defects
BIST
Data bus

Fig. 1. Our System Architecture with Memory Error-Resilient H.264 decoder : our system has large memory which is defect location. The Defect map and BIST will be used for the problem.

2 Background

In [7], it was shown that one does not need to discard the manufactured chips with memory defects only if we implement the error resilience features at the design time with application specific data redundancies. In [7], we found that it is possible to compensate for the loss due to imperfect data memories using application-level error concealment techniques. By nature, multimedia data has redundancies and such a redundancy may be used to recover data properly. Therefore, H.264 is an ideal example to demonstrate the embedded memory yield enhancement by our approach. Assume an HDTV decoder with 1920 x 1080 pixels image size per frame and the decoded frame image is stored in a memory called Decoded Picture Buffer (DPB) which can be used later for inter-prediction task. Since 4:2:0 sampling mode is assumed and each components has 8 bits per pixel the DPB size is 24Mbits per frame. Since 2 – 5 frames are usually required as a reference frame, at least 48 Mbits of data memory is required. Currently, this DPB memory exists as an external memory chip connected to the main processor chip. As device geometries continue to shrink, the forthcoming SoC design is going to integrate this external memory into a chip. As mentioned before, such huge memories with future VDSM technology will suffer from higher defect rate resulting in lower SoC yield problem.

Fig 2 shows overall structure of our H.264 decoder with memory defect resilient feature. A defect map memory stores the defect pixel location and is used for error concealment schemes with image filter. The image filter is applied only to the defect pixels to get the correct pixel value for each defect pixel reading operation. We may assume that the defect map is constructed at the manufacturing test time. Alternatively, we may assume that it is reconstructed at each power up time by scanning memory reflecting the changing memory status.

Fig 3 shows the overhead of using defect map compared with the existing redundancy approach in [4] and [5] for 100% embedded SRAM yield. Note that the Y-axis in Fig 2 is a log scale. The redundancy scheme in [4] can at most repair only 0.003% defects thus the data used for that level of defects is based on approach in [4].

For higher defect rates we used the data from [5]. The defect map can be implemented in one of two ways: (1) using CAM (Content-Addressable Memory) and (2) using a tag bit for every data word (TAG). In comparing redundancy-based approaches with the proposed defect map-based approach, and assuming 0.1 % memory BER, our approach (defect map + filter) of [7] requires only 2.29% memory area overhead with the CAM-based defect map. The data in Fig 3 indicates clearly that the defect map with filter approach requires much less area overhead for 100% memory yield.

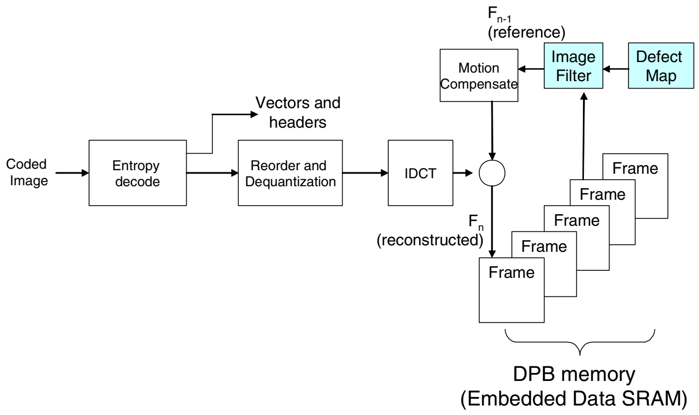


Fig. 2. H.264 decoder with error resilience feature by defect map and filtering

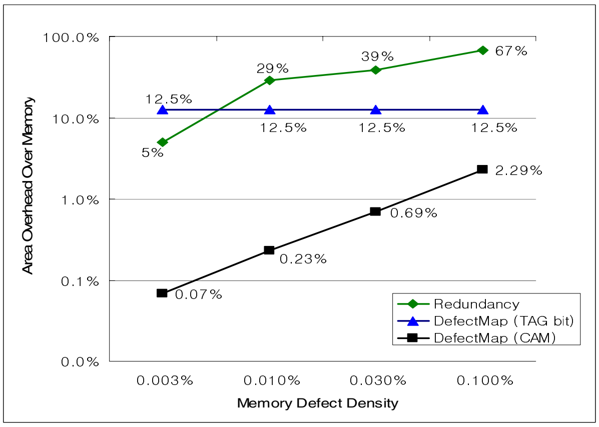


Fig. 3. Overhead Comparison between defect resilience schemes

3 Problem Definition (Filter Design Constraints)

We are going to design filters under the following assumptions and design requirements.

(1) The filters are for the H.264 decoder side.
(2) The errors come not from transmission but from the *storage defect* for the image reference frame data.
(3) The filters work on the Y,U and V domain only.
(4) The defects are distributed in a *randomly uniform fashion.*
(5) The defect rates to cover are in the range *from 0.001% to 1.0%.*
(6) Every defect location is known by defect map.
(7) The filter will only target the defect pixels whose locations are in the defect map.
(8) Our memory defect model is based on the stuck-at fault model in bit level.
(9) The defective pixel map is provided for each Y, U, and V component as separate and independent memories and the defect map memory is assumed to be error free, which is reasonable assumption for the defect map is quite small.
(10) The filter should not be too complicated. The filter overhead to overall system should be minimized.

4 Our Filter Design

The best image filter for error concealment shown in [7] is the median filter which sorts the neighboring eight pixels and finds the median value [10]. This filter shows relatively high PSNRs and good visual quality. However, the problem with the median filter is that it blurs the image when used repeatedly, which makes it inappropriate to use for the higher error rate cases. The other problem is that the median filter requires complex processing as pixel values have to be sorted and therefore consumes a lot of power. One difference between our problem and the traditional image error concealment problem is that the error rates of memory for our problem domain are typically smaller than those assumed in the traditional image processing area. Thus, we don't need to use the existing complex image processing filters to recover the image. In this section, we develop several special image filters which are adjusted for our unique situation.

4.1 Basic Filter Designs

We devised a set of basic simple filters based on the idea that image pixel value can be recovered by finding image direction [11]. Basically, our filters consider only the 3 x 3 pixels around each of the defective pixel to find the image direction as shown in Fig 4 in order to minimize the filter complexity.

(1) **Two pixel mean value filter (MEAN2 filter):** This filter finds the defect pixel value as the mean value of two pixels horizontally neighboring the defective pixel. This filter is the lowest cost filter but works well at very low error rates. For example, The MEAN2 filter for P8 is computed as follows :

Pixel_value (P8) $\leftarrow$ (pixel_value(P0) + pixel_value (P4))/2

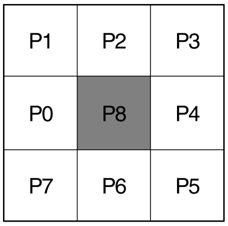


Fig. 4. A 3 x 3 array of pixels round defect pixel P8 : Every filter of our paper assumes this basic 3x3 pixel array to find the correct value of the defect pixel P8

- **Directional filter (DIR filter):** This is an advanced version of blind mean of two pixel filter (MEAN2). Instead of taking the mean of horizontally adjacent two pixels, this filter tries to find the better direction among four possible directions (horizontal, vertical, and two diagonals) of image. Basically, our new filter consists of two phases: the image direction identification phase and the computing the pixel value phase. In the phase 1 the image direction is detected by the absolute differences of two pixel pair for each direction. The direction related to the pixel pair with the least difference is considered the image direction. Fig 5 explains the four candidate directions. DIR filter shows very good performance (PSNR and visual) only if the surrounding pixels are all sound (non-defect pixel or non-boundary cases). According to our analysis of defect images with 1.0% memory BER, DIR filter can compare four all directions properly only for about 53% cases and it fails to decide the direction at all for about 0.02%.

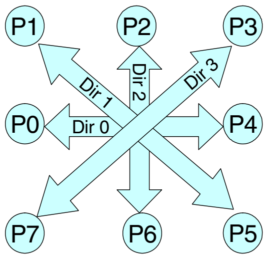


Fig. 5. Directional filter with four candidate directions : DIR filter decide the image direction among the four candidate directions by the pixel value difference computation of the pairs

- **Extended Directional Filter (EXT filter):** In some cases DIR filter fails finding image direction because there are too many corrupted pixels around the defect pixel. In this case, Extended Directional (EXT) filter may be an alternative method. The EXT filter uses more pixels to detect image direction in phase 2 of DIR filter by searching the 5x5 surrounding pixels as shown in Fig 6. According to our observation, EXT filter evaluates all four directions at about 75% cases. It shows relatively higher performance but is more complicated than DIR filter.

Table 1 summarizes the filter performance results with “foreman” video image (encoded with QP=28) with 1.0% BER in memory. In the table 1, “Median_filter1” column shows the median pixel value excluding the neighboring defect pixels while "Median_filter2" column shows median filter including every eight neighboring pixels. The results in Table 2 show that DIR filter and EXT filter outperform the median filters for luma but are not as good as median filters for chroma.

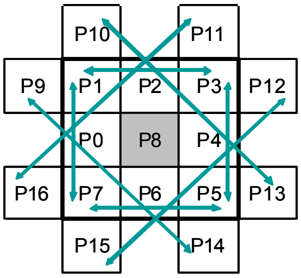


Fig. 6. Pixels used for EXT filter : 5x5 pixels are used for more reliable image direction detection for Y component in phase 2 of EXT filter

Table 1. PSNR of filters for foreman video image with 1.0% error

Components	Median filter1	Median filter2	DIR filter	EXT filter
Y	28.73	28.54	29.95	32.06
U	38.55	38.37	33.53	34.81
V	39.63	39.85	36.16	36.56

4.2 Enhanced Chroma Filter Designs

In Table 1, it is clear that we need to develop better filters for the chroma image components (U and V). The required filters must match the quality but achieve lower cost (power) than median filter. We designed two alternative chroma filters. For these new filters we have an assumption that every luma values are available because these filters use luma component to evaluate image direction. To support this assumption we should use the luma filters twice per pixel to get the correct luma pixel values.

- **Four Directional UV filter (UV4 filter):** The UV4 filter is designed for U and V components only. When we have many consecutive defect pixels there are higher probabilities that every four directions are discarded from the candidates resulting in unfiltered pixels. As a result, with an increasing defect density, we may have more unfiltered condition with DIR filter. Because the distortion of chroma is known to be less sensitive to human eyes, the sampling rate of luma to chroma is already assumes 4:1. Because of this sampling rate if we have peculiar value for a chroma of a pixel, the image may be degraded with big ugly spot. Thus, we need a new filter that does not miss any pixel even under the higher defect density. Our idea is to use luma information in the corresponding positions with the chroma component that we want to filter without failure. Fig 7 shows the required luma pixels for defective chroma value U8. The image direction detection idea is similar to that of the DIR filter.

If the estimated direction is horizontal in Y component, we take the mean value of U4 and U0 to find the filtered U8 value in U component. Assume that the estimated direction from UV4 filter is U0 and U4. But, if U0 is also a defective pixel our strategy is to apply the same filter for U0 to get the estimate of its correct value and use that value of U0 to get an estimate of the U8 value. Under extreme conditions this may lead to too many levels of recursion which results in additional power consumption. In order to prevent that, we restrict the number of recurrence depth. Our simulation results tell the recursion depth one is enough for the quality of filtered color value.

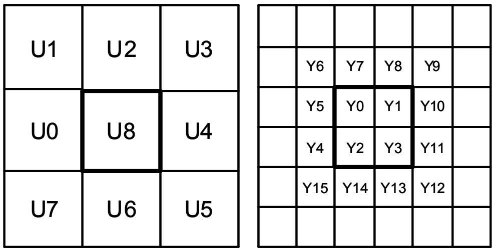


Fig. 7. Directional UV filter concept with lumma : one pixel in Chroma (U8) corresponds four pixels in Lumma (Y0~Y3). We use Y0 ~ Y15 to find the image direction of U8.

- **Eight Directional UV filter (UV8 filter):** The Eight Directional UV filter (UV8) is a modified version of UV4 filter. Instead of taking the mean value of two pixels, it just copies the neighboring pixel value. The UV8 filter is based on the idea that the neighboring pixel values in chroma are not so different and the color difference is less sensitive to the human eyes. The computation complexity of UV8 and quality is usually comparable to UV4.

5 Experimental Results and Filter Analysis

We estimate the performance and power consumption of filters to decide which filter combination is the best for our problem. We assume that our target H.264 system-on-a-chip consists of DSP core plus embedded memories as shown in Fig 8. Therefore, our filter will be implemented in forms of a program code added to the H.264 decoder

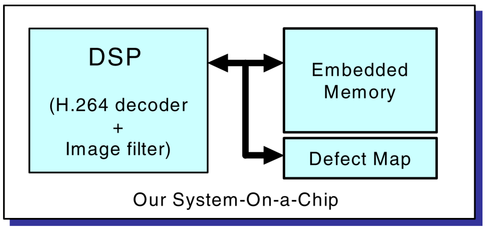


Fig. 8. Our System Configuration with DSP and embedded memory

program code launched on a DSP processor. The DSP processor used in our experiment is TMS320C5510 with 1.6 V supply voltage.

We used Code Composer Studio from TI [12] to compute the clock cycles and utilization of each filter. We used the power computing provided by TI web site [13] to get the estimated DSP power by entering frequency and utilization. Our final power consumption data includes memory power of defect map (CAM) and DPB (SRAM) as well as the core power consumption as shown in the following equation.

$$Total\_Power = Core\ Power + Defect\ Map\ Power + DPB\ Power$$
$$Total\_Cycles = Cycles/Pixel \times P_{pixel} \times Image\_Size/Frame \times Frame\_Num/Sec$$
$$P_{pixel} = 1\text{-}(1\text{-}\ P_{bit}\ )^{Pixel\_Depth}$$

In the above equations, P_{pixel} is a pixel error rate, P_{bit} is a bit error rate, Pixel_Depth is the number of bits for each pixel component (8 for our case), and Cycles/Pixel means the number of instruction cycles for each filter execution on a DSP.

We applied all of our filter combination in one of two manners : *priority and weighted selection*. The priority selection with notation *filterA>filterB* means that primary we use the filterA and then we use the filterB only if filterA fails finding the proper value (because of too much defect pixels or boundary pixel case). And, the notation *filterA+filterB* means applying two filter at the same time equally and selecting one of the results according to some criteria. In the following pseudo code, we summarize the priority and weighted combination. A priority combination approach consumes less power compared with weighted combination.

```
Function FilterA>FilterB (pic : pixel_position ) {
   New_value ← Compute Filter A (pic)
   IF (Filter A fails to find the filtered value)
        New_value ← Compute Filter B(pic)
   Pixel_value(pic) ← New_value
}

Function FilterA+FilterB (pic : pixel_position ) {
   (New_value1, quality_metric1) ← Compute Filter A (pic)
   (New_value2, quality_metric2) ← Compute Filter B(pic)
   IF (quality_metric1 is better than quality_metric2)
        Pixel_value(pic) ← New_value1;
   ELSE
        Pixel_value(pic) ← New_value2
}
```

Our assumptions on the input streams are that: (1) videos are encoded with quantization parameter QP= 28 for all the I, B, and P frames and (2) the frames are transferred in a IPBPBPBP… sequences, and (3) one I frame comes every 60 frames. We use a sample H.264 video sequence named “foreman”, at 30fps and 144 x 176 (QCIF) frame image size with 4:2:0 sampling rate. We assume our target system has five reference frames in all cases. Every PSNR value is computed relative to the original sample YUV format video image before encoded in H.264. The PSNR value is computed by the following equation.

$$MSE = \frac{\sum [f(i,j) - F(i,j)]^2}{N^2}$$

$$PSNR = 20 \log_{10} \left(\frac{255}{\sqrt{MSE}} \right)$$

Where *f(i,j)* and *F(i,j)* are the pixels at location *(i,j)* of the output and reference images, respectively.

First, we evaluate the chroma filters. We apply different filter combinations to our 1.0% error image and we measure the performance (the quality of image measured by PSNRs) and the cost (consumed power). The candidate filters for chroma are DIR, EXT, UV4, UV8, and their combinations. We also compare our filter results with those of the median filter used in [7]. Because the luma filter quality influences the result of UV4 and UV8 filter, we set luma component to be error-free for UV filter performance experiments. Considering both the power consumption and quality, best choice seems to be (DIR>UV8) and (DIR>Median) from experiments.

Secondly, The chroma (Y) filter candidates are tested. The candidate filters for luma are DIR, EXT, MED (MEDIAN filters), and their combinations. In order to decide the visual quality of only luma, we set chroma component error free. From the experiments we find that DIR, DIR>EXT and DIR>MED are good choices in terms of power and quality.

Finally, we put together the best filter choices for luma and chroma and try to find the best filter combinations. From the experiments, we find DIR>EXT for luma and DIR>MED for chroma is the best choice in terms of both power and quality of image. Table 2 shows the experimental result of filters with PSNR values and power consumption data including defect map and DPB power as well as core power consumption at BER 1.0%. This filter combination saves about 64% of filter power compared with the median filter as shown in Table 2.

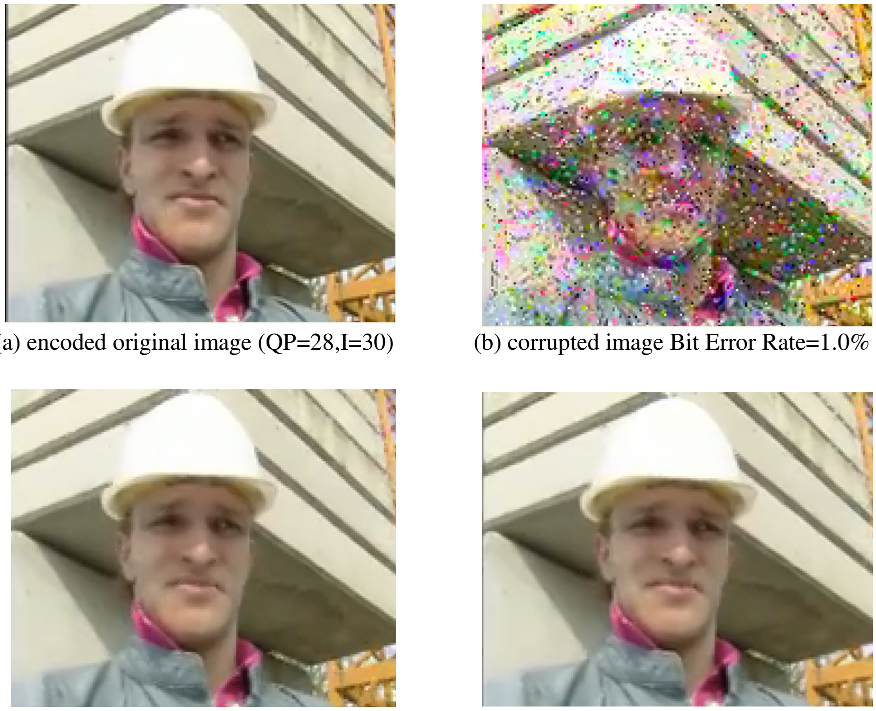
Fig 9 shows the video image of 29th frame of foreman with 1.0% memory BER and filtered image with median and our best. Note that median filter (c) and our filter (d) show no significant difference from the originally encoded image (a) without error.

Table 2 and Fig 9 show that our best filter combination achieves compatible visual quality with MED filter application while consuming less power.

Fig 10 and Fig 11 show the PSNR values of Y, U component respectively for each different filter at different defect densities. The result of V component is very similar to that of U component. We can have a range of filter choice according to defect density. Some filters show same result with less complexity and less overhead at different defect density. With mean2 filter we can achieve the same quality as median or DIR filters as shown Fig 10 and Fig 11 to save power consumption. If our goal is to select the highest quality filter with lower power consumption, at the BER range below 0.1% then we will choose mean2 filter. At the BER below 0.001% we do not need to apply any filtering. Note that 1.0% BER is at least 1000 times larger error rate

Table 2. Filter Test Result (Luma Error=1.0%, Chroma Error=1.0%)

Y filter	UV filter	Y	U	V	Visual Qulaity	Power Consumption (mW)
DIR	DIR>UV8	29.95	38.72	39.86	Poor	18.50
DIR>EXT	DIR>UV8	31.72	38.62	39.85	Poor	18.61
DIR>MED	DR>UV8	29.86	38.72	39.86	Poor	18.55
DIR	DIR>MED	29.95	38.68	39.95	Acceptable	9.80
DIR>EXT	DIR>MED	31.72	38.68	39.95	Acceptable	**9.91**
DIR>MED	DIR>MED	29.86	38.67	39.93	Acceptable	9.80
MED	MED	28.54	38.37	39.85	Acceptable	27.53


(a) encoded original image (QP=28,I=30) (b) corrupted image Bit Error Rate=1.0%

(c) median filter result from the corrupted (d) our low power filter result from corrupted

Fig. 9. Comparison with Corrupted and Recovered Foreman Video Capture : encoded image without error (a) and filtered image (c) and (d) from corrupted image show almost same visual quality. Our best filter result and median filter results shows almost as visual quality.

than the current technologies. So, currently most cases will choose mean2 filter or just leave the defect pixel as it is. But, as the technology advances the DIR>EXT filter is more likely to be chosen.

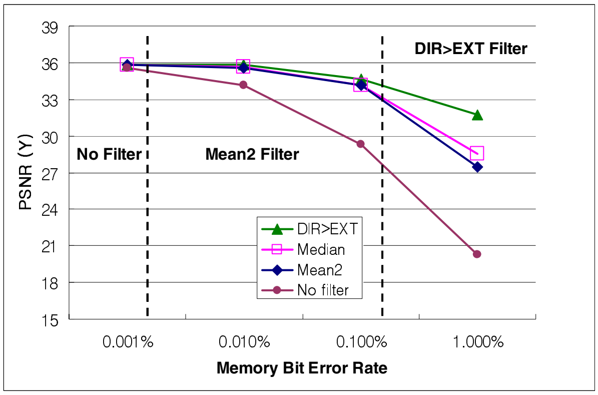


Fig. 10. Luma (Y) Filter Choice at Various Defect Densities : DIR>EXT filter is adequate for high BER and mean2 filter for low error rate. For BER< 0.001% we don't need any filter.

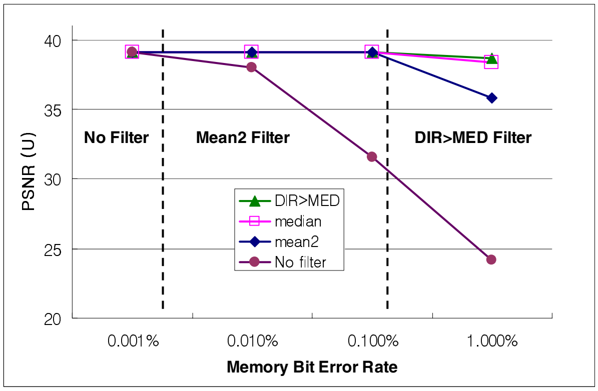


Fig. 11. Chroma (U) Filter Choice at Various Defect Densities : DIR>MED filter is adequate for high BER and mean2 filter for low error rate. For BER< 0.001% we don't need any filter.

Summarizing from the above observation, we can conclude the filter choice as follows:

(1) At BER $\leq$ 0.001%, any filter is not required for all cases.
(2) At BER in the range of 0.001% through 0.1%, mean2 filter is the best choice for all cases.
(3) At BER $\geq$ 0.1%, DIR>EXT is the best filter for luma and DIR>MED is the best filter for chroma.

Fig 12 shows the core power + DPB memory power consumption from our filter selection and median filter for a range of defect memory density. We excluded the defect map power from Fig 12 to highlight the power saving effect by the proper filter selection. In Fig 12, “Median” means power consumption by median filter and “Our Best” means the power consumed by our filters choice above. At higher defect densities, savings of 3.2 x to 12 x in power are observed.

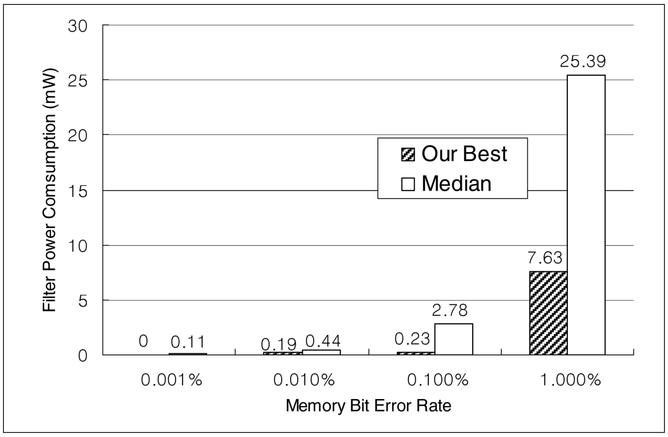


Fig. 12. Power Consumption Data by our Best Filters and Median Filter : Our filter shows about 12 times energy saving compared with median filter at 1.0% BER.

6 Conclusion

In this paper, we addressed the problem of high memory defect density in the near future nano technology era. It is a well known problem that the increasing embedded memory defect density is one of the hardest problems in memory-hungry SoC design. Our approach is based on the idea that we can recover data errors from defective memories at the application level. This is done by making use of the characteristics in the multimedia application itself. For multimedia filtering can be employed to perform such application-specific error recovery. Our newly designed directional filters coupled with defect map hardware recover the image in defective memory while consuming less power and area than conventional approaches. Our new filters show about 64 % power saving relative to conventional median filters. Our simulation results show that the new scheme achieves 3.3 to 12 times power reduction at higher defect density in memory. For the future, we are working to find more efficient filters exploiting the temporal features of H.264 as well as spatial features.

References

1. Shoukourian, S., Vardanian, V., Zorian, Y., “SoC yield optimization via an embedded-memory test and repair infrastructure”, Design & Test of Computers, IEEE Volume 21, Issue 3, May-June, Page(s):200 – 207, 2004.
2. Y. Zorian, S. Shoukourian. “Embedded-Memory Test and Repair: Infrastructure IP for SoC Yield,” IEEE Design and Test of Computers, vol. 20, no. 3, pp. 58-66, May/June, 2003.
3. T. Gupta, A.H. Jayatissa, "Recent advances in nanotechnology: key issues & potential problem areas," i*n Proceedings of IEEE Conference on Nanotechnology*, Vol. 2, pp. :469 - 472, 2003.

4. J. F. Li , R.-F. Huang, and C.-W. Wu, "A Built-In Self-Repair Design for RAMs with 2-D Redundancy", Proceedings of ITC International Test Conference, pp. 393 – 402, 2003.
5. L. Anghel, N. Achouri, M. Nicolaidis. "Evaluation of Memory Built-in Self Repair Techniques for High Defect Density Technologies," Proc., pp. 315-320, 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04), 2004.
6. S. Prihar et. Al. " A High Density 0.10um CMOS Technology Using Low-L Dielectric and Copper Interconnect." Proc. of IEDM, 2002.
7. F. J. Kurdahi, A. M. Eltawil, Y.-H. Park, R. N. Kanj, S. R. Nassif, "System-Level SRAM Yield Enhancement", Proceedings of the 7th International Symposium on Quality Electronic Design, (ISQED 2006), pp. 179 – 184, 2006.
8. K. Pagiamtzis, A. Sheikholeslami, "Contents-Addressable Memory (CAM) Circuits and Architectures : A Tutorial and Survey", IEEE journal of Solid-State Circuits, vol 41, No. 3, March 2006.
9. S. Hamdioui, G. Gaydadjiev, A. J. Goor, "The State-of-art and Future Trends in Testing Embedded Memories", Records of the 2004 International Workshop on Memory Technology, Design and Testing (MTDT'04), pp. 54 – 59, 2004.
10. Digital Image Processing, Gonzalez & Woods, Prentice Hall, 2002
11. Kyeong-Hoon Jung, Choong Woong Lee, "Projection-based error resilience technique for digital HDTV," Proc. HDTV Workshop, pp.8.A.2.1-8.A.2.9, Torino, Italy, Oct. 1994.
12. http://focus.ti.com/dsp/docs/dspsupport.tsp?sectionId=3
13. http://www-s.ti.com/sc/psheets/spra972/spra972.zip