

Fault Tolerant Design for Low Power Hierarchical Search Motion Estimation Algorithms

Charvi Dhoot^{‡¶}, Vincent J. Mooney^{§#¶¶}, Shubhajit Roy Chowdhury[‡] and Lap Pui Chau^{§¶¶}

[‡]International Institute of Information Technology, Hyderabad, India

[§]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

[#]School of Computer Engineering, Nanyang Technological University, Singapore

[&]School of Electrical and Computer Engineering, Georgia Institute of Technology, Georgia, USA

[¶]Institute of Sustainable and Applied Infodynamics, Nanyang Technological University, Singapore
charvi.dhoot@research.iit.ac.in, mooney@ece.gatech.edu, src.vlsi@iit.ac.in, elpchau@ntu.edu.sg

Abstract—Highly scaled CMOS devices are predicted to show probabilistic behavior due to process variations or the presence of noise sources such as thermal noise. Past research dealing with characterizing CMOS devices with probabilistic behavior has shown that computing via these devices, termed *probabilistic computing*, can help realize highly efficient circuits in terms of energy consumption. In this paper, we explore low power motion estimation, specifically low power hierarchical search algorithms for motion estimation, in the context of probabilistic computing. With the fault tolerant algorithm design (MC-TSS) proposed in this paper, we show that energy savings that can be realized with probabilistic computing increase to 70% versus 57% with the conventional algorithm (TSS), with minor impact on the quality of motion estimation. Furthermore, a 1.8 dB improvement in PSNR under the same energy savings of 70% for both algorithms is shown establishing the superior resilience of the proposed algorithm to probabilistic computing over the conventional algorithm.

Keywords—PCMOS architecture, probabilistic computing, fault-tolerant design, low power design, motion estimation

1. Introduction

In 2001, Kish predicted that computation through future CMOS devices will likely cease to be always correct [1]. His proposition was that as the CMOS device sizes are made smaller, scaling down the operating voltage along with them, their noise immunity to thermal noise particularly will decrease enough to induce false bit flipping. Korkmaz et al. developed PCMOS (Probabilistic CMOS) models for these CMOS devices whose output is *probabilistically* correct by coupling a noise source with Gaussian distribution at the output of gates realized through these devices [2,3]. Computing with these gates was then termed as *probabilistic computing*. Experiments with PCMOS models showed that as the operating voltage was scaled down, the errors increased but the energy savings grew exponentially with these errors, demonstrating that probabilistic computing could be a means to realize energy efficient computing in applications that can tolerate errors [2-4].

Signal processing applications involving image or speech processing lend themselves as an appropriate choice as they have a certain error tolerance limit which comes in due to the fact that human perception can tolerate a certain amount of errors in signals such as images and voice. This paves the way for design of error tolerant algorithms for such applications which can capitalize on the energy savings that voltage scaling can provide while reducing the impact of errors through these devices. In our previous work [5], we proposed a low power design for Full Search Block Matching Algorithm (FSBMA) used in motion estimation under probabilistic computing. In this paper, we propose a more robust algorithm for Hierarchical Search Algorithms (HSA) for motion estimation such as Three Step Search.

Motion estimation is computationally the most intensive part of video compression. Motion estimation's contribution to the total computational complexity of the H.264 video codec is estimated to be between 66% and 94% [6]. The quality of motion estimation, however, is critical to video compression as it can lower significantly the amount of bits required to encode the video. We show that the algorithm proposed herein is able to achieve energy savings as high as 70% exhibiting a 1.8 dB improvement over the conventional algorithm at such high energy savings, this being a significant improvement for motion estimation.

In the sections that follow, Section 2 briefly discusses the prior work in the area of probabilistic computing and design of motion estimation algorithms. Section 3 provides background for motion estimation and PCMOS modeling. Section 4 describes the proposed algorithm with a brief discussion of the PCMOS modeling of the architectures for the conventional and proposed algorithms. This is followed by results in Section 5. Section 6 concludes the paper.

2. Prior Work

Prior work in the domain of probabilistic computing has shown that energy savings can grow exponentially with errors for PCMOS based gates [2,3]. Methods for realizing PCMOS based architectures for various applications including motion estimation have been proposed in [4,5]. Significant efforts have been made to characterize thermal noise based probabilistic primitives such as logic gates, probabilistic adders and multipliers to realize a more efficient model to predict the error rates for these probabilistic primitives [7,8].

Motion estimation is a computationally intensive procedure, and extensive literature exists on low power design for motion estimation algorithms and architectures [9,10,12]. However, the potential of voltage scaling for low power motion estimation, and the design of motion estimation algorithms and architectures under process variations, e.g., thermal noise arising due to voltage scaling, is a less explored domain.

Varatkar et al. [11] propose voltage scaling for low power motion estimation with errors occurring because of delay based variations (due to, e.g., scaled supply voltage or process variations). We use a different noise model that emulates hardware faults such as false bit flipping due to thermal noise possible in future CMOS technology nodes. We propose a fault tolerant design for the previously established algorithm that does not require parallel architectures running at different supply voltages simultaneously for error correction as in the case of [11], which also differs from our approach in [5]. Another significant difference from our previous work in [5] is that

we address fault tolerant techniques for HSAs like Three Step Search in this paper whereas the work in [5] dealt with low power design with probabilistic computing for FSBMA which has far more computations than an HSA.

3. Background

This section provides a background for (i) the motion estimation algorithm, (ii) its associated architecture used in this paper, (iii) modeling PCMOS circuits and (iv) our method used for error rate estimation through such circuits.

A. Motion Estimation

Motion estimation describes the transformation of two subsequent frames in a video sequence in terms of displacement vectors. These displacement vectors map the movements in subsequent frames in the video and are known as motion vectors. Video compression through motion estimation is achieved by coding the video sequence in terms of these motion vectors.

The most popularly employed technique for motion estimation is block-matching. In block-matching the current frame to be encoded is partitioned into non-overlapping macro-blocks of size $N \times N$ pixels (e.g., $N=16$). For each macro-block in the current frame, a matching macro-block is searched in the previous frame. A motion estimation algorithm efficiently searches for the best matching macro-block by selecting the positions of candidate blocks in the previous frame most likely to be a match. The best match is the macro-block with the minimum Sum of Absolute Differences (SAD). SAD is calculated by summing up the absolute difference between pixel luminance values of the current macro-block ‘a’ and the corresponding pixel luminance values of the candidate-macro-block ‘b’.

$$SAD = \sum_j^N \sum_i^N |a(i, j) - b(i, j)|$$

The relative position of the current macro-block and the best match for it determines the motion vector.

B. Three Step Search (TSS) Algorithm

Many variants of motion estimation algorithms exist as these are not standardized by the video codec. The Full Search Algorithm is the most optimal algorithm in terms of visual quality [5]; however, sub-optimal search algorithms are widely used for a faster search requiring less hardware for computation. The Three Step Search algorithm (TSS) belongs to the class of HSAs for motion estimation which perform a coarse to fine search [12].

In TSS, the search area size is usually fixed to be $(\pm 7, \pm 7)$ around the position of the current macro-block. It begins with an initial step size of $\Delta = 4$. Nine candidate macro-block locations, positioned symmetrically around the position of the current macro-block, are first evaluated to find the winner candidate block. The winner candidate is decided to be the one with the minimum SAD among the nine candidate locations. The step size is then halved in the next step. The position of the candidate with the minimum SAD in the previous step is the new reference position for a finer search. The search is carried out in three steps which explains the origin of the name ‘three step search.’ A possible outcome of the TSS algorithm is illustrated in Fig. 1. The positions of candidate macro-blocks decided by the algorithm are known as search points. The search points are shown in Fig. 1.

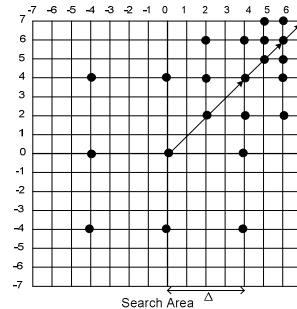


Fig. 1: Three-Step Search Algorithm

C. Architecture for TSS Algorithm

We use the tree architecture proposed in [13] to implement the unit that calculates SAD for the TSS algorithm. The SAD calculations make up the datapath architecture for motion estimation. The datapath architecture for TSS accounts for 60% of the involved processing in the entire motion estimation block which includes the control logic, address generation unit, etc. [9].

Tree architectures are popular for HSAs as the pixel data common between two candidate blocks is significantly reduced. The elements in the pipeline stages of the tree can be decreased or increased according to the processing requirements of the system being implemented. We choose a tree architecture suitable to process a frame size of 352×288 at 25 fps, which is required by most videophones today. The tree architecture used is shown in Fig. 2. Elements D, A, Acc and M function as absolute difference unit, adder, accumulator and comparator respectively as shown in Fig. 3.

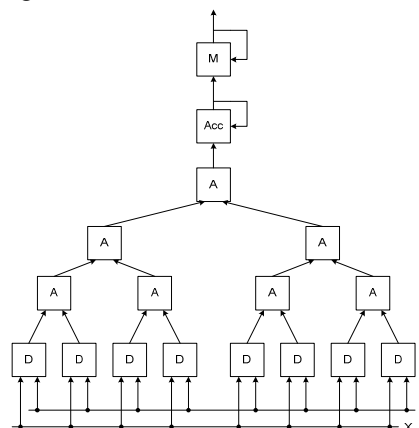


Fig. 2: Tree Architecture for Three-Step Search Algorithm

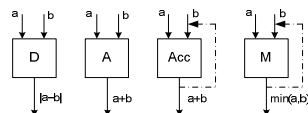


Fig. 3: Elements of the Tree Architecture

D. PCMOS Circuit Modeling and Error Estimation

Korkmaz et al. [2,3] proposed a model for a PCMOS gate by coupling noise sources at the outputs of the deterministic version of the gate being modeled. The approach used in [2,3] goes with the assumption that the equivalent noise source at the output estimates the impact of the noise present in each gate or transistor in an actual noisy circuit. These noise sources are modeled by a Voltage Controlled Voltage Source (VCVS) whose gain is determined by the distribution considered for modeling the noise. References [2,3] use a Gaussian distribution of an

appropriate RMS to model the thermal noise. Fig. 4 shows a model of a noisy Probabilistic (Pr.) FA built using a Deterministic (Det.) FA. The error rates for the outputs of a probabilistic gate/circuit are determined by simulating the gate/circuit in HSPICE for a large number of inputs and comparing its output with the deterministic version of the gate. The error rate is then calculated as the number of incorrect computations divided by the total number of times data is provided to the input.

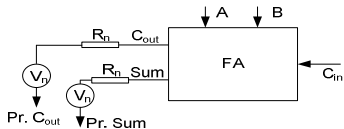


Fig. 4: Probabilistic Full Adder

Using HSPICE simulations to determine error rates is a time consuming and tedious process for larger circuits. Bhanu et al. [7] and Singh et al. [8] proposed methods to predict the error rates observed at the outputs of PCMOS circuits for larger cascade circuits such as a Ripple Carry Adder (RCA) or a Wallace Tree Multiplier (WTM). In a cascade circuit, the interconnections of the gate account for a filtering of the noise waveform. This phenomenon is described in detail in [8]. In the method proposed by Singh et al. [8] to predict the error rates of cascade circuits, the error rate for each gate in the circuit is measured using a *three-stage-model* for the gate shown in Fig. 5. The Pr. Gate is the noisy gate for which the error rates are being calculated and has been modeled as shown in Fig. 4 for the case of an FA. The Det. Gate in Fig. 5 is the non-noisy version of the Pr. Gate. The Filter used in Fig. 5 is a non-noisy version of the gate which is connected to the Pr. Gate in the actual cascade circuit. It is called a filter because a filtering of the noise waveform dependent on the propagation delay of the filter gate is observed in HSPICE simulations. The load is a non-noisy version of the gate connected to the filter in the cascade circuit. To determine the error rates for the Pr. Gate, the outputs of the Filter gates for the two configurations shown in Fig. 5 are compared. The error rate is calculated as the number of non-matching outputs divided by the total number of input vectors. With a three-stage-modeling of all possible unique configurations in the circuit, the error rate for each gate in the circuit is determined. C-simulations of the circuit with false flipping of gate outputs according to the calculated error rates to determine the final error probability of cascade circuit results in error rates which match what is observed when the entire cascade circuit is simulated in HSPICE, details of which are given in [7,8].

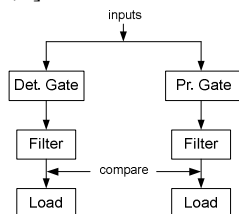


Fig. 5: Three-Stage-Model to Estimate Error Rates for Pr. Circuits

Most gate configurations (three-stage-models) are repetitive in cascade circuits, largely reducing the number of unique configurations. This type of modeling for calculation of error rates is much less time consuming than HSPICE simulations for the entire cascade circuit and

provides a means to map the behavior of probabilistic circuits into C code for testing their effect on applications such as motion estimation, JPEG, etc.

4. Proposed Methodologies

We target the datapath architecture (tree architecture) of TSS to analyze the possible energy savings and effect of probabilistic computing on the performance of motion estimation. In this section we propose an algorithmic modification to TSS which was found to perform better than TSS with probabilistic computing. We also describe the architecture that we will use to implement the proposed algorithm. This is followed by a short description of the PCMOS model of the architecture for both the TSS and the proposed algorithm.

A. Multiple Candidate TSS (MC-TSS) Algorithm

Under the PCMOS model, computations may be erroneous because of false bit flipping due to thermal noise. We propose a Multiple Candidate Three Step Search (MC-TSS) instead of the conventional TSS algorithm. In the conventional TSS algorithm, an incorrect computation of the SAD or an incorrect decision from the comparator can result in selecting an incorrect location for the next steps with finer search. Since the steps begin with a coarse search, a faulty selection of the location could result in significant degradation in PSNR.

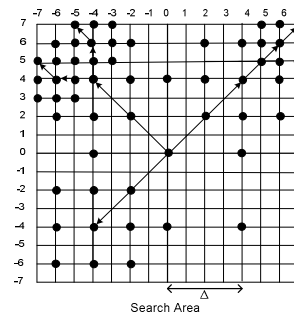


Fig. 6: Multiple Candidate Three-Step Search Algorithm

The Multiple Candidate TSS evaluates nine candidate locations in the first step to select three winner candidate locations with the least SAD. The next step involves a finer search around all three winner candidates to select the next three winner candidates. In every step three locations with the least SAD are kept for the finer search, so there may arise cases in which two or more winner locations could belong to the same group as can be seen in the top left corner of Fig. 6. The flow diagram for MC-TSS is shown in Fig. 7. Furthermore, we also suggest a modification in the calculations of SAD. The SAD is calculated as

$$SAD = \sum_j^N \sum_i^{N/2} |a(2i, j) - b(2i, j)|$$

The above equation shows that only alternate pixel values will be used to compute the absolute differences for the SAD which reduces the number of computations by half. The SAD calculations are halved to make up for the extra calculations performed by MC-TSS over TSS. The candidate macro-block locations are known as search points. The total number of search points in MC-TSS is 57 as shown in Fig. 6 as opposed to 25 in TSS (Fig. 2). When the number of SAD calculations is halved, the total number of calculations for determining the motion-vector for each macro-block through both algorithms is almost equal.

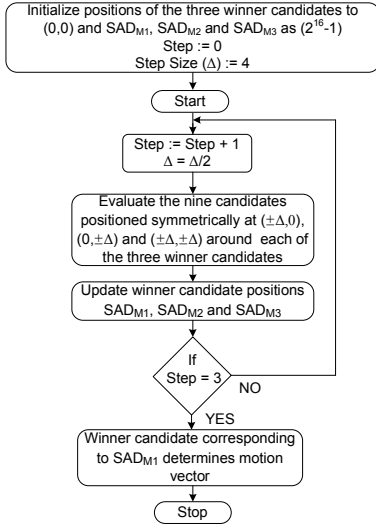


Fig. 7: MC-TSS Flow Diagram

B. Architecture for MC-TSS

The architecture for MC-TSS is also the tree architecture with a simple modification for the comparator and register unit that stores the minimum SAD. The required number of comparators increases to three. In Fig. 8, SAD_C corresponds to the SAD of the candidate block, and SAD_{M1} , SAD_{M2} and SAD_{M3} correspond to the three least SADs.

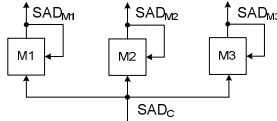


Fig. 8: Comparator Unit for MC-TSS

The number of register units required to store the least SADs also increases to three. The movement of data between these registers is dependent on the outcome of the three comparators. The logic to implement this is shown in Fig 8.

```

IF  $SAD_C < SAD_{M1}$ 
THEN  $SAD_{M3} = SAD_{M2}$ 
       $SAD_{M2} = SAD_{M1}$ 
       $SAD_{M1} = SAD_C$ 
ELSE IF  $SAD_C < SAD_{M2}$ 
THEN  $SAD_{M3} = SAD_{M2}$ 
       $SAD_{M2} = SAD_C$ 
ELSE IF  $SAD_C < SAD_{M3}$ 
THEN  $SAD_{M3} = SAD_C$ 

```

Fig. 9: Logic for Data Movement between Register Units

The logic described in Fig. 9 can be implemented with the help of shift registers. In Fig. 10, Sign_bit1, 2 and 3 are provided by the comparator unit. Fig. 10 describes the shift register unit for the j^{th} bit of SAD_C , SAD_{M1} , SAD_{M2} and SAD_{M3} , and the gate level implementation of the logic required for movement of SAD values between registers dependent on the Sign bits provided by the comparators. This unit is replicated sixteen times for all the 16 bits of the SADs.

C. PCMOs Modeling of Datapath Architecture

The tree architecture in Fig. 3 consists of adders of bit widths ranging from 8 to 16 bits. These adders are modeled as ripple carry adders (RCA). Pipelined stages of the tree architecture are required to run sequentially on a clock. So, blocks A, Acc and M include internal registers to facilitate the sequential processing.

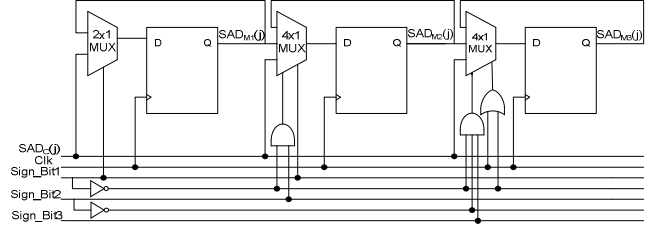


Fig. 10: Shift Register Unit for Data Movement

These registers are modeled with the help of D-flip flops. The architecture for the TSS algorithm consists of Full Adders, D-flip flops, inverters and EXOR gates. The inverters and EXOR gates are used in block D of Fig. 3 for calculation of absolute difference. The architecture for MC-TSS algorithm also requires NAND, MUX NOR and INV gates, which are required to implement the logic in Fig. 10. The PCMOs modeling involves coupling a noise source at the output of the constituent gates of the architecture. The error rates for the gates are measured using the *three-stage model* described earlier in Section 3.B [8]. The HSPICE simulations required to measure the error rates for noisy Pr. gates are carried out using the Synopsys 90nm generic library with a nominal supply voltage of 1.2 V. As the supply voltage is scaled down, the energy consumption through the architecture decreases but the magnitude of the errors increase. Energy consumption for the architecture is calculated using HSPICE simulations of the transistor level netlist of the entire architecture. A detailed discussion follows in Section 5.

5. Results

This section provides simulation results comparing the performance of TSS and MC-TSS under probabilistic computing in terms of the possible energy savings and quality of motion estimation.

A. Experimental Set-Up for PCMOs based Datapath Architecture and Energy Estimation

To estimate the energy consumption through the PCMOs based datapath architecture, we develop a transistor level netlist for the architecture shown in Fig. 2 for TSS and Fig. 2 with modifications suggested in Fig. 8 and Fig. 10 for MC-TSS. The HSPICE simulations of the netlist are carried out using the Synopsys 90nm generic library. The noise modeling is done only for Full Adders and D-flip flops out of all the gates present in the architecture (discussed in Section 4.C) as most of the processing occurs through these gates and they account for about 90% of the total energy consumption through the architecture. All the Full Adders in the circuits are designed using a 24-transistor mirror adder circuit [14], and the D-flip flops are designed using the edge-triggered D-flip flop circuit provided in [15].

The gain of the VCVSes, used as noise sources for PCMOs modeling of the gates, is modeled with a Gaussian distribution of appropriate RMS. As we are predicting with a noise RMS a possible behavior of a future noisy silicon technology node (e.g., 16nm) [2,3], the choice of noise RMS is made such that no errors are observed at the output of the gates at the nominal supply voltage of 1.2 V, which was found empirically to be 0.2 V. Supply voltage is then scaled down to 1.15 V, 1.1 V down till 0.7 V with a step size of 0.05 V.

Table 1: Energy savings with PSNR loss less than 0.5 dB

Video Sequences	Base Case: TSS algorithm at 1.2 V and no energy savings PSNR (dB)	TSS			MC-TSS		
		PSNR (dB)	Energy Savings	Circuit Supply Voltage (V)	PSNR (dB)	Energy Savings	Circuit Supply Voltage (V)
Susie	35.64	35.21	40%	1.05	35.43	55%	0.95
Mobile Calendar	23.72	23.23	57%	0.95	23.36	70%	0.85
Flower Garden	25.2	24.74	57%	0.95	25.02	70%	0.85
Foreman	31.3	31.035	49%	1.00	30.89	64%	0.90

Energy consumption is calculated by simulating the circuit of the architecture for the scaled voltage values. The inputs to the architecture netlists are test vectors from standard video sequences which are also used to test the performance of motion estimation.

The HSPICE simulations required for error estimation are also carried out by simulating the unique *three stage model* of the Full Adders and D-flip flops in the architecture; four unique three-stage configurations for Full Adders, two in case of TSS and three in the case of MC-TSS for D-flip flops were found. Error rates for these gates are measured for the required range of voltage values considered.

B. Experimental Set-Up for Motion Estimation

The input video sequences to motion estimation are standard CIF video sequences of size 352x288 at 25 fps. The search area size used is 7 as required by the TSS and MC-TSS algorithms. The block size used for the experiments is 16. The simulations are carried out in the MPEG-2 Test Model 5 codec, at main profile and main level [16]. The performance of the two cases is evaluated by fixing the average quality degradation Δ PSNR of the motion compensated frame within 0.5 dB from the base case. PSNR is calculated as

$$PSNR = 10 \times \log \left(\frac{255^2}{(1/(H \times W)) \sum_{i,j}^{H,W} (F_I(i,j) - F_{MC}(i,j))^2} \right)$$

where H and W are the dimensions of the frame. $F_I(i,j)$ and $F_{MC}(i,j)$ are the pixel luminance values for the input and the motion compensated frames.

The modeling of probabilistic gates proposed in [7,8] shows that the error rates calculated by HSPICE simulations for the constituent gates of larger PCMOs circuits such as RCA or WTM can be used to predict the error rates for the entire circuit using C-based simulations. Thus, C-based simulations to model errors in the SAD calculations can be used to determine the error tolerance of motion estimation algorithm. The calculation of SAD through the tree architecture is modeled in the C code for motion estimation as it would occur in the actual hardware according to gate level implementation. False bit flipping is then introduced for the gates by using the error probabilities obtained from HSPICE simulations. The error probabilities increase with the scaling of the supply voltage to the circuit. The appropriate supply voltage is chosen as the one that limits the quality degradation of motion estimation to 0.5 dB. This procedure to model errors in the motion estimation code was also used in [5].

C. Experimental Results

The possible energy savings for TSS and MC-TSS with voltage scaling when the quality reduction is limited to 0.5 dB are tabulated in Table 1. The search area size is limited to 7 for TSS, which makes it an appropriate algorithm for applications such as video conferencing which need faster processing and in which the movement of objects in the video is small. Thus, the video sequences considered for showing the results in Table 1 are slow and medium motion video sequences in which movement of objects is mostly limited to the search area size of 7. The circuit supply voltages corresponding to the possible energy savings with these video sequences are also provided in Table 1. Motion estimation as an application exhibits high resilience to probabilistic computing, and the decrease in PSNR is much less compared to the increase in energy savings. This is because the SAD is evaluated as a summation of 256 differences for a block size of 16. Probabilistic bit computation affects only a small number out of these differences and, hence, the overall effect of incorrect computations is minimized. However, note that in the case of MC-TSS, the number of SAD computations are halved decreasing these to 128 differences per SAD (block size is still 16). We surmise that the main reason that MC-TSS does better than TSS is because the nature of MC-TSS to keep three winners for each step increases the percentage of errors that MC-TSS can tolerate over TSS.

D. Working of MC-TSS and Comparison with TSS

The working principle that causes MC-TSS to outperform TSS is that if an incorrect computation of SAD or an incorrect decision from the comparator leads to an incorrect selection of the winner candidate for a given step, it is quite likely that the correct winner candidate will be captured by the next best candidates. Table 2 shows the percentage of the times the correct winner candidate is likely to be amongst one, three and four best candidates for the choice of operating voltages from Table 1 required for the 0.5 dB constraint on quality degradation.

Table 2: Percentage of the times the correct winner candidate is present amongst 1, 3, or 4 best candidates

Video Sequence	Circuit Voltage	Best-1	Best-3	Best-4
Susie	0.95	92.54%	98%	98.8%
Mobile Calendar	0.85	89%	96%	97.15%
Flower Garden	0.85	82%	94.21%	96.35%
Foreman	0.90	92.47%	97.26%	98%

It can be seen from Table 2 that increasing the winner candidates of MC-TSS to four hardly introduces any

difference over keeping three, as the percentage increase of finding the correct winner candidate in the best four over best three candidates is a meager 0.8% to 2.14%; this trend continues for lower voltage values as well. Thus, keeping three winner candidates for every step in MC-TSS is an apt choice. Fig. 11 shows the improvement in visual quality when the energy consumption through architectures for both TSS and MC-TSS is approximately same. Notice that in Fig. 11 the distortion in the frames is mostly in the moving parts of the frame: this is because incorrect computation of SAD values affects the decision for the moving parts of the frame more than the stationary ones. Fig. 12 shows the variation of PSNR with the frames of the video sequence for the example video sequence ‘flower garden.’ It can be seen from Fig. 12 that not only is the achievable PSNR via MC-TSS higher, the variation in PSNR for the MC-TSS algorithm with probabilistic computing is much less than TSS with probabilistic computing, highlighting MC-TSS as a more stable algorithm over TSS with inexact computation. Fig. 13 shows the variation of average PSNR for four example video sequences with the operating voltage supply for the architecture. It can be seen that the decrease in PSNR with increase in errors due to voltage scaling is steeper in the case of TSS as compared to MC-TSS.

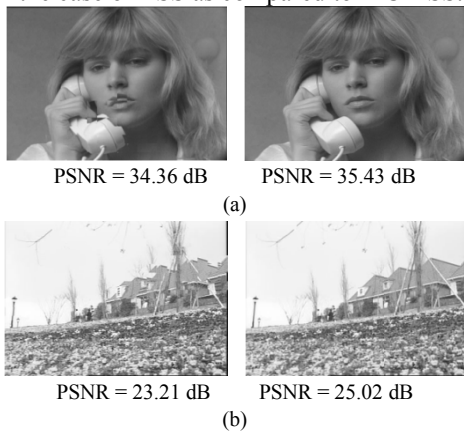


Fig. 11: TSS (left) and MC-TSS (right) at approx. same energy savings for video sequence (a) Susie (55% savings) (b) Flower Garden (70% savings)

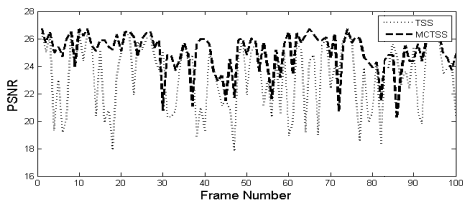


Fig. 12: PSNR variation for TSS and MC-TSS Algorithm over the frames of the video sequence ‘Flower Garden’ at approx. energy savings of 70%

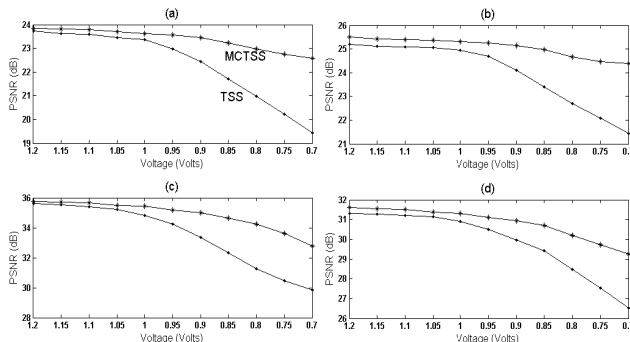


Fig. 13: PSNR v/s Voltage Scaling for (a) Mobile (b) Flower Garden (c) Susie (d) Foreman

6. Conclusion

First of all, with both standard prior art (TSS) and our new algorithm (MC-TSS), we have demonstrated the applicability of probabilistic computing. More significantly, we have demonstrated that the proposed algorithm, MC-TSS, outperforms the previously established TSS algorithm when computation is inexact. An increment of 1.81 dB is noticed in the quality of motion estimation when the energy savings through both algorithms is the same, this being a significant increase for motion estimation. Also, under the limit of 0.5 dB for quality reduction, energy savings increase by about 13% to 15% with MC-TSS over that achievable through TSS with overall energy savings as high as 70%. Thus, algorithmic modifications such as the one proposed in this paper can result in better error resilience to probabilistic computing while capitalizing on the energy savings and area reductions that the future technology nodes can provide.

7. References

- [1] L. B. Kish, “End of Moore’s law: Thermal (noise) death of integration in micro and nano electronics,” *Physics Letters A*, vol. 305, no. 3-4, pp. 158-168, 2002.
- [2] P. Korkmaz, B. E. S. Akgul, K. V. Palem, and L. N. Chakrapani, “Advocating noise as an agent for ultra-low energy computing: probabilistic complementary metal-oxide-semiconductor devices and their characteristics,” *Japanese Journal of Applied Physics*, vol. 45, no. 4, pp. 3307–3316, Apr. 2006.
- [3] P. Korkmaz, B. E. S. Akgul, and K. V. Palem, “Energy, Performance and Probability Trade-offs for Energy-Efficient Probabilistic CMOS circuits,” *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 8, pp. 2249-2262, Sep. 2008.
- [4] L. Chakrapani, B. E. S. Akgul, S. Cheemalavagu, P. Korkmaz, K. Palem, and B. Seshasayee, “Ultra-Efficient (Embedded) SOC Architectures based on Probabilistic CMOS (PCMOS) Technology,” *Proceedings of Design Automation and Test in Europe (DATE '06)*, vol. 1, no. 1, pp. 1-6, Mar. 2006.
- [5] C. Dhoot, V. J. Mooney, L. P. Chau, and S. R. Chowdhury, “Low Power Motion estimation with Probabilistic Computing,” *International Symposium on Very Large Scale Integration (ISVLSI '11)*, pp. 176-181, July 2011.
- [6] P. Kuhn, “Complexity Analysis and VLSI Architectures for MPEG-4 Motion estimation,” Boston, MA: Kluwer, 1999.
- [7] A. Bhanu, M. S. K. Lau, K. V. Ling, V. J. Mooney III, and A. Singh, “A More Precise Model of Noise Based PCMOS Errors,” *Fifth IEEE International Symposium on Electronic Design, Test & Applications (DELTA '10)*, pp. 99-102, Jan. 2010.
- [8] A. Singh, A. Basu, K.V. Ling and V. J. Mooney, “Modeling Multi-output Filtering Effects in PCMOS,” *Proceedings of the VLSI Design and Test Conference (VLSIDAT '11)*, pp. 414-417, April 2011.
- [9] R. S. Richmond II and D. S. Ha, “A low-power motion estimation block for low bit-rate wireless video,” *Proceedings of IEEE Workshop Signal Processing*, pp. 60-63, Aug. 2001.
- [10] M. A. Elgamel, A. M. Shams, and M. A. Bayoumi, “A comparative analysis of low power motion estimation VLSI architectures,” *Proceedings of IEEE Workshop Signal Processing*, pp. 149-158, 2000.
- [11] G. V. Varatkar and N. R. Shanbhag, “Error-Resilient Motion Estimation Architecture,” *IEEE Transactions on VLSI Systems*, vol. 16, no. 10, pp. 1399-1412, Oct. 2010.
- [12] T. Koga, K. Inuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion Compensated Interframe Coding for Video-Conferencing” *Proceedings of National Telecommunications Conference*, New Orleans, pp. G5.3.1- 5.3.5, Nov. 1981.
- [13] Y. S. Jehng, L.G. Chen, and T.D. Chieh, “An Efficient and Simple VLSI Tree Architecture for Motion Estimation Algorithms,” *IEEE Transactions on Signal Processing*, Vol. 41, No. 2, Feb. 1993.
- [14] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, “*Digital Integrated Circuits: A Design Perspective*,” 3rd ed. Prentice Hall, 2003.
- [15] M. Morris Mano, “*Digital Logic Design*,” 3rd ed. Prentice Hall, 2004.
- [16] “MPEG-2 Test Model 5. Internet: www.mpeg.org/MPEG/MSSG/tm5.”