DYNAMIC VOLTAGE SCALING ALGORITHMS FOR POWER CONSTRAINED MOTION ESTIMATION

In Suk Chong and Antonio Ortega

Signal and Image Processing Institute, University of Southern California, Los Angeles CA 90089

ABSTRACT

In this paper, we apply dynamic voltage scaling (DVS) to the matching metric computation (MMC) used within motion estimation (ME) in typical video encoders. Our approach is based on "soft DSP" concepts. We analyze the effect of ME errors (due to DVS) in overall coding performance. We propose a model for the resulting rate increase (at a given fixed quantization parameter) as a function of input characteristics and input voltage, for given ME algorithm and MMC architecture. This model is validated using simulations. We then compare ME algorithms and MMC architectures, and propose a method for power saving of the ME process that depend on input characteristics and desired coding performance. As an illustration of the potential benefits of allowing computation errors, we show that allowing errors that lead to a small rate increase (about 3%) produces 37% power savings in the ME process, as compared to not using DVS. An essentially "error-free" DVS approach (no rate penalty) can achieve around 10% power savings.

Index Terms— Dynamic voltage scaling (DVS), error tolerance (ET), Soft DSP, matching metric Computation (MMC)

1. INTRODUCTION

Power (or energy) is the most important design constraint in many VLSI design scenarios [11]. Many approaches have been proposed for power constrained VLSI, ranging from circuit level to architectural and algorithmic level [7, 9]. Dynamic voltage scaling (DVS) is an attractive technique to reduce power consumption, as lowering input voltage by a factor J, reduces energy dissipation by almost a factor J^2 [9]. Soft DSP is an efficient approach for DVS [9] that has been applied to low level systems, such as adders and multiplier-accumulators (MACs) often used in signal processing applications (e.g., linear filters and multi-input-multi-output, MIMO, systems). In soft DSP systems the input voltage is below critical voltage (i.e., we have voltage over scaling, VOS), which leads to input-dependent soft errors. Then, soft-error tolerance is achieved by using explicit error control blocks that provide error concealment so as to operate with negligible loss in algorithm performance.

In our previous work, we have shown that image/video compression systems exhibit error tolerance (ET) characteristics, *even if no explicit error control block is added*, and this under both hard errors (due to deterministic faults) [6, 5] and soft errors (due to DVS) [3]. Errors due to VOS in these applications are either i) concealed by other parts of the system (e.g., quantization can conceal errors affecting a transform computation) or ii) are "acceptable" [2]. Determining what constitutes acceptable errors is obviously an application-specific decision; both performance criteria and acceptability thresholds are highly dependent on the application. In [5], we studied the impact of hard errors on matching metric computation (MMC) within the motion estimation (ME) process in a video compression system. In this case, we showed that both video encoder and decoder remain operational, and thus these errors can be evaluated in terms of the compression performance penalty they produce (i.e., more bits are needed to code data at a given quality level as compared to the bit-rate required by a fault-free system operating at the same QP). This performance penalty may be acceptable for specific application scenarios. We have also provided a primarily *experimental* evaluation of the behavior of several ME algorithms under "soft error" conditions applying soft DSP approaches to MMC within ME process [3].

This paper is based upon work supported in part by the National Science Foundation under Grant No. 0428940. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

In this paper, we extend our previous work [3] to a DVS scenario. The main novelty comes from i) a model for degradation in video coding performance due to voltage scaling, as a function of input characteristics and for given ME algorithms and MMC architectures (this model can used to select input voltage values for target coding performance criteria), and ii) using this model to compare various ME algorithms and MMC architectures in terms of their coding performance under DVS. Our proposed models for DVS performance are designed to be used in hardware-based video encoders, but could also prove useful in the context of general purpose processors for which power control is enabled (see [8] for an example). Since ME is performed at the encoder, our work is primarily applicable to scenarios where power-constrained devices (e.g., cellphones) are used for video capture and encoding.

To introduce our model, we first briefly explain the ME process, introduce different MMC architectures we will use, and describe the basic setting for analysis (Section 2). Each MMC architecture involves several "soft" adders, such as those used in the soft DSP context. We provide a detailed analysis of errors due to voltage scaling for a single adder (Section 3). Then we extend it to model errors in typical MMC architectures and the performance degradation due to soft error as a function of input voltage and input characteristics. This model is validated using simulations (Section 4). Using this model, we propose a voltage control method, which based on our simulations can achieve about 37% power savings in the ME process, as compared to not applying any voltage scaling, with very slight increase in rate (around 3%).

2. MOTION ESTIMATION WITH SOFT ERROR

The ME process comprises a search strategy of the motion vector (ME algorithm) and a matching metric computation (MMC). The search strategy decides a set of candidate MVs and then proceeds to compute the matching metric for the candidates and select the one that minimizes the matching metric (typically, sum of absolute differences (SAD) or sum of squared differences (SSD); in our case

SAD is used).

There are several types of hardware architectures [12] to compute the matching metrics, with different levels of parallelism. We will refer to them as MMC architectures. Among those, we choose a serial and parallel architecture for analysis (see Figure 1). The serial architecture has M^2 serially connected adders for SAD computation between two $M \times M$ macroblocks. It is simple but requires longer running time compared to the parallel one. As shown in Figure 1, the parallel architecture has M parallel groups of "leaf" adders and M "central" adders (in total $M^2 + M$ adders are needed). Each group of leaf adders consists of M adders and computes the sum of M AD values. Then, the central adders compute the final SAD adding up M partial SAD values. In leaf adders, outputs are small compared to the final SAD value (on average partial SAD values in a set of leaf adders will be smaller than $\frac{SAD}{M}$). Thus errors with magnitude larger than $\frac{SAD}{M}$ (small compared to the final SAD) are unlikely to be generated in the leaf adders, because soft errors in an adder cannot be greater than the output of the adder (see Section 2.1.2). Thus we only need to focus on the central adders in parallel MMC architecture case, as this is where the larger errors are likely to happen. These central adders can be modelled as M serial adders whose inputs are the sum of M AD values.

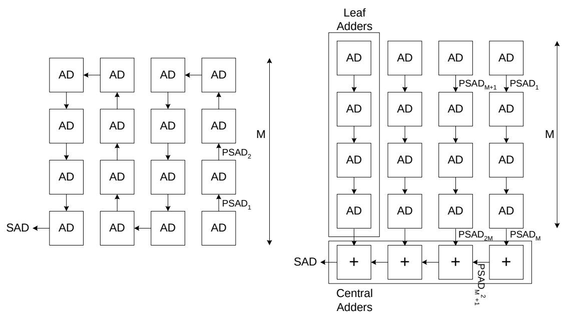

Fig. 1. MMC Architectures. Left: Serial Architecture, Right: Parallel Architecture

For each macroblock of size $M \times M$ in the current frame, the MMC process computes the matching metric for each candidate block in the reference frame's search window; these are denoted SAD_1, $SAD_2, \ldots, SAD_N$ (sorted in magnitude of SAD_i, with SAD_1 the largest one) where N is number of candidates. We define MV_{min} as the best MV, which corresponds to the index such that SAD_i is minimum (here $MV_{min} = N$), and SAD_{min} as a minimum SAD among all SAD_i (here $SAD_{min} = SAD_N$).

When the MMC process operates with Vdd below its normal operating range (i.e., lower than Vdd_{crit}), the above SAD_i values may be corrupted; those possibly erroneous SAD_i values are denoted $SAD'_1, SAD'_2, \ldots, SAD'_N$, where $SAD'_i = SAD_i - E_i$, with E_i denoting the soft error due to voltage control. Denote MV_f the MV chosen when SAD'_i are used. If $MV_f \neq MV_{min}$, the residual block's distortion (as measured by the SAD) increases by $E_{SAD} = SAD_{MV_f} - SAD_{min}$. This increase in distortion (E_{SAD}) may lead to rate increases for a given QP, which we propose to model using the quadratic (Q2) model [4]. This model has been applied in implementations of existing video encoding standards ITU-T H.264/MPEG4 AVC [1] and tends to be accurate for large data sets, such as one Group of Pictures (GOP) or one whole sequence (its accuracy increases with the number of frames being modelled). The main Q2 model is as follows:

$$R = S_1 \frac{MAD}{QP} + S_2 \frac{MAD}{QP^2}, \quad (1)$$

where R is the rate, MAD is the energy of the prediction residual measured in terms of mean absolute difference (MAD), QP is the quantization parameter and S_1, S_2 are parameters to be estimated. One can see that the Q2 Model can be rewritten as a linear function of SAD for a fixed QP. Now we take derivative of both terms. Then the following relation holds for a SAD increase (E_{SAD}) and rate increase (ΔR):

$$\Delta R = X_1 E_{SAD} \quad (2)$$

where X_1 is a parameter to be estimated for each set of frames (i.e., one GOP or whole sequence). Therefore if we know the model for E_{SAD}, we know the model for ΔR. Now we focus on modelling E_{SAD} as a function of Vdd and input characteristics. For this purpose, in the next section we study the characteristics of soft errors in the MMC process (E_i).

2.1. The MMC process with Soft Errors

An MMC system includes several n-bit adders. We assume that ripple carry adders with voltage scaling are used, as they provide useful functionality for DVS [9]. We assume that all soft adders in the MMC process have the same input voltage (Vdd), as is typically assumed in soft DSP techniques [9]. When we decrease Vdd for the adders, the circuit delay for one full adder (T_{FA}) increases, but the sampling time (T_S) remains the same. Thus, R_S, the number of full adder (FA) operations possible in one T_S, will decrease. From now on, we will use R_S instead of Vdd as the parameter that controls the operating point of the system; R_S is a function of Vdd and depends on several gate parameters (see [9]). Here we will use the parameters used in [9]. If the number of FA operations required to complete one addition (i.e., the path delay divided by T_{FA}, which is obviously input dependent) is larger than R_S, then an error is generated. Our target is to model errors (E_i) due to applying $Vdd < Vdd_{crit}$ to the MMC hardware. As a first step, we need to understand the behavior of an n-bit soft adder (n-bit ripple carry adder with voltage scaling).

2.1.1. *n-bit Soft Adder*

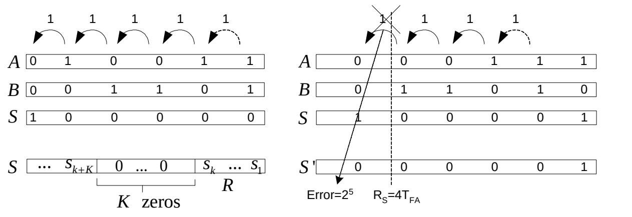

Fig. 2. Upper left: input with path delay equal to $6T_{FA}$, Lower left: output S with path delay equal to $(K+1)T_{FA}$, Right: soft error due to Vdd control

An n-bit ripple carry adder comprises n serially connected full adders. Denote its inputs $A = [a_n ... a_1]$ and $B = [b_n ... b_1]$, and let $S = A + B = [s_n ... s_1]$, with the carry denoted by $C = [c_n ... c_1]$. Each FA has inputs a_i, b_i, c_{i-1} and outputs s_i, c_i, and each input pair (a_i, b_i) is introduced to corresponding FA simultaneously. If a carry c_i is generated in each FA, it is propagated to the next FA (see Figure 2).

If $Vdd < Vdd_{crit}$ an error can be generated if R_S is smaller than the path delay required for the computation. The total *path delay* is determined by the *longest consecutive carry propagations*. Consecutive carry propagations are generated after an initial carry generation (by an $(1,1)$ input pair) is followed by carry propagation inputs ($(1,0)$ or $(0,1)$ input pairs). Thus if an input has a path delay larger than R_S, carry input to the (R_S+k)-th FA is lost (k is the starting position of carry propagation) and an error with magnitude 2^{R_S+k} is generated (see Figure 2).

It is interesting to note that for the path delay to be $K+1$ for one addition, the result, S, has to include a 1 followed by K consecutive 0s (from the $(k+1)-th$ bit to the $(k+K)-th$ bit), i.e., $S = m2^{K+k}+R$, where $m>0$ and $R<2^k$. Thus if i) $S = m2^{K+k}+R$ where $K \succeq R_S$, and ii) $a_{k+1}=1$ (automatically $b_{k+1}=1$), then an error with magnitude 2^{R_S+k} is generated. Here we can see that errors can be no greater than S and that an error cannot be generated if $S<2^{R_S}$. Since a_{k+1} corresponds to a lower significance bit (as compared to S), we can assume that S and a_{k+1} are independent. And if we assume i) $P(S = m2^{R_S+k}+R)$ is similar for all R (S has smooth distribution), and ii) that $p(a_{k+1}=1)=\frac{1}{2}$, we can model the error in a soft adder as:

$$P(error = 2^{R_S+k}) = \sum_{m=1}^{T} P(S = m2^{R_S+k})2^{k-1}, \qquad (3)$$

where $T = \lceil \frac{2^n}{2^{R_S+k}} \rceil$ and n is the width of the adder.

2.1.2. *Soft Error in the MMC process*

If at least one of the intermediate soft adders satisfies the condition for error generation, an error occurs. This condition depends on R_S and on the outputs of intermediate adders, each of which is a partial SAD of the $l-th$ node ($PSAD_l$). Thus we can model the error, E_i, given SAD_i, if we know characteristics of $PSAD_l$. In [10], given a final SAD (SAD_i), characteristics of $l-th$ partial MAD ($PMAD_l$) are modelled as random variables with mean $\frac{SAD_i}{M^2}$. Since $PSAD_l$ is a multiple of $PMAD_l$, a model for $PSAD_l$ can be easily derived for a given final SAD. We can then derive a model for E_i; this will be the probability of error for the single-adder case summed over all intermediate nodes, i.e.,

$$P(E_i = 2^{R_S+k}) = \sum_{m=1}^{T}\sum_{l}^{L} P_{PSAD_l}(PSAD_l = m2^{R_S+k})2^{k-1} \qquad (4)$$

Here we assume that an error occurs on only one intermediate node so that $E_i = 2^{R_S+k}, k=0,1,....$ From our simulations, we observed that the probability that errors occur in more than one node tends to be negligible. We make use of this single-error assumption as it simplifies the modelling and the experimental results validate it.

Any given SAD_i value can be produced by many different combinations of intermediate node outputs ($PSAD_l$). Thus, since E_i depends on those intermediate computations, different E_i can be generated for a given SAD_i value. We model E_i as a random variable which is independent of SAD_i if $SAD_i \geq 2^{R_S}$ (if $SAD_i < 2^{R_S}$ then $E_i = 0$), but cannot take arbitrary values. In particular we will have that $E_i \leq SAD_i$; our simulation shows that correlation between SAD_i and E_i is very small (< 0.03).

3. CODING PERFORMANCE MODELS

With our proposed model for E_i we can now derive an analytical model for E_{SAD} (the expected value of the increase in prediction residual SAD) as a function of SAD_i and R_S. For each SAD_i there is a possibility that due to an error, i will be chosen as the MV ($P(i = MV_f)$), instead of the correct vector. When this happens the SAD of the prediction residual increases by $SAD_i - SAD_{min}$. But only $SAD_i \geq 2^{R_S}$ can result in a error and thus lead to an erroneous MV choice. Thus we define Q as the set of SAD_i for which errors can occur so that i is selected as MV_f ($Q = \{i | 2^{R_S} \leq SAD_i\}, N_Q = |Q|$). Then E_{SAD} can be written as follows:

$$E_{SAD} = \sum_{i\in Q}(SAD_i - SAD_{min})P(i = MV_f), \qquad (5)$$

To estimate E_{SAD} according to (5), we evaluate $P(i = MV_f)$ first. For $i = MV_f$, SAD'_i needs to be smaller than all other SAD'_w, which can be stated as follows:

$$P(i = MV_f) = \prod_{w\neq i} P(SAD'_w > SAD'_i) \qquad (6)$$

This equation is based on the observation that each error (E_w) is almost independent of corresponding SAD value (SAD_w) (see Section 2.1.2). For i to be chosen as the MV_f, SAD'_i should be the minimum among all SAD'_w, so that $SAD'_i < SAD_{min}$ and thus the following holds:

$$P(i = MV_f) = \int_0^{SAD_{min}} P(SAD'_i = x)\prod_{w\neq i} P(SAD'_w > x)dx \qquad (7)$$

Here we can derive a simple expression of $P(SAD'_w < x)$ which is a linear function of x, assuming that $\sum_l^L P_{PSAD_l}(PSAD_l = m2^{R_S+k})$ (probability that the value of one of the intermediate outputs ($PSAD_l$) is $m2^{R_S+k}$) is independent of k and takes a constant value p_0.

$$P(SAD'_w < x) \approx \frac{x}{2^{R_S+1}}p_0 \qquad (8)$$

Thus applying (8) to (7), we obtain:

$$P(i = MV_f) \approx \frac{1}{N_Q}(1-(1-\gamma\frac{SAD_{min}}{2^{R_S}})^{N_Q}), \gamma = \frac{p_0}{2} \qquad (9)$$

Generally p_0 will depend on the MMC architecture. In a serial architecture there are M^2 intermediate nodes to consider, while we only need to consider M nodes in a parallel architecture (the central nodes, as discussed in Section 2.1.2). If the number of intermediate nodes is larger for the same final SAD, there are more chances for one of the intermediate nodes to be $m2^{R_S+k}$. Thus p_0 corresponding to a serial MMC architecture is larger than for a parallel one. These p_0 values can be precalculated for given SAD_w, R_S, and MMC architectures. Now we apply (9) to (5) to get the following expression for E_{SAD}:

$$E_{SAD} \approx (\overline{SAD_Q} - SAD_{min})(1-(1-\gamma\frac{SAD_{min}}{2^{R_S}})^{N_Q}) \qquad (10)$$

where $\overline{SAD_Q}$ is a mean SAD value over set Q, and $\Delta R = X_1 E_{SAD}$.

4. SIMULATION AND DISCUSSION

To evaluate our proposed model, the Foreman and Stefan sequences were tested. We simulated the effect of a series of R_S values using an H.264/AVC baseline profile encoder with full search/EPZS ME algorithms and serial/parallel MMC architectures. Only 16×16 block partitions and a single reference were considered for ME; QP was fixed and rate distortion optimization was turned on. We assign 15 frames to each group of pictures (GOP), and use an IPPP GOP structure. We collect real rate increase (ΔR) data by encoding each GOP with/without errors (for $R_S = 5, 7, ...15$ and Vdd_{crit} corresponding to $R_S = 16$). We estimate ΔR with E_{SAD} computation and X_1 estimation for each GOP; E_{SAD} is computed by collecting SAD_i statistics during the normal encoding operation without errors, and X_1 is estimated by evaluating the ratio of real ΔR and computed E_{SAD} for one specific R_S point with nonzero E_{SAD} (R_S^L). Figure 3 shows the variations of ΔR as a function of Vdd, which will be useful to design a power control mechanism. This result shows that we can precisely estimate ΔR with our analytical model in the R_S range of interest.

Using simulation result and model, we can compare MMC architectures and ME algorithms. The slope of ΔR for serial MMC architecture is larger than one of a parallel architecture, because p_0 of a serial MMC architecture is larger (see Section 3). But the saturated ΔR value is similar because it only depends on $SAD_{min}, \overline{SAD_Q}$, which is the same for both cases. Since a EPZS search strategy uses a good prediction algorithm to select a small number of MV candidates, which are already near the minimum SAD point, EPZS has smaller $N_Q, \overline{SAD_Q}$ than the full search algorithm. Thus ΔR of EPZS is always smaller than that for the full search case. In summary, EPZS search algorithm and parallel MMC architecture are better than full search algorithm and serial MMC architecture respectively. Note that we do not consider the inherent difference in complexity, regularity, and memory usage between EPZS and FS algorithm; FS algorithm has more searching points but has more regular structure and memory usage than EPZS.

If we can estimate ΔR as a function of R_S for a given sequence before encoding, we can control Vdd in a optimal fashion during the encoding process, thus saving power. A normal video encoder optimization scheme only considers rate and distortion. But in encoding scenarios that require small power consumption, e.g., hand held devices, we need to take power consumption into consideration by adding this to the cost function. To estimate ΔR data, we need information about SAD_i (SAD value for each MV candidate in one macroblock). Since information about SAD_i is not available before encoding, it can be estimated, for example, by encoding without DVS a single frame within a GOP. This approach will be effective if E_{SAD} and X_1 do not change much in within one GOP. Selecting an optimal Vdd point can be done using various optimization techniques, such as those based on lagrange multipliers. A heuristic method would be to choose a threshold for rate change (ΔR_{th}), and select Vdd such that estimated ΔR is less than ΔR_{th}. Using these algorithms, we can change Vdd dynamically depending on input characteristic with a slight additional complexity in the encoding system, but with potentially large power savings. Figure 4 highlights the potential for savings in the ME process using DVS; setting $\Delta R_{th} = 0.1$, leads to 37% power reduction when using EPZS and the parallel MMC architecture. Considering that a significant percentage of power consumption is due to the ME process (at least 20% in case of MPEG2 encoder), total power savings within the video encoding system can be significant.

5. REFERENCES

[1] Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC in Joint Video Team(JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.

[2] M. A. Breuer, S. K. Gupta, and T. M. Mak. Defect and error tolerance in the presence of massive numbers of defects. *IEEE Design & Test of Comp.*, 21:216–227, May–June 2004.

[3] H. Cheong, I. Chong, and A. Ortega. Computation error tolerance in motion estimation algorithms. In *IEEE International Conference on Image Processing, ICIP'06*, Oct. 2006.

[4] T. Chiang and Y. Q. Zhang. A new rate control scheme using quadratic rate distortion model. *IEEE Trans. Circuits Syst. Video Technol.*, 7(1):246–250, Feb. 1997.

[5] I. Chong and A. Ortega. Hardware testing for error tolerant multimedia compression based on linear transforms. In *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT'05*, pages 523–534, 2005.

[6] H. Chung and A. Ortega. Analysis and testing for error tolerant motion estimation. In *Proc. of IEEE Int. Symp. on Defect Fault Tolerance in VLSI Syst.*, pages 514–522, 2005.

[7] E. Debes. Recent changes and future trends in general purpose processor architectures to support image and video applications. *Proceedings of the 2003 IEEE International Conference on Image Processing (ICIP'03)*, 3:85–88, 2003.

[8] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: A low-power pipelined based on circuit-level timing speculation. *MICRO-36*, Dec. 2003.

[9] R. Hedge and N. R. Shanbhag. Soft digital signal processing. *IEEE Trans. on VLSI systems*, 9(6), 2001.

[10] K. Lengwehasatit and A. Ortega. Probabilistic partial-distance fast matching algorithms for motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 11(2):139–152, Feb. 2001.

[11] T. Mudge. Power: A first class design constraint. *IEEE Computer*, 34(4):52–57, April 2001.

[12] P. Pirsch, N. Demassieux, and W. Gehrke. VLSI architectures for video compression-a survey. In *Proc. IEEE*, volume 83(2), pages 220–246, Feb. 1995.

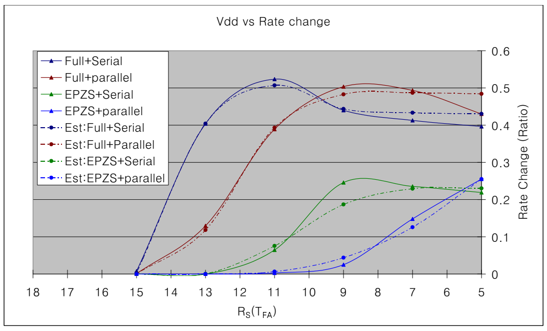

Fig. 3. Rate change due to DVS in ratio compared to original rate (dot: estimated data using our model, solid: real data); using a FOREMAN sequence, QP=20

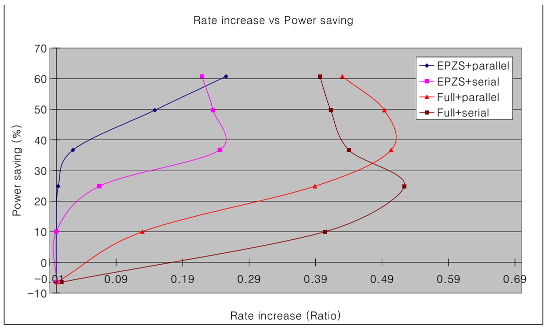

Fig. 4. Power saving effect of ME process using DVS for various ME algorithms and MMC architectures; considering redundancy due to encoding one frame of every GOP with Vdd_{crit} and R_S^L for SAD_i information and X_1 estimation, FOREMAN sequence, QP=20, gate parameters ($\alpha = 2.0, Vdd_{crit} = 3.3V, V_t = 0.62V$)