Abstract—We study the impact of a novel prioritized frame dropping scheme in buffer-constrained multiprocessor system-on-chip (MPSoC) platforms. Accurate buffer dimensioning has attracted lot of research interest as large on-chip buffers result in increased silicon area and higher costs. Multimedia applications present the flexibility of trading off quality for buffer space without any noticeable deterioration in video quality. The frame dropping scheme is crucial here to drop frames appropriately such that the required buffer size is reduced and target quality requirement is satisfied. Towards this, we propose a simple prioritized frame dropping mechanism which reduces the required buffer space more than existing frame dropping policies. We also provide a fast iterative procedure to find the minimum buffer size for a video clip with \( O(\log(N_{\text{drop}})) \) number of iterations, where \( N_{\text{drop}} \) is the maximum number of frames that can be dropped for a video clip so that a prespecified quality in terms of peak signal to noise ratio (PSNR) value is satisfied.

I. INTRODUCTION

A. Motivation

Video decoders require significant amount of on-chip buffer resources in order to store the incoming/partially processed frames. A large on-chip buffer size increases the cost of the device running the video decoder. This is because large on-chip buffers are one of the major reasons for increase in the chip area ([1] and [9]) and the power consumed ([3] and [10]). Lowering power consumption is becoming increasingly important, especially in mobile devices, where extended battery life is one of the main design targets. Therefore, accurate buffer dimensioning in multimedia MPSoC platforms has attracted lot of research attention. All prior works in buffer sizing ([4] and [7]) discounted the idea of frame losses in favor of maximum output quality. There have also been works on frame dropping policies ([2] and [8]) to maximize output quality in the presence of scarce buffer resources. However, there has been no work on quality driven buffer dimensioning using efficient frame dropping strategies such that the required buffer size is reduced while satisfying a target output quality. This work can be appropriately used for multimedia decoders running on MPSoC platforms as these decoders can tolerate some quality loss without significant deterioration in video perception.

In this paper, we first study the effect of existing quality aware frame dropping policy on the required buffer size such that a target PSNR value is achieved. Here, we study a frame dropping policy ([2]) which prioritizes frames based on their frame sizes (in bits) i.e., frames are dropped in the increasing order of their frame sizes. This dropping policy is dynamic as it can be implemented online when the frames arrive on the MPSoC architecture. In contrast, though [8] provides an optimal frame dropping policy (as it drops frames in increasing order of distortion caused), it cannot be implemented online as it requires the complete decoding of the video stream. We conducted simulations to derive the buffer savings obtained by dropping frames using the frame drop policy from [2]. The results for the benchmark MPEG-2 video \textit{susi\_080} ([15]) with target PSNR values 35 dB and 40 dB are presented in Fig. 1. The buffer savings obtained are compared with the optimal frame dropping policy. The difference in buffer savings are shown as \( B_{35_{\text{diff}}} \) and \( B_{40_{\text{diff}}} \) for PSNR values 35 dB and 40 dB, respectively. It is observed that these differences are considerably large values. The buffer savings using frame drop policy in [2] correspond to a drop of 181 and 81 frames respectively, for PSNR values of 35 and 40 dB. These figures for the optimal framedrop policy are 242 and 147 frames. In order to reduce the \( B_{35_{\text{diff}}} \) and \( B_{40_{\text{diff}}} \) values, we propose a simple prioritized frame dropping policy based on motion vectors, thereby enabling a buffer savings closer to the optimal. The motion vector based dropping policy can be implemented online. The on-chip frame buffer sizes, without frame drops, are of the order of 100 Mb ([7]). Therefore, in this context our savings are quite significant.

![Fig. 1. Evaluation of buffer savings using frame dropping policy from [2] versus optimal frame dropping policy from [8] for a benchmark MPEG-2 video \textit{susi\_080} ([15]).](image-url)
Our proposed motion vector based frame dropping policy paves the way for efficiently reducing buffer size when the required output quality is known. However, determination of the minimum buffer size is a time consuming process because it requires many system simulations with various buffer sizes. Therefore, we further propose an efficient iterative strategy to derive the appropriate minimum buffer size for a given video stream. A system designer can perform this exercise with a representative set of video clips in the library (covering all possible characteristics). This is the current practice for evaluating architectures (which is similar to testing software for functional correctness using a representative test case suite). The buffer sizes obtained for individual clips can be used to decide the final buffer size requirement such that any encoded clip adhering to the bounds on input data bursts, exhibited by the library, will be decoded to achieve the required output quality.

B. Contributions

This is the first work where the effect of existing frame drop policies is studied in the context of buffer sizing when a loss in quality of a decoded video is accepted. The results obtained highlight the need of a new quality-aware frame dropping policy that can provide buffer savings closer to what the optimal frame dropping policy provides. Therefore, firstly we propose a motion vector based frame dropping policy which bridges the difference in buffer savings observed in Fig. 1 thereby enabling higher buffer savings than existing quality-aware frame dropping policy ([2]).

Secondly, given the frame dropping policy, the MPSoC architecture parameters (number of processing elements (PEs) and PE frequencies) and task mappings on PEs, we provide a fast iterative method to derive the appropriate minimum buffer size required so that a target quality constraint is satisfied. This significantly reduces the number of iterations required to find the appropriate minimum buffer size to $O(\log(N_{drop}))$, where $N_{drop}$ is the maximum number of frames that can be dropped for a video clip. This does not depend on the length of the video clip if it adheres to the bounds on quality variation and input data bursts exhibited by the input library. We provide experimental results using MPEG-2 decoder as the case study application, but the concepts discussed here are relevant in the case of other multimedia decoders also. We obtain a buffer savings of 3.9 Mbits for a video library with target PSNR = 30 dB even when the buffer is provisioned for the worst-case clip.

C. Related Work

On-chip buffers take up a lot of chip silicon area. This is evident from [1], in which experiments clearly show the enormous amounts of silicon area increase due to the increase in FIFO size in the router. In [9], this same concern is demonstrated in the context of on-chip network design for multimedia applications. However, the authors do not drop any incoming packet from the buffer thereby giving importance to maximum application quality. There are various objective functions that are considered while choosing the appropriate buffer size. A buffer allocation strategy is proposed in [1] in order to increase the overall performance in the context of a networks-on-chip router design.

Buffer dimensioning is an important aspect of designing media players. In the past, there has been lot of work in this area where several design factors have been taken into consideration while choosing the appropriate buffer size. All these works did not tolerate any loss as a result of buffer overflow and underflow ([4] and [7]). Hence, none of these works considered the trade off between buffer and video quality allowing some buffer overflows (i.e. with constrained buffer). Here, video quality is not the end-to-end QoS, but the distortion in the received frames.

II. OVERVIEW OF THE FRAMEWORK

The problem is formally defined here before getting into the components of the framework used.

A. Problem Formulation

In this paper, we address the problem of buffer dimensioning for MPSoC platforms such that a target quality constraint is satisfied. Limited buffer sizes result in the loss of macroblocks/frames constituting an encoded video, which in turn leads to a drop in the quantitative quality level of the video decoded data received. Hence, given a library of video clips that covers all types of video test cases, it is essential for a system designer to quickly find the minimum buffer size for the required output quantitative quality level, which here is measured in terms of peak signal to noise ratio (PSNR).

The formal definition of the problem is stated as follows:

Given an exhaustive library of video clips $V$ covering a wide variety of video characteristics, the operating frequency of the processing elements (PEs) $f PE_i$, where $1 \leq i \leq N_{PE}$ ($N_{PE}$ is the number of PEs in the MPSoC platform), workload values to execute each task in the video decoding application for all constituent blocks in the multimedia data, the task is to find

$$B_{min} = \max(\min B_j)$$  \hspace{1cm} (1)

subject to

$$\min(psnr_{out}) \geq psnr_{req}$$  \hspace{1cm} (2)

Here, $psnr_{out}$ is the PSNR value received at the output of the MPSoC platform, $psnr_{req}$ is the target PSNR value that needs to be achieved for the library of clips. Finally, $B_{min}$ is the maximum of the minimum buffer sizes required for each video clip $v \in V$, at the input to the $j$-th stage (which as shown in Eqn. 1 is the minimum of all possible buffer values $B_j$) in order to satisfy Eqn. 2. To efficiently solve the above problem, we need a proper quality-aware frame dropping function $FD()$ and a fast iterative strategy to derive the minimum buffer size $B_j$. Hence, we now highlight the importance of a quality-aware frame dropping policy in deriving a minimal buffer size satisfying a target PSNR value.
B. Quality-Aware Frame Dropping

Lemma 2.1: Given the operating frequency of the PEs, the task workload values and the video library, Eqn. 1 can be strictly satisfied only if the frame dropping strategy is aware of the relative importance of the frames (in terms of distortion introduced) in the video stream.

Proof: Let us consider that \( FD1() \) is a random frame dropping function and \( FD2() \) is a frame dropping function aware of the importance of the frames and its contribution to PSNR, if dropped. Given the target PSNR value \( psnr_{req} \) and the maximum buffer size \( Buf \) causing no frame drops for a video clip, let the maximum number of frames dropped by \( FD1() \) before it achieves the PSNR output value of \( psnr_{out} \) satisfying Eqn. 2 be \( n1 \) frames. Let the number of frames dropped by \( FD2() \) for the same case be \( n2 \). As \( FD2() \) is aware of the relative importance of the frames, it will drop frames in the increasing order of how they reduce the PSNR output value i.e. frames which reduce the PSNR output value least will be dropped first, while \( FD1() \) does not do this strictly. Hence, it is easy to conclude that \( n2 > n1 \). If \( Buf1 \) is the minimum buffer size estimated (as in Eqn. 1) with \( FD1() \), \( Buf2 \) is the minimum buffer size estimated with \( FD2() \) and \( FSIZE \) is the frame size, then the following conditions hold: \( Buf1 \leq Buf - n1 \times FSIZE \), \( Buf2 \leq Buf - n2 \times FSIZE \) and hence \( Buf1 \geq Buf2 \). This proves that a good estimate of minimum buffer size (in macroblocks) satisfying the PSNR requirement can be achieved only with a good frame dropping strategy. However, if \( n2 \gg n1 \) (due to quality aware frame dropping of \( FD2() \)), we see that \( Buf1 > Buf2 \) (in bits).

A state-of-the-art frame dropping method [2] discussed in literature prioritizes frames to be dropped based on the frame types namely I-type, P-type or B-type (as in the MPEG-2/MPEG-4 decoder context) with frame size based prioritization within frame types. This strategy works well in comparison to a random frame dropping strategy, but does not take the aspect of motion across frames into consideration. Hence, the PSNR values obtained with the frame size based prioritization strategy does not work particularly well for motion videos when a dropped frame is replaced with the previous processed frame which has considerable movement. Therefore, we adopt a frame dropping strategy whereby motion-vector based frame dropping is employed. The motion-vectors can be easily obtained by parsing the video stream. The output of this stage is the prioritized order in which frames have to be dropped for a video. The gaps created in buffer with such dropping can be consolidated with minimal logic.

C. Determination of \( B_{min} \)

In the second stage of the framework, we estimate the minimum buffer size required to achieve a prespecified PSNR value for a library of video clips using the task workloads and frame drop priorities obtained earlier. This stage employs a fast iterative process in order to determine the minimum buffer size. The main idea here is to maximize the frame drops subject to the condition that the prespecified PSNR value is met. As PSNR is dependent on the mean square error (MSE) of the dropped frames when replaced by the concealment frames, the condition of Eqn. 2 is translated from the PSNR domain into the MSE domain. As MSE is inversely proportional to PSNR, the condition is changed to \( \max(\text{mse}_{out}) \leq \text{mse}_{req} \). Let this be the \textbf{MSE satisfaction criterion}. Here, \( \text{mse}_{out} \) is the MSE value received at the output. The goal of the entire framework is to achieve a \( \text{mse}_{out} \) value equal to the maximum possible MSE value less than or equal to the target MSE value (\( \text{mse}_{req} \)). MSE is preferred over PSNR as it is an easier quantity to work with. However, the problem of finding the minimum buffer size for a library of video clips which satisfies the above mentioned conditions is time consuming if all the possible buffer sizes are tested. We use a faster iterative approach to find \( B_{min} \), which will be discussed in detail later.

III. System Architecture Description

The MPSoC platform architecture used for our case study of the MPEG-2 decoder application consists of multiple interconnected PEs as shown in Fig. 2. The PEs communicate by passing data units or stream objects between them. \( PE_1 \) and \( PE_2 \) are the two programmable processors. In our case study, \( PE_1 \) performs the VLD and IQ tasks, while \( PE_2 \) performs the IDCT and MC tasks. The stream objects on which the PEs operate are macroblocks (MBs). The input encoded MBs are stored in the input buffer of size \( B_1 \). Partially decoded MBs are sent from \( PE_1 \) to \( PE_2 \) through a buffer of size \( B_2 \) while fully processed MBs are sent out of \( PE_2 \) to the output interface through a playout buffer. \( f_{PE_1} \) and \( f_{PE_2} \) are the frequencies of the PEs 1 and 2 respectively. Frames are dropped only at the first stage of the architecture and a size \( B_2 \) will be determined accordingly. This has a ripple effect in the reduction of buffer sizes in the following stages because the number of frames to be stored are less.

Fig. 2. MPSoC platform architecture for case study

IV. Quality-Aware Frame Dropping

In order to improve the quality-awareness in assigning priorities to frames during frame drops, we propose a motion-vector (MV) based prioritization of frames. This is a fast method because the motion vectors of each frame can be extracted rapidly from the encoded bitstream that is received. Moreover, as it takes the motion information into consideration, it takes into account the quality degradation experienced when a frame is dropped. One of the advantages of this prioritization method and the one discussed in [2] are that they are easy to compute. Motion vectors and frame sizes can be extracted from the bitstream quickly when compared to the MSE computation of dropped frames with all possible concealment frames (frames that replace dropped frames).
1) There are two motion vectors in the MPEG-2/MPEG-4 encoded bitstream - one for the upper half of the 16 × 16 macroblock and the other for the lower half. Each motion vector has a forward and a backward component. Each of the above four components have a vertical and a horizontal part. All these quantities are at the MB granularity. We now obtain a consolidated value combining the horizontal and vertical parts for each component. Let the MVs be denoted by \( mv_{l,m,n} \), where \( l \) represents the upper or lower half (takes on the values 0/1 respectively), \( m \) represents the forward/backward component (takes on the values 0/1 respectively) and \( n \) denotes the vertical/horizontal parts (takes on the values 0/1 respectively). The four consolidated values (per MB) combining the horizontal and vertical parts are

\[
\begin{align*}
mv_{0,0} &= \sqrt{(mv_{0,0,0})^2 + (mv_{0,0,1})^2} \\
mv_{1,0} &= \sqrt{(mv_{1,0,0})^2 + (mv_{1,0,1})^2}
\end{align*}
\]

The consolidated motion vector components for each frame is computed as summation of MB level motion vector components as shown below

\[
MV_{l,m} = \sum_{mbno=1}^{FSIZE} mv_{l,m,\{0,1\},m \in \{0,1\}}
\]

where \( mbno \) is the MB number.

2) After obtaining the motion vectors per frame given by Eqn. 4, we find the priority of dropping the frames. As in any efficient frame dropping strategy, we drop B-frames first followed by P-frames and finally I-frames in the case of a buffer overflow. We implement this by creating three separate lists of priorities, one each for B-frame, P-frame and I-frame denoted by \( b\_priority \), \( p\_priority \) and \( i\_priority \). These lists contain the indexes of the frames arranged in the order of increasing motion vector values.

3) \( b\_priority \) holds the B-frame priority list. There are two types of B-frames - odd numbered B-frames and the even numbered B-frames. The even numbered B-frames are always dropped first when possible as they can then be replaced by the immediate odd numbered B-frame before it in the temporal sequence. Once all the possible even numbered B-frames are dropped, the odd numbered B-frames are discarded. Among the odd numbered B-frames, they are prioritized based on the increasing order of the sum of forward components i.e. the comparison metric is \( MV_{0,0} + MV_{1,0} \). The comparison metric for the priority computation of even numbered B-frames is \( MV_{0,0} + MV_{1,0} + MV_{0,prev} + MV_{1,prev} \). If the comparison metric of a frame is high, it will be dropped later. \( MV_{prev} \) are the motion vector components for the previous frame. For the even numbered B-frames, we need to consider the backward motion components of the previous odd numbered B-frame.

4) \( p\_priority \) holds the P-frame priority list. Here it is desirable that all the P-frames in a GOP are kept together in the list, the last P-frame to be dropped being the one closest to the I-frame as it acts as a reference frame for a lot of following frames. Among the group of P-frames, the order is decided by the comparison metric \( \sum \{MV_{0,0} + MV_{1,0}\} \), where the summation is over the number of P-frames in the GOP. These lists contain the indexes of the frames arranged in the order of increasing motion vector values.

5) \( i\_priority \) holds the I-frame priority list. It orders the I-frames based on the comparison metric used for P-frames.

We extracted the MSE information of every frame with its reference frame in some motion and still videos along with their motion vectors with respect to the reference frame and the frame sizes. The plots are shown in Fig. 3. From the plot for the motion video \( susi_080 \), we can see that the motion vector emulates the MSE behavior much better than framesize. This behavior has also been observed in MPEG-4 videos, however we do not present it here due to space constraints. We conclude that it is desirable to use a quality-aware frame dropping mechanism with MV-based prioritization of frames.

V. MINIMUM BUFFER SIZE ESTIMATION

In this section, we propose an iterative procedure to estimate the minimum buffer size required at each stage of the MPSoC architecture so that a library of video clips satisfies the required target PSNR value. The input to this stage from the previous stage is an order in which frames should be dropped. In order to obtain the minimum buffer size, the best strategy is to drop the maximum number of frames but still satisfy the MSE saturation criterion. A straightforward approach of using brute force method to check all the buffer sizes is not desirable.
Algorithm 1 Computing minimum buffer size for a clip in first stage

Input: \( \text{mse\_stat}() \) - mse values for a threshold number of frames \((B_{th})\);  
Output: \( B_{1}^{\text{min}} \) for a given \( f_{PE} \) and frame discard algorithm  
1: \( N \leftarrow 0, \text{mse\_eval} \leftarrow 0 \) and \( \text{frm\_ind} \leftarrow 0; \)  
2: for \( i = 1 \) to \( B_{th} \) do  
3: \( \text{if} \ \text{mse\_eval} > \text{mse\_req} \) then  
4: \( N_{\text{drop}} = i - 2 \)  
5: break  
6: else  
7: \( \text{frm\_ind} = b_{\text{priority}} \rightarrow \text{next} \)  
8: \( \text{mse\_eval} = \text{mse\_eval} + \text{mse\_stat}(\text{frm\_ind}) \)  
9: end if  
10: end for  
11: \( N_{\text{prev}} \leftarrow 0, N_{\text{int}} \leftarrow 0, \text{mse\_eval} \leftarrow 0, \text{Buf}_1 \leftarrow 0, \)  
12: \( n_{\text{drop}} \leftarrow 0, N_{\text{new}} \leftarrow 0, \text{start} \leftarrow 0, \text{end} \leftarrow N_{\text{drop}} \)  
13: Reset list next pointer to start of list  
14: repeat  
15: \( N_{\text{prev}} = N_{\text{new}}, N_{\text{new}} = N_{\text{drop}} - ((\text{end} - \text{start})/2), \)  
16: \( N_{\text{int}} = 0, \text{mse\_eval} \leftarrow 0, \text{Buf}_1 \leftarrow 0, n_{\text{drop}} \leftarrow 0 \)  
17: Reset list next pointer to start of list  
18: while \( \text{mbin\_no} \leq N_{\text{tot}} \times \text{FSIZE} \) do  
19: if \( \text{Buf}_1 = \text{MAX\_BUFFER} - N_{\text{new}} \times \text{FSIZE} \) then  
20: \( \text{frm\_ind} = b_{\text{priority}} \rightarrow \text{next} \)  
21: \( \text{Buf}_1 = \text{Buf}_1 - \text{FSIZE} \)  
22: \( n_{\text{drop}} = n_{\text{drop}} + 1 \)  
23: \( \text{if} \ \text{mse\_eval} > \text{mse\_req} \) then  
24: \( N_{\text{int}} = n_{\text{drop}} - 1 \)  
25: else  
26: \( \text{if} \ \text{mse\_eval} = \text{mse\_req} \) then  
27: \( N_{\text{int}} = n_{\text{drop}} \)  
28: end if  
29: end if  
30: else  
31: \( \text{Buf}_1 = \text{Buf}_1 + 1 \)  
32: end if  
33: end while  
34: if \( N_{\text{int}} = 0 \) then  
35: \( \text{start} = N_{\text{new}}, \text{end} = \text{end} \)  
36: else  
37: \( \text{start} = \text{start}, \text{end} = N_{\text{int}} \)  
38: end if  
39: until \((\text{start} = \text{end})||((\text{start} = \text{end} - 1))\)  
40: \( B_{1}^{\text{min}} = \text{MAX\_BUFFER} - N_{\text{prev}} \times \text{FSIZE}, \forall v \in V \)  

as it is very time consuming considering the number of clips in the library and the range of buffer sizes used. Therefore, we use an iterative mechanism to select the minimum buffer size.

In order to obtain the minimum overall buffer size considering all stages, the best strategy is to drop all the required frames in the first stage. This will not only reduce the buffer size \( B_1 \), but will also reduce the number of frames that stage 2 and buffer size \( B_2 \) have to handle. In order to start the iterative process of finding the minimum buffer size, we need to find the MSE values for a limited number of frames which are dropped first according to the drop order until the accumulated MSE just exceeds the target MSE value \( \text{mse\_req} \). The PSNR value of a video sequence with frame drops is expressed as

\[
psnr = 10 \times \log_{10} \left( \frac{255 \times 255}{\text{MSE}} \right)
\]

\[
(\text{MSE}, n) = \sum_{n=0}^{W-1} \sum_{h=0}^{H-1} (r(h,w,n) - r_c(h,w,n))^2, \text{ MSE}_r = \sum_{n=0}^{N_{\text{drop}}-1} (\text{MSE}_r, n, c_d) \text{ is the red pixel intensity of the dropped frame and } r_c \text{ is the red pixel intensity of the concealment frame (immediately preceding successfully processed frame).} h, w and n are the height, width and frame drop number indices. Similar explanations hold true for \( \text{MSE}_g \) and \( \text{MSE}_b \). \( W \) and \( H \) are the horizontal and vertical resolution of each frame in the video. \( N_{\text{tot}} \) is the total number of frames in the video sequence and \( N_{\text{drop}} \) is the number of frames dropped in the sequence. Let \( \text{mse} = (\text{MSE} + \text{MSE}_g + \text{MSE}_b) \). Then the target MSE value can be expressed in terms of the target PSNR value \( \text{psnr\_req} \) as \( \text{mse\_req} = \left((\frac{255 \times 255 \times N_{\text{tot}}}{2})\right)^2 \).

The minimum buffer size for the first stage can be calculated using Algorithm 1. The terms used are:

- \( \text{mse\_eval} \) - Cumulative MSE value, \( \text{frm\_ind} \) - Index of the frame dropped, \( \text{mbin\_no} \) - Arriving MB number, \( N_{\text{prev}} \) - Value of the number of frames dropped in the previous iteration, \( \text{MAX\_BUFFER} \) - Buffer size required if there are no frame drops, \( \text{Buf}_1 \) - Buffer size at stage 1 of the architecture, \( N_{\text{new}} \) - Number of frames to be dropped (in Step 2), which is computed every iteration until we achieve the maximum value which does not exceed \( \text{mse\_eval} \) and finally \( \text{start} \) and \( \text{end} \) variables are used to speed up the search of the possible number of frame drops. In this algorithm, there are two steps.  

**Step 1 (Lines 1-9):** Here, we find the maximum number of frame drops \( N_{\text{drop}} \) possible given the priority of frame drops and the \( \text{mse\_req} \) value. It is assigned as \( i = 2 \) because \( \text{mse\_eval} > \text{mse\_req} \) for \((i-1)\)-th frame drop. We only consider the B-frame drops here because this itself leads to a large PSNR drop covering the PSNR ranges we are exploring and hence the \( B_{th} \) (line 3) is equivalent to the number of B-frames.

**Step 2 (Lines 10-39):** Once \( N_{\text{drop}} \) is computed, we can compute \( B_{1}^{\text{min}} \) as shown in line 39 of Algorithm 1. The minimum buffer size \( B_{1}^{\text{min}} \) will be due to frame drops less than or equal to this maximum number \( N_{\text{drop}} \) computed in (line 4) Step 1. It can be less than \( N_{\text{drop}} \) if certain frames, with higher priority to be dropped, are already processed and sent to the next stage. This target number of buffer drops will be computed iteratively. Once the minimum buffer size is computed for all the clips, the resultant buffer size of stage 1 can be computed as \( B_{1}^{\text{min}} = \max_{\forall v \in V} (B_{1}^{\text{min}}) \). The iterative procedure can be run using \( O(\log(N_{\text{drop}})) \) iterations, where \( N_{\text{drop}} << N_{\text{tot}} \) and \( N_{\text{drop}} \) is independent of the length of the clip, but dependent on nature of the clip and required PSNR.
value. In our example, the maximum number of iterations for \( PSNR = 40\, dB \) was 7, which is low.

VI. EXPERIMENTAL RESULTS

In this section, we conduct two sets of experiments to validate our proposals earlier. The first experiment involves verifying the effect of MV-based frame dropping on buffer size reduction. The second result gives us the minimum buffer size values for various PSNR values at the first stage of the MPSoC architecture for the library of video clips we have used here. We use 11 video clips from [5] - 5 still clips and 6 motion clips. It has been observed that the still clips require lesser buffer size. Hence, we show the buffer sizes of motion clips only (as they decide minimum buffer size required) in both experiments. The MPEG-2 decoder source code used was from [6]. In a shared buffer scenario with multiple cores, the reduction of required buffer size helps in allocating the free buffer resources to other cores requiring more space.

A. Evaluation of MV-based frame dropping

![Comparison of buffer savings for susi_080](image1)

![Comparison of buffer savings for tens_080](image2)

Fig. 4. Comparison of buffer savings for susi_080

Fig. 5. Comparison of buffer savings for tens_080

It is clear from Fig. 4 and Fig. 5 that MV-based prioritized frame drops help in improving buffer savings when compared to the framesize based frame drops. As shown in Fig. 4, we achieve 22.89% and 55% more buffer savings over framesize based dropping for PSNR values of 35 dB and 40 dB respectively in susi_080. We also observe from Fig. 5 that the additional savings (over framesize based drop from [2]) in tens_080 is 2.87 times and 5.29 times for PSNR values of 30 dB and 35 dB respectively.

B. Minimum Buffer Size Estimation

We conducted experiments to find the minimum buffer size required at the first stage of the MPSoC architecture shown in Fig. 2 using the video clips from [5]. The prespecified PSNR values for which we estimated the minimum buffer values are shown in Table I. Due to space constraints, we show the buffer sizes for only 3 motion videos which required highest buffer sizes. For the entire library, with a target PSNR value of 30 dB, we achieved a buffer savings of 3.9 Mbits (Maximum buffer size for the entire library without drops - Maximum buffer size for PSNR = 30 dB (required for flwr_080)). The \( \times \) symbol against the clip susi_080 indicates that the quality never drops to 30 dB even if all the B frames are dropped. Hence, the iterative process can be immediately terminated because the resultant buffer size will be lesser compared to other video clips.

VII. CONCLUDING REMARKS

In this work, we study the effect of frame drops in buffer dimensioning for multimedia MPSoC platforms. We propose a quality-aware framed dropping scheme based on motion vectors that reduces the buffer size required for a prespecified quality constraint. Further, we also propose a fast iterative strategy to derive the minimum buffer size required for a target quality output. In future, we would like to extend this work to find appropriate scheduling parameters for a library of clips so that a PSNR value is satisfied under resource constraints.

ACKNOWLEDGMENT

We would like to acknowledge that this research was carried out at the Centre of Social Media Innovations for Communities (COSMIC), sponsored and supported by the Singapore National Research Foundation and Interactive & Digital Media Program Office, MDA.

REFERENCES


