Power Aware H.264/AVC Video Player on PAC Dual-Core SoC Platform

Jia-Ming Chen[1], Chih-Hao Chang[2], Shau-Yin Tseng[2],
Jenq-Kuen Lee[1], and Wei-Kuan Shih[1]

[1] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
jonathan@rtlab.cs.nthu.edu.tw, {jklee, wshih}@cs.nthu.edu.tw
[2] SoC Integration Division of STC, ITRI, Hsinchu, Taiwan
{chchang, tseng}@itri.org.tw

Abstract. This paper proposes a novel power-aware scheme of H.264/AVC video player for the PAC SoC platform based on its modern dual-core architecture with DVFS capability. Energy/power is saved by the global view of power state transitions on the dual-core subsystem according to a user's behaviors of playing a video. When the user stays in continuous video decoding, a fine-grain power-aware scheme is devised to save the energy/power in advance. Especially the fine-grain model is suitable for any standard coded H.264/AVC video without extra modifications. We also discuss a workable reduction technique when imprecise video decoding time is permitted under soft real-time constraint. For a similar SoC platform with the dual-core architecture and DVFS capability, the idea presented here is, to the best of our knowledge, the first power-aware design of H.264/AVC video player.

Keywords: Power-aware, Dual-Core SoC, DVFS, H.264/AVC

1 Introduction

Multimedia applications on mobile/portable devices, such as PDAs, smart phones, and Portable Media Players (PMPs) become more and more popular nowadays. Due to these portable devices are battery-operated, one imperative objective is conserving energy consumption to lengthen the battery service time. Especially within these multimedia applications, video services play an important role such as video conferences, video phones, digital TV broadcasting, and DVD players. However, processing video data requires a huge amount of computation and energy. Based on the state-of-the-art video coding standard, H.264/AVC [1, 2], many researches [3, 4, 5] revealed that there is a great variances in computational complexity between sub-procedures during video decoding, which is mainly induced by different frame types and variation of moving objects among scenes. This characteristic makes further investigation of reducing energy consumption during video decoding possible.

Traditional low power techniques, such as clock gating, power gating and dynamic voltage and frequency scaling (DVFS) have been commonly used in modern hardware and software designs, and have been proved to be a powerful and practical solution on power/energy reductions. The main idea of these techniques is to provide just enough computation power without degrade the system performance while seeking possible for minimizing power/energy dissipation. Thus, many researchers have applied DVFS technique to reduce power for video decoding. Based on prediction models for the decoding time of oncoming frames, the processor can be set to the proper voltage and frequency to minimize the power consumption without violating the quality of video streaming. These studies have been compared and summarized in [6]. Similarly, [7] has further investigated the prediction through decomposition of video workloads together with a methodology, called "inter-frame compensation" to reduce the predication errors. In addition, since prediction methods essentially evoke errors, and are harmful to video decoding with hard real-time constraints, [8] proposed a power-aware video decoding scheme according to information of non-zero coded macroblocks, where these information need to be recoded in video streams.

In general, most researches of power-aware video decoding by DVFS techniques only concentrated on a single processor. However, in recent trends of hardware design for portable devices, system designers [9, 10, 11] often integrate a RISC processor with a coprocessor (DSP or ASIC) into one SoC chip by taking advantages of modern VLSI design processes in order to achieve higher cost-effective hardware solutions. The primary reason for this design is that customers require more complex functionalities on smaller portable devices. At the same time, while concerning saving more energy/power, built-in a DVFS technology into a dual-core SoC platform is a promising solution. For example, Texas Instrument Inc. will provide the SmartReflex™ technology on their next generation OMAP™ 2 architecture [12]. Likewise, the PAC (Parallel Architecture Core) SoC platform, which is developing in an on-going project held by STC/ITRI organization at Taiwan [13], provides another comparable solution.

Unfortunately, in regard to these modern SoC platforms with dual-core architecture, previous researches cannot simply be adopted to efficiently solve the power-saving problems for video decoding. Therefore, in this paper, we first propose a valuable power-aware scheme of H.264/AVC video player, and demonstrate how to apply this technology onto the PAC SoC platform. In fact, the principle of the proposed methodology can be also applied to similar platforms (e.g., a dual-core platform with DVFS capability) without or with minor modifications.

The remainder of this paper is organized as follows: The architecture of PAC dual-core SoC platform with DVFS capability is introduced in Section 2. Then based on this platform, a coarse-grain power-aware scheme for H.264/AVC video player is presented in Section 3 and a fine-grain power-aware scheme for continuous video decoding process is devised in Section 4. After that, in Section 5, we proposed a workable reduction techniques based on the devised scheme in Section 4 when imprecise decoding time are admitted under soft real-time constraint and some experiments are given. Finally, conclusions are made in Section 6.

2 The PAC SoC Platform

Before proposing the power-aware scheme of H.264/AVC video player, we briefly introduce the DVFS capability of the PAC SoC platform in this section. Detail information can be found in [13]. The core of PAC SoC platform is divided into eight power domains: MPU, DSP-Logic, DSP-Memory, on-chip memory, AHB modules, APB modules, analog modules (e.g., PLLs), and others (with fixed voltage), where DSP, MPU, and AHB modules have DVFS capabilities, which can be triggered by the DVFS controller. According to the DVFS capabilities of MPU and DSP, their operation modes can be further classified into *active*, *inactive*, *pending* and *sleep* as shown in Table 1.

Thus from the viewpoint of the dual-core subsystem, the interactions of global power states between MPU and DSP can be illustrated in Fig. 1(a). For example, when some functions are implemented on DSP and is activated right away, the power state may be transited from Active state 3 (i.e., MPU is in active and DSP is in sleep) to Active state 1 (i.e., both MPU and DSP are in active). Furthermore, in the Active state 1, MPU and DSP have different voltage and frequency settings (as shown in Table 1) for their power-saving purposes. Along with taking advantage of this feature, we can devise our power-aware scheme for H.264/AVC video player, explained in the next section.

Table 1. Power and action states of MPU and DSP

Power mode	Operation condition		Power consumption(II)	Computation power(I)	Transition latency(us)
	Freq.(MHz)	Voltage(V)			
MPU_active-1	228	1.2	100%	1	1
MPU_active-2	152	1.2	76%	2	1
MPU_active-3	114	1.2	65%	3	1
MPU_inactive	0	1.2	<30%	None	1
MPU_sleep	0	0	<1%	None	120
DSP_active-1	228	1.2	100%	1	120
DSP_active-2	152	1.0	60%	2	120
DSP_active-3	114	0.8	50%	3	120
DSP_inactive	0	1.2	<30%	None	1
DSP_pending	0	0.8	<30%	None	120
DSP_sleep	0	0	<1%	None	120

(I): 1>2>3; (II): theoretical value

3 Power-Aware H.264/AVC Video Player

Fig. 1(b) displays the behaviors mapping onto Fig. 1(a) when a typical H.264/AVC video player is running on the PAC SoC platform. Once the video player changes its behavior, the dual-core subsystem may switch its power states. For example, while the video player changes its behavior from “Play” to “Pause within 15 min”, the dual-core subsystem switches its power state from “Active state 1” to “Active sate 2”,

in which MPU keeps in active mode, but DSP switches from active mode into inactive mode. At this moment, the frequency of DSP is turned off by the DVFS controller to save the dynamic power dissipation. As a result, through this mapping policy H.264/AVC video player can indicate the DVFS controller to dynamically adjust the supply voltage/frequency of dual-core subsystem in order to conserve the energy of the whole system.

In experience, as soon as H.264/AVC video player enters "Play" behavior, it would keep quite a long time staying in that behavior due to users' habits. Many researches [14, 15] have been presented to convince that power consumption is reduced by about 30%~40% during continuously video decoding in a single processor. Likewise, when dual-core subsystem stays in "Active state 1" to proceed H.264/AVC video decoding, fine-grain frame-based voltage/frequency adjustment for DSP can further save energy consumption (i.e., switching between 3 active modes, 228MHz, 152MHz and 114MHz for DSP depending on the required computation power). Therefore, in the following sections, we explain how to achieve this purpose to complete our power-aware scheme for H.264/AVC video player on PAC SoC platform.

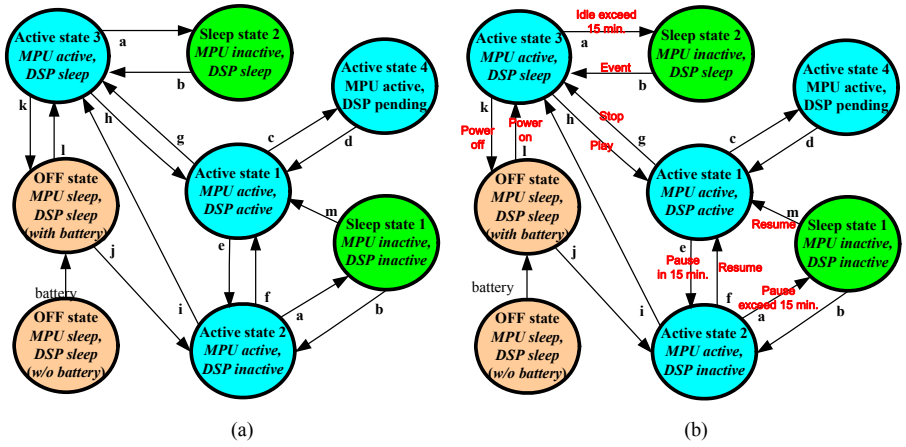

Fig. 1. (a) States of power transition for dual-core subsystem; (b) Mapping a typical video player into states of power transition for dual-core subsystem

4 Power-Aware H.264/AVC Video Decoding

First, we describe how to achieve the fine-grain power-aware H.264/AVC decoding by introducing a partition scheme of H.264/AVC decoding algorithm based on the dual-core architecture. Then, we inlay the power-aware video decoding technique into the system via DVFS capability.

4.1 Partitioning and mapping scheme

As shown in Fig. 2(a), the decoding flow of H.264/AVC is classified into four main procedures: Entropy Decoding (ED), Inverse Quantization/Inverse Transformation (IQ/IT), Predictive Pixel Compensation (PPC), and Deblocking Filter (DF). First, the ED procedure decodes and reorders the compressed bitstream from the NAL to produce a set of quantized DCT coefficients X. Second, the IQ/IT procedure scales and inverse transforms X to D'_n (residual data). Third, the PPC procedure creates a prediction block PRED via Motion Compensation (MC) using reference data F'_{n-1} or Intra prediction where the decision is made by the header information decoded from the bitstream. Finally, PRED is added to D'_n to produce uF'_n which is filtered to produce decoded block F'_n by the DF procedure.

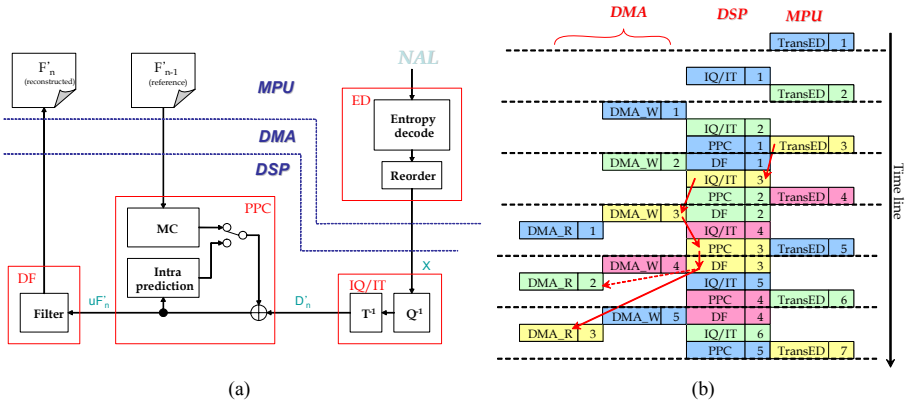

Fig. 2. (a) H.264 decoding algorithm and partitioning; (b) Parallel execution of decoding a picture via MB-by-MB basis

To map the decoding procedure onto PAC SoC platform, we partitioned the decoding procedures into two parts: the **MPU_Part** and the **DSP_Part**. The **MPU_Part** includes the ED and Reference Picture Management (RPM) procedures while the **DSP_Part** includes the IQ/IT, PPC, and DF procedures. There are two primary reasons for MPU to execute the ED and RPM procedures instead of DSP. First, on the one hand the innate behavior of the ED procedure performs back and forth between bit extraction and table-look-up operations, and on the other, the RPM procedure is I/O-intensive as compared with the IQ/IT, PPC, and DF procedures. Both procedures are superior to executing on MPU (large memory) rather than on DSP (less memory). Second, the information gathered during executing the ED procedure on a picture benefits the calculations of computation power for DSP so that it is better to keep the ED and RPM procedures executed on MPU and the other procedures on DSP.

4.2 Execution flow of decoding process with power-aware technique

Based on the partition scheme described in Section 4.1, we propose the decoding flow of H.264/AVC with power-aware technique as explained in Fig. 3. Note that in ***Step3***, the data transfer between MPU and DSP are parallel executed in pipeline concept using the TransED, DMA_Ref, and DMA_Out procedures as described in Table 2. Fig. 2(b) displays the executing flow, where every MB is processed in TransED, IQ/IT, DMA_Ref, PPC, DF, and DMA_Out order (as depicted in Fig. 2(b) by arrow solid lines). Consecutive MBs are also pipelining executed. For instance, when executing the TransED on MB 4, the DF on MB 1, the PPC and DMA_Ref on MB 2, and the IQ/IT on MB 3 are executed in parallel. At this moment, MPU only executes the TransED so that it can utilize the leisure time to execute the ED procedure on the next picture.

Step1 MPU executes the ED procedure to extract quantized coefficients, MVs of MBs and other parameters for the current picture, called **CurPic**. Then those data is stored on a buffer, called **EDBuf_1**.

Step2 MPU adjusts frequency and voltage for DSP including two substeps:

- ***Step2.1*** MPU calculates the required computation power of the remaining decoding procedures (IQ/IT, PPC and DF) for **CurPic** based on information provided from **EDBuf_1**; (Discuss later in Section 4.3);
- ***Step2.2*** MPU decides and adjusts the most moderate frequency/voltage for DSP to decode **CurPic** in accord with result from ***Step2.1***;

Step3 IQ/IT, PPC and DF procedures are executed on **CurPic** with macroblock basis in raster order by MPU, DMA, and DSP cooperatively. Meanwhile, MPU executes the ED procedure on the next picture, called **NextPic,** and produces the entropy decoded data into a buffer, called **EDBuf_2**.

Step4 After DSP has completely decoded **CurPic** and MPU has already prepared **EDBuf_2** for **NextPic**, Step2~3 are applied to **EDBuf_2** repeated, and the entropy decoded data of **NextPic** is restored in the **EDBuf_1.**

Fig. 3. Decoding flow of H.264/AVC with power-aware technique on PAC SoC platform

Table 2. Procedures of data transfer between DSP and MPU.

Procedure	Description
TransED	MPU transfers quantized coefficients data of one MB from entropy decoded buffer (**EDBuf_1**) to DSP memory.
DMA_Ref	DMA transfers required reference data (gathered from reconstructed inter or intra pictures) and its related parameters (e.g., luma CBP, MB type information, MVs…) when DSP executes PPC and DF procedures for one MB.
DMA_Out	DMA transfers reconstructed MB (finished by DSP) to MPU memory.

Next, we explain how to calculate the required computation power for ***Step2.1*** and derive the equation in order to apply DVFS technique to DSP.

4.3 Determination of clock frequency and supply voltage for DSP

Generally, in the proposed scheme as stated in Section 4.2, the calculation of the proper supply frequency f_{dsp} (in MHz) for DSP in ***Step2*** can be formulated as Eq. (1), where $T_{IQ/IT}$, T_{PPC}, and T_{DF} are decoding time (in clock cycles per picture) spent by IQ/IT, PPC, and DF procedures respectively. Besides, since DSP and MPU cooperate to decode the picture in parallel fashion and MBs of a picture must follow the execution order as described in ***Step3*** of Section 4.2, we introduce a control flow procedure in DSP to handle this matter, which spends T_{Ctrl} time (in clock cycles per picture). Furthermore, the parameter $\mathbf{FrameRate_{video}}$ (in fps) is the frame rate provided by the video stream (for example, 30fps in real-time). It is a conservative estimation because the ED procedure for decoding a picture is slack-stealing and is executed by MPU instead of by DSP (as described in ***Step3*** of Section 4.2). Particularly note that during decoding a picture, the overhead of transferred procedures (i.e., DMA_Out, DMA_Ref, and TransED) are not counted here since they are hidden from overlapped parallel fashion as depicted in Fig. 2(b). In Eq. (1), we omit the overhead of executing the TransED procedure on the first MB because it can be compensated by the conservative estimation of aforementioned overlapping executing procedure between MPU and DSP.

$$f_{dsp} = (T_{IQ/IT} + T_{PPC} + T_{DF} + T_{Ctrl}) \times \mathbf{FrameRate_{video}} \times 10^{-6} \quad (\mathbf{1})$$

Previous research [16] proved that apparently different computation power is required when decoding distinct type of pictures (e.g., I/B/P/SI/SP-types). It implies that IQ/IT, PPC, and DF procedures for decoding each picture may consume different power. Here, in order to simplify but not limited to our work, we merely focus on the baseline profile of H.264/AVC decoding, which only contains I-type and P-type pictures. Thus, we can further distinguish Eq. (1) into two cases as follows:

Case1. Decoding an I-type picture

According to H.264/AVC standard, an I-type picture only contains intra-coded MBs such that only three modes, intra 4x4 mode (i.e., a luma MB with 4x4 intra prediction), intra 16x16 mode (i.e., a luma MB with 16x16 intra prediction) and intra 8x8 mode (i.e., a chroma MB with 8x8 intra prediction), are supported. Based on the values of coded block pattern (CBP) for each MB which is known during executing the ED procedure, we can further decompose Eq. (1) into Eq. (2).

$$f_{dsp} = (T^{I}_{IQ/IT} + T^{I}_{intra} + T^{I}_{DF} + T_{Ctrl}) \times \mathbf{FrameRate_{video}} \times 10^{-6} \quad (\mathbf{2})$$

, where

$$T^{I}_{IQ/IT} = N^{I}_{4x4CBP} \times T_{4x4IQ/IT} + N^{I}_{16x16CBP} \times (17 \times T_{4x4IQ/IT}) + N^{I}_{ChrCBP} \times (8 \times T_{4x4IQ/IT} + 2 \times T_{2x2IQ/IT}) \quad (\mathbf{2.1})$$

$$T^{I}_{intra} = N^{I}_{intra4} \times (16 \times T_{4x4PRED}) + N^{I}_{intra16} \times (16 \times T_{4x4PRED}) + N^{I}_{intraChr} \times (4 \times T_{4x4PRED}) \quad (\mathbf{2.2})$$

$$T^{I}_{DF} = N_{bS3} \times T_{bS3} + N_{bS4} \times T_{bS4} \quad (\mathbf{2.3})$$

Eq. (2.1) represents the IQ/IT procedure applying on a picture containing intra 4x4 mode, intra 16x16 mode and intra 8x8 mode MBs (including chroma blue and chroma red). We implement a subroutine of IQ/IT operation on a 4x4 block since IQ/IT operations on 4x4 AC or 4x4 DC blocks (no matter luma or chroma MBs) have the same cycle counts. The cycle counts spent in intra 16x16 mode which contain 16 4x4 AC blocks and 1 4x4 DC block can be summarized to 17 times of $T_{4x4IQ/IT}$. Similarly, cycle counts spent in intra 8x8 mode can be summarized to 8 times of $T_{4x4IQ/IT}$ plus 2 times of $T_{2x2IQ/IT}$ for 8 4x4 AC blocks and 2 2x2 DC blocks respectively. Especially note that N^{I}_{4x4CBP}, $N^{I}_{16x16CBP}$, and N^{I}_{ChrCBP} parameters for each prediction mode only count the frequencies for nonzero coefficient blocks which are known from corresponding CBP values (i.e., CBP=1 if a block contains nonzero coefficients) during executing ED procedure.

In the same way, we consider the cycle counts spent in intra prediction mode for the PPC procedure which is conducted by Eq. (2.2). Parameters N^{I}_{intra4}, $N^{I}_{intra16}$, and $N^{I}_{intraChr}$ stand for the occurrences of three distinct modes in an I-type picture respectively. Moreover, we implement each subroutine for various prediction types (e.g., *DC*, *vertical*, *horizontal*, *planar* and etc.) on a 4x4 block basis. In the H.264/AVC standard, there are 9 types in intra 4x4 mode, 4 types in intra 16x16 mode for luma, and 4 types in intra 8x8 modes for chroma. In order to meet the precisely timing requirement for video decoding, we choose the maximum one as the parameter $T_{4x4PRED}$ (i.e., the *planar* type in our implementation). The values 16, 16, and 4 represent the frequencies of calling subroutines for intra 4x4, intra 16x16 and intra 8x8 modes respectively.

Finally, within an I-type picture, only bS (boundary strength) value equals 3 or 4 occurs in the DF procedure. Therefore, two subroutines (for bS=3, and bS=4) are applied and N_{bS3} and N_{bS4} stand for their occurrence respectively.

Case2. Decoding a P-type picture

Relatively, with the information of CBPs, MB types, and MVs extracted from the ED procedure, Eq. (1) can be decomposed into Eq. (3).

$$f_{dsp} = (T^{P}_{IQ/IT} + T^{P}_{intra} + T^{P}_{inter} + T^{P}_{DF} + T_{Ctrl}) \times \mathbf{FrameRate_{Video}} \times 10^{-6} \quad (\mathbf{3})$$

, where

$$T^{P}_{IQ/IT} = N^{P}_{4x4CBP} \times T_{4x4IQ/IT} + N^{P}_{16x16CBP} \times (17 \times T_{4x4IQ/IT}) + N^{P}_{ChrCBP} \times (8 \times T_{4x4IQ/IT} + 2 \times T_{2x2IQ/IT}) \quad (\mathbf{3.1})$$

$$T^{P}_{intra} = N^{P}_{intra4} \times (16 \times T_{4x4PRED}) + N^{P}_{intra16} \times (16 \times T_{4x4PRED}) + N^{P}_{intraChr} \times (4 \times T_{4x4PRED}) \quad (\mathbf{3.2})$$

$$T^{P}_{inter} = \sum_{i=0}^{8} \sum_{j=0}^{7} N_{inter(i,j)} \times T_{inter(i,j)} + \sum_{i=0}^{8} \sum_{j=0}^{7} N_{interChr(i,j)} \times T_{interChr(i,j)} \quad (\mathbf{3.3})$$

$$T^{P}_{DF} = N_{bS1} \times T_{bS1} + ... + N_{bS4} \times T_{bS4} \quad (\mathbf{3.4})$$

Eq. (3.1) and Eq. (3.2) have the similar meanings as Eq. (2.1) and Eq. (2.2) in **Case1** except for the various execution frequencies of each subroutine. Moreover, the PPC procedure of **Case2** introduces additional motion compensated (MC) inter prediction mode as formulated in Eq. (3.3). It can be divided into 72 subroutines by applying 9 variances of interpolation operations (according to distinct MVs) to 8 distinct MB partitions (including the skip mode) by our implementation in DSP codes. Finally, during the DF procedure, we need to take bS = 1, 2, 3, and 4 into consideration due to the coexistences of inter and intra-coded MBs in a P-type picture. The definitions of symbols in each equation are summarized in Table 3.

Table 3. Definition of subroutines and cycle counts used in Eq. (2) and Eq. (3)

Symbol	Meaning
$T^{I}_{IQ/IT}$, $T^{P}_{IQ/IT}$	Total cycle counts spent in IQ/IT procedures for I-type and P-type picture respectively
$T_{4x4IQ/IT}$	Cycle counts spent in IQ/IT operation for a 4x4 block
$T_{2x2IQ/IT}$	Cycle counts spent in IQ/IT operation for a 2x2 block
T^{I}_{Intra} , T^{P}_{Intra}	Total cycle counts spent in intra-coded MBs (including luma and chroma) for I-type and P-type pictures respectively
$T_{4x4PRED}$	Cycle counts spent in the maximum prediction type operation for a 4x4 block (i.e., the planar type)
T^{P}_{inter}	Total cycle counts spent in inter-coded MBs for P-type pictures
$T_{inter(I,j)}$	Cycle counts spent in interpolation on each inter-coded luma MB or subMB, there are totally 72 cases (9x8) where i=0,..,8 indicates 9 variances of interpolation based on MVs (*) j=0,..,7 indicates 16x16,16x8,8x16,8x8,8x4,4x8,4x4 MB partitions and skipped mode
$T_{interChr(i,j)}$	Similar to above $T_{inter(i,j)}$, but proceed on chroma MB or sub-MB
T^{I}_{DF} , T^{P}_{DF}	Total cycle counts spent in DF procedures for I-type and P-type pictures respectively
T_{bS1}, .., T_{bS4}	Cycle counts spent per 4x4 block for bS=1, 2, 3, 4 respectively within DF procedure (luma and chroma MBs are considered together)

(*) {(0,0)}, {(1/2,0)}, {(1/4,0), (3/4,0)}, {(0,1/2)}, {(0,1/4),(0,3/4)}, {(1/4,1/4),(3/4,3/4),(3/4,4/1),(1/4,3/4)}, {(1/4,1/2),(3/4,1/2)}, {(1/2,1/4),(1/2,3/4)}, {(1/2,1/2)} totally 9 variances for i=0,..,8; We can group several MVs together since their cycles are the same in our implementation on DSP

Consequently, during executing the ED procedure on a picture, MPU can calculate the computation power required by DSP through Eq. (2) and Eq. (3). As a result, the proper supply voltage/frequency for DSP to execute the remainder decoding procedures is determined and triggered via DVFS controller according to the action states of DSP as listed in Table 1.

5 Saving Energy for MPU under Soft Real-Time Constraint

The methodology proposed in Section 4.3 exactly seeks for saving energy by estimating the required frequency of DSP under the hard real-time constraint that the frame rate indicated by the parameter $\mathbf{FrameRate_{video}}$ cannot miss. However, by taking a closer look at Eq. (2) and Eq. (3), $T^{I}_{IQ/IT}$, $T^{P}_{IQ/IT}$, and T^{P}_{inter} consider the calculations of all different cases of prediction modes and macroblock types such that they bring MPU significant computation power. It indicates that MPU may stay in the highest frequency (e.g., *MPU_active-1* in Table 1) and consume large power. On the contrary, if the video decoding only requires satisfying soft real-time constraint and the imprecise estimation are allowed, we can simplify several subroutines in Eq. (1) and Eq. (2) to let MPU stay in lower frequency to save its power dissipation. We summarized the techniques in the following paragraphs.

First, the IQ/IT procedure is simplified by estimating $N^{I}_{16x16CBP}$, N^{I}_{ChrCBP}, $N^{P}_{16x16CBP}$, and N^{P}_{ChrCBP} in Eq. (2.1) and (3.1) based on the profiling results as shown in Table 4(a). Eq. (2.1) is reduced to Eq. (2.1.1), where $N^{I}_{16x16,}$ N^{I}_{8x8} are the total amount of MBs in intra 16x16 mode and intra 8x8 mode for luma and chroma, and P_{16x16}, P_{Cb}, P_{Cr} are extracted from Table 4(a), which individually indicate the probabilities of N^{I}_{16x16} and N^{I}_{8x8} with nonzero coded blocks. Eq. (3.1) is similarly reduced to Eq. (3.1.1). Second, the PPC procedure is simplified in two cases: (i) average value is used for $T_{4x4PRED}$ in Eq. (2.2) and Eq. (3.2), instead of using the maximum one, for the purpose of saving more power of DSP, and (ii) the calculations of MVs for various MB partitions in Eq. (3.3) is simplified by replicating the MVs of subMBs to other subMBs through the policies as shown in Fig. 4. For instance, a MB partition with 16 4x4 luma subsamples is treated as having 4 MVs instead of 16 MVs. Finally, the DF procedure is simplified by unifying the T_{bS1}, $T_{bS2,}$ T_{bS3}, and T_{bS4} in Eq. (2.3) and (3.4) into average value such that we only execute the DF procedure on each MB depending on the bS value (i.e., skip the calculation when bS=0).

We have evaluated the aforementioned mechanisms by investigating the imprecise estimation of DSP. In the experiment, all test sequences are baseline profile in *IPPPPIPP...* frame sequences and ±16 search range including 6 video sequences: (i) container – QCIF, 100 frames, (ii) silent – QCIF, 100 frames, (iii) foreman – QCIF, 100 frames, (iv) news – QCIF, 100 frames, (v) mobile – CIF, 100 frames, and (vi) football – CIF, 90 frames. The result revealed that MPU has chance to switch its frequency into lower level but incurred estimation errors of DSP as listed in Table 4(b). Take the peek error in 246 cycles/MB (i.e., the football in Table 4(b)) as the worse case with D1 (totally 1350 MBs) resolution encoded in real-time (30 fps), the total es-

timation error is within 246 × 1350 × 30 = 9.96 MHz/sec, which is quite acceptable comparing to the topmost frequency of DSP (228 MHz).

$$T^{I}_{IQ/IT} = \left[N^{I}_{4x4CBP} + N^{I}_{16x16} \times (16 \times P_{16x16} + 1) + N^{I}_{8x8} \times 8 \times \left(\frac{P_{Cb} + P_{Cr}}{2}\right) \right] \times T_{4x4IQ/IT} + N^{I}_{8x8} \times 2 \times T_{2x2IQ/IT} \quad \textbf{(2.1.1)}$$

$$T^{P}_{IQ/IT} = \left[N^{P}_{4x4CBP} + N^{P}_{16x16} \times (16 \times P_{16x16} + 1) + N^{P}_{8x8} \times 8 \times \left(\frac{P_{Cb} + P_{Cr}}{2}\right) \right] \times T_{4x4IQ/IT} + N^{P}_{8x8} \times 2 \times T_{2x2IQ/IT} \quad \textbf{(3.1.1)}$$

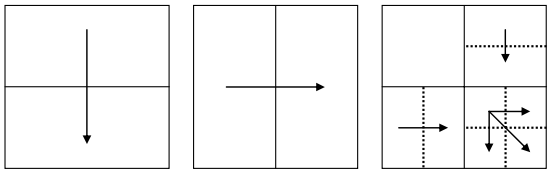

Fig. 4. Policy for reducing MV calculations according to distinct MB partitions

Table 4. (a) Probabilities for nonzero 4x4 blocks of a luma MB in intra 16x16 mode, and non-zero 4x4 blocks of a chroma MB in intra 8x8 mode; (b) Estimation errors of each test sequence for DSP

(a)

Test sequence	Probabilities (%)		
	luma	chroma	
		Cb	Cr
Container	84.5	47.6	42.4
Silent	87.5	43.5	43.8
News	67.2	41.3	37.9
Foreman	66.4	43	40
Mobile	61.6	38.5	39.7
Football	84.1	43.8	44.2

(b)

Test sequence	Max. Error (cycle/MB)	Avg. Error (cycle/MB)
Container	62	13
Silent	85	33
Foreman	105	41
News	89	27
Mobile	138	86
Football	246	55

6 Conclusion

In this paper, we proposed a novel power-aware scheme of H.264/AVC video player for PAC SoC platform based on its dual-core architecture and DVFS capability. The power-aware scheme is derived from a coarse-grain model according to a user's behaviors on playing a video and a fine-grain model along with continuous video decoding. A workable reduction scheme is also proposed to seek for further energy-saving under soft real-time constraint. Our work provides a valuable solution for designing a state-of-the-art H.264/AVC video player on similar platforms, such as PAC SoC platform, which is a promising hardware solution in modern VLSI design process.

References

[1] Wiegand, T., Sullivan, G.J., Bjntegaard, G., and Luthra, A, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, issue 7, July 2003, pp. 560-576.

[2] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," *SPIE Conference on Applications of Digital Image Processing*, vol. 5558, part 1, Aug. 2004, pp. 454-474.

[3] Horowitz, M., Joch, A., Kossentini, F., and Hallapuro, A., "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, issue 7, July 2003, pp. 704-716.

[4] Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., and Wedi, T., "Video Coding with H.264/AVC: tools, performance, and complexity," *IEEE Circuit and Systems magazine*, Q1, 2004, pp. 7-28.

[5] Hari Kalva 1 and Borko Furht, "Complexity Estimation of the H.264 Coded Video Bitstreams," *The Computer Journal Advance Access published*, June 24, 2005.

[6] Nurvitadhi, E., Lee, B., Yu, C., and Kim, M., "A Comparative Study of Dynamic Voltage Scaling Techniques for Low-Power Video Decoding," *International Conference on Embedded Systems and Applications*, June 2003, pp. 23-26.

[7] Kihwan Choi, Ramakrishna Soma, and Massoud Pedram, "Off-chip latency-driven dynamic voltage and frequency scaling for an MPEG decoding," *Proceedings of the 41st annual conference on Design automation*, June 2004, pp. 07-11.

[8] Seongsoo Lee, "Low-Power Video Decoding on Variable Voltage Processor for Mobile Multimedia Applications," *ETRI Journal*, vol.27, no.5, Oct. 2005, pp.504-510.

[9] Song, J.; Shepherd, T.; Minh Chau; Huq, A.; Syed, L.; Roy, S.; Thippana, A.; Shi, K.; Ko, U., "A Low Power Open Multimedia Application Platform for 3G Wireless," *Proceedings of IEEE International SoC Conference*, 17-20 Sept. 2003, pp. 377-380.

[10] Knight, W., "Two heads are better than one [dual-core processors]," *IEE Review*, vol.51, no.9, Sept. 2005, pp. 32- 35.

[11] Zhaowei Teng; Peng Liu; Liya Lai, "Physical design of dual-core system-on-chip," *Proceedings of IEEE International Workshop on VLSI Design and Video Technology*, May 2005, pp. 36- 39.

[12] Texas Instruments Inc., white paper of SmartReflex™ Technologies, "SmartReflex™ power and performance management technologies — reduced power consumption, optimized performance", Sept. 2005. *http://focus.ti.com/pdfs/wtbu/smartreflex_whitepaper.pdf*

[13] Juin-Ming Lu et al., "DVFS SoC Architecture and Implementation," *SoC Technology Journal, Taiwan*, vol. 3, Nov. 2005, pp. 84-91.

[14] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power Aware Video Decoding," *Proc. Picture Coding Symposium*, 2001, pp. 303-306.

[15] K. Choi, K. Dantu, W. Cheng, and M. Pedram, "Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder," *Proc. Int'l Conf. Computer-Aided Design*, 2002, pp. 732-737.

[16] A.C. Bavier, A.B.Montz, and L.L.Peterson, "Predicting MPEG execution times," in *Proceedings of ACM SIGMETRICS'98*, 1998, pp. 131-140. 97.