J Sign Process Syst (2011) 64:161–175
DOI 10.1007/s11265-010-0574-6

# A Fast H.264 Intra Frame Encoder with Serialized Execution of 4×4 and 16×16 Predictions and Early Termination

Jin-Su Jung · Young-Joon Jo · Hyuk-Jae Lee

Received: 19 January 2010 / Revised: 12 December 2010 / Accepted: 13 December 2010 / Published online: 5 January 2011

**Abstract** This paper presents a fast H.264 intra frame encoder that processes a single macroblock of 1920×1080 size video in 334 cycles on average which is 20% faster than the previous best design. The speed-up is mainly achieved by early termination of either 4×4 intra prediction or 16×16 intra prediction. The executions of intra 4×4 and 16×16 predictions are serialized and the second prediction is terminated early by using the cost of the first prediction as the stop criterion. A simple and efficient algorithm by making use of spatial locality is proposed to select the mode that is processed first. To avoid the bubble cycles caused by this serialized execution of 4×4 and 16×16 predictions, the modified processing order presented in (Jung et al. 2008) is employed for intra 4×4 prediction in order to schedule dependent 4×4 blocks apart from each other. To further reduce the execution time of 4×4 prediction, neighboring pixels with the same value are grouped, and only one prediction mode in the group is evaluated. Experimental results show that the PSNR drop is 0.0619 dB and the bitrate increase is 0.842% when compared with the JM reference software. The additional hardware cost to support the proposed methods is less than eight thousand gates which are very small when compared with the hardware size of a whole intra frame encoder.

This is an extended version of the paper presented at IEEE Workshop on Signal Processing Systems in October 2009 in whichParts of Section 3.D and Section 4 are already published.

J.-S. Jung · Y.-J. Jo · H.-J. Lee (✉)
Inter-university Semiconductor Research Center,
Department of Electrical Engineering, Seoul National University,
Seoul, South Korea
e-mail: hjlee@capp.snu.ac.kr

J.-S. Jung
e-mail: janghack@capp.snu.ac.kr

Y.-J. Jo
e-mail: tigrage@gmail.com

**Keywords** H.264 · Intra prediction · Intra frame encoder · Early termination · Mode selection

## 1 Introduction

The H.264/Advanced Video Coding (AVC) standard [1] introduces aggressive compression tools such as spatial prediction, adaptive block size motion compensation and 4×4 block based prediction. As a result, the H.264/AVC standard outperforms previous video coding standards in compression efficiency [2]. Intra frame prediction is one of those tools for compression enhancement that uses neighboring pixels to predict the current coding block. H.264/AVC compression with only intra frame prediction is especially suitable for low cost and low power applications such as a digital still camera or a video recorder, which cannot afford the complexity of inter frame prediction.

Extensive research efforts have been made to reduce the computational complexity of intra prediction [3–12]. One of the most popular techniques for complexity reduction is an early decision of prediction modes among the nine prediction modes for the 4×4 block size and the four prediction modes for the 16×16 block size. A number of previous techniques utilize the fact that a prediction mode is strongly related to the edge, texture, or direction of the contents of the block. Therefore, the contents of the block are analyzed first, and then only a subset of prediction modes are computed according to the contents [4–6]. In other techniques for fast intra prediction, only one of the

4×4 and 16×16 predictions is evaluated [7–9]. The smoothness of a macroblock is estimated and then 16×16 prediction is chosen when the block is smooth whereas 4×4 prediction is performed otherwise.

The complexity reduction techniques based on early mode decision are widely used for the software implementation of intra prediction. However, they are seldom employed by a hardware implementation because of two main reasons. The first reason is that a precise early decision often requires a complex algorithm which is too expensive to be implemented in hardware. Thus, a hardware-based mode decision using a relatively simple algorithm often makes an inaccurate selection. To avoid a performance drop-off by a wrong decision, an early termination algorithm is employed in [10]. The risk of performance drop is somewhat reduced in the early termination scheme which does not completely discard the unselected mode but terminates the computation only when further computation of the unwanted mode leads to a very small chance for the mode to be determined as the final mode.

The second reason that prevents a hardware implementation from employing early mode decision is that the hardware utilization may be decreased when early mode decision is employed. For efficient utilization of hardware resources for intra prediction, one of the main obstacles is the dependence between consecutive 4×4 predictions. Intra prediction of a 4×4 block depends on the reconstructed pixels in its neighboring 4×4 blocks, and therefore, intra prediction hardware must remain idle while the reconstruction of the neighboring blocks is being completed. Hence, the execution of intra prediction and reconstruction are often serialized. In [3, 11, 12], the idle cycles (often called bubbles) are avoided by performing 16×16 intra prediction (denoted by I16 hereafter) during the bubbles of 4×4 intra prediction (denoted by I4 hereafter). This interleaved execution of I4 and I16 is reasonable when both I4 and I16 are always executed. However, this interleaved execution makes it almost impossible to employ the early mode decision between I4 and I16 because interleaved execution implies that both I4 and I16 must be executed in parallel.

This paper attempts to solve the hardware under-utilization problem when an early mode decision/termination is employed and to achieve a speed-up of hardware-based intra prediction without a significant degradation of compression efficiency. To this end, I4 and I16 are executed in a serial manner and the speed-up is achieved by early termination of the mode that is processed second between I4 and I16 with the termination criterion obtained from the cost of the mode processed first. The processing order is determined from the intra prediction modes of neighboring macroblocks. The serialized execution of I4 after I16 (or I16 after I4) prevents the widely-used technique that interleaves I4 and I16 for removing bubble cycles of I4 [3, 11]. In order to reduce the bubble cycles even for the serialized execution of I4 and I16, this paper employs the modified processing order of I4 presented in [10]. In the hardware implementation of intra prediction in [10], the execution order of 4×4 blocks are changed to avoid the dependence between consecutive intra predictions and consequently allow consecutive executions of independent intra predictions resulting in the reduction of the bubble cycles without interleaved execution with I16. An additional speedup technique is also proposed for the case where predictor pixels have identical values (see Section 3.5). As a result, the average execution time for a single macroblock is reduced to 334 cycles for a 1920×1080 size video whereas the previous best design requires 417 cycles.

The rest of this paper is organized as follows. Section 2 briefly introduces previous hardware-based intra prediction techniques and Section 3 presents the proposed fast intra prediction. Section 4 explains the details of the pipeline schedule of the proposed intra prediction. Section 5 presents the hardware implementation of the proposed pipeline and Section 6 gives comparisons with previous works. Section 7 concludes the paper.

## 2 Previous Pipeline Schedules for Fast Intra Prediction

In the baseline or main profile H.264/AVC compression standard, intra prediction is performed in two block sizes: 4×4 block for I4 prediction and 16×16 block for I16 prediction. In I4 prediction, a 16×16 macroblock is decomposed into 16 4×4 blocks. Each of these 4×4 blocks is predicted from its neighboring pixels. Figure 1(a) shows a 4×4 block with its predictor pixels labeled from A to M. The sixteen pixels labeled from a to p represent the 4×4 block to be predicted. Nine prediction modes are supported in I4 prediction. Depending on the prediction mode, some of the predictor pixels are chosen and used as the predictors. Figure 1(b) shows the directions of eight prediction modes. The number given with each arrow represents the prediction mode number. One mode called the DC prediction mode (Mode 2) is not shown in this figure because it does not have a prediction direction but it uses the average of upper and left predictor pixels. In I16 prediction, the entire 16×16 block is predicted with four prediction modes: Horizontal, Vertical, DC, and Plane. Similar to I4 prediction, the neighboring pixels of a 16×16 macroblock are used for the predictors of the I16 prediction. For 8×8 Chroma block, intra prediction is performed in the same manner as I16 prediction. Details of intra prediction are available in [11].

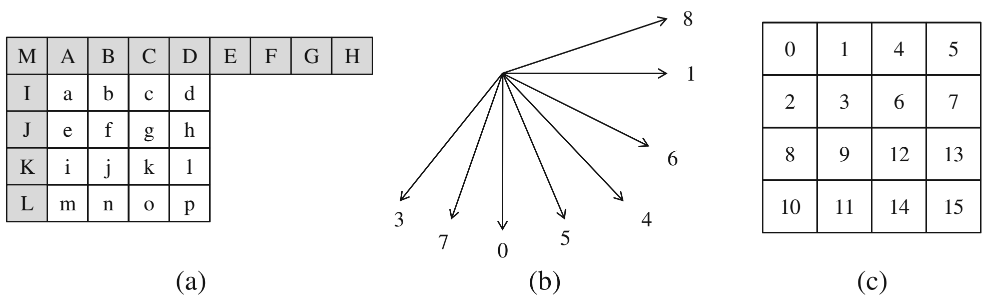


**Figure 1** Intra prediction of a 4×4 block. **a** 4×4 block labeled from a to m predicted from neighboring pixels labeled from A to M **b** eight prediction directions **c** processing order of 16 4×4 blocks defined in the H.264/AVC standard.

Figure 1(c) shows the processing order of the 16 4×4 blocks in a macroblock. In this figure, each small square represents a 4×4 block and a large square represents a 16×16 macroblock. The number inside the small square represents the processing order of the 4×4 block prediction defined in the H.264/AVC standard. There exists dependence between consecutively-executed 4×4 blocks such that the intra prediction of one block depends on that of the previous 4×4 block. For example, Block 1 depends on Block 0 because some of the predictor pixels of Block 1 belong to Block 0. In other words, the intra prediction of Block 1 needs the pixels in Block 0. Note that the H.264 standard requires these predictor pixels to be the reconstructed pixels from the result of intra prediction of Block 0. Thus, Block 1 needs to wait for the completion of both intra prediction and reconstruction of Block 0. This implies that there exists a period between the intra predictions of Block 0 and Block 1 and the reconstruction of Block 0 is performed during this period. This period is called a bubble in [11] as the hardware resource for intra prediction remains idle during this period.

For a hardware-based intra prediction, previous research efforts have focused on the utilization of hardware resources during this bubble. In [11], the underutilization is avoided by interleaving I4 prediction with I16 prediction. Figure 2(a) shows the intra prediction presented in [11]. In Fig. 2(a), B0 represents I4 prediction of Block 0 shown in Fig. 1(c) whereas B1 and B15 represent I4 prediction of Block 1 and Block 15, respectively. For I16 prediction, a macroblock is decomposed into 16 4×4 blocks and intra prediction is performed for each 4×4 block. In Fig. 2(a), I16-B0 represents the first 4×4 block of I16 prediction. This figure shows how I4 and I16 are performed from the left to the right as the processing time proceeds. Note that a macroblock is decomposed into 4×4 blocks and the executions of I4 and I16 for each 4×4 block are interleaved. The main reason for this interleaved execution is to utilize the bubble cycles for the execution of I16. Recall that I4 for B1 depends on the reconstruction of B0 and consequently bubble cycles are required between I4 for B1 and B0. To avoid the waste of hardware resources during these bubble cycles, they are used to perform I16-B0 [11]. This interleaved execution is one of the main contributions presented in [11]. This figure also shows the execution of Chroma prediction and DCTQ (Discrete Cosine Transform and Quantization) operation for the best Chroma and I16 mode. In the three rightmost boxes of the figure, Chroma, I16-best and Chroma-best represent the intra prediction of Chroma data, DCTQ for the best I16 mode, and the DCTQ for the best Chroma mode, respectively. Further details are available in [11]. In this schedule, the number inside parenthesis represents the execution cycle. For example, B0 requires 36 cycles whereas I16-B0 requires 12 cycles. In total, 1,060 cycles are required to complete intra prediction of a single

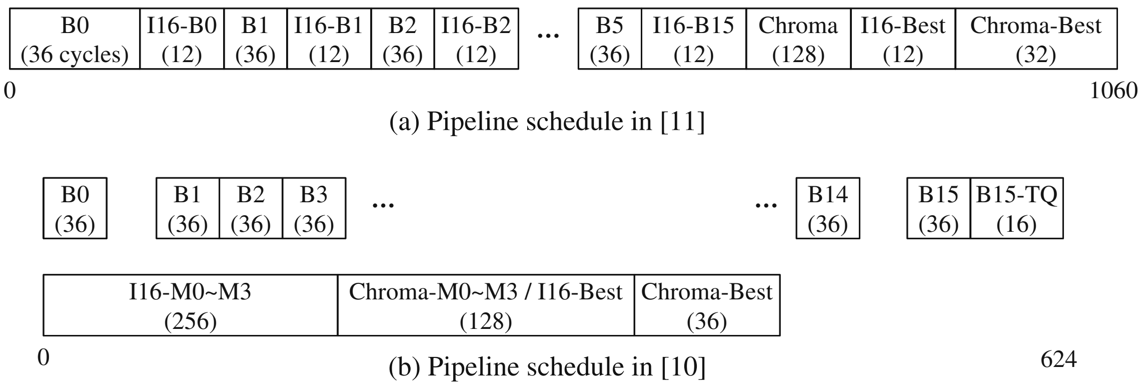


**Figure 2** Previous pipeline schedules. **a** Pipeline schedule in [11]. **b** Pipeline schedule in [10].

**Figure 3** Modified processing order of I4 predictions [10].

| | | | |
|---|---|---|---|
| 0 | 1 | 3 | 5 |
| 2 | 4 | 7 | 9 |
| 6 | 8 | 11 | 13 |
| 10 | 12 | 14 | 15 |

macroblock. Note that the execution cycles depend on the hardware architecture and this number is obtained with the assumption that the hardware is implemented with 4-pixel parallelism [10].

Another approach to avoid the underutilization during bubbles is proposed in [10]. The speed up of I4 is achieved with the modified processing order as shown in Fig. 3. Recall that the processing order shown in Fig. 1(c) causes the prediction of one 4×4 block to be dependent on the pixels from the previously processed 4×4 block so that bubble cycles need to be inserted between the executions of the two blocks. For example, I4 for B1 requires the pixels in block B0. Therefore, it is necessary to wait for the reconstruction of B0 before I4 for B1 begins. In order to avoid the dependence between consecutively executed 4×4 blocks, the intra prediction in [10] proposes to change the processing order as shown in Fig. 3. In the new processing order, the number of dependent blocks between consecutively-executed blocks is significantly reduced. In the original order, 12 blocks are dependent on their previous blocks so that 12 bubbles are required [10]. In the rescheduled order shown in Fig. 3, the number of dependent blocks is reduced to five: two left dependencies between Blocks 0 and 1 as well as Blocks 14 and 15, and three up-right dependencies between Blocks 1 and 2, Blocks 7 and 8, as well as Blocks 13 and 14. Left dependencies affect six I4 modes (1, 2, 4, 5, 6, 8) whereas up-right dependencies affect two I4 modes (3, 7). As the up-right dependencies account for only two modes (I4 modes 3 and 7), these modes are not evaluated for Blocks 2, 8 and 14 (i.e., the best I4 mode is chosen among modes 0, 1, 2, 4, 5, 6, and 8). Consequently, no bubbles are inserted for these blocks. As a result, only two bubbles are placed between Blocks 0 and 1 as well as Blocks 14 and 15. The hardware resource designed for I4 is used efficiently from the start to the end of I4 prediction with only two bubbles during which the hardware remains idle.

In [10], additional speed-up is achieved by early termination of I4 using the cost of I16 as the stop criterion. This schedule is shown in Fig. 2(b). I4 and I16 are performed in parallel as the upper pipeline represents the I4 whereas the lower pipeline does I16 and Chroma execution. In the upper pipeline, B0, B1, and B15, represent Block 0, Block 1, and Block 15, respectively. In the lower pipeline, I16-M0~M3 represents the execution of the four prediction modes (from M0 to M3) for I16. The next box Chroma-M0~M3/I16-Best represents the parallel execution of the four prediction modes for Chroma prediction and the DCTQ for the best I16 mode. Note that the parallel executions of I4 and I16 require hardware resources to be doubled when compared with them for the pipeline in Fig. 2(a). As I16 requires less computation time than I4, I16 completes earlier than I4. By using the final cost of I16 as the stop criterion, the speed up of I4 is achieved by early termination [10]. To this end, the cost of I16 is compared with the expected cost of I4 estimated from the intermediate result of I4 and then I4 is terminated early whenever the estimated cost of I4 is larger than the cost of I16.

The above two schemes attempt to have an efficient schedule for I4 and I16 predictions for the speed-up of intra prediction. Another approach for fast intra prediction is a skip of prediction modes of I4. A modified three step algorithm is the technique for hardware-based fast I4 prediction employed by the design in [3]. The algorithm is shown in Fig. 4. In the first step, modes 0 and 1 are performed. In the second step, modes 2, 3, and 4 are performed. By the time modes 5 or 6 start, the costs of mode 0 and 1 are available. In the final step, modes 5, 7 or modes 6, 8 are selectively performed according to the costs of mode 0 and 1. As a result, two I4 modes are always excluded by this algorithm.

## 3 Proposed Fast Intra Prediction

This section presents a new fast intra prediction algorithm that overcomes the limited speed up achieved by the algorithm in [10]. The limitation of this schedule lies in the fact that the early-termination rate of I4 is not high. When the I4 mode is chosen as the final mode, the execution of I4 cannot be terminated early because the cost

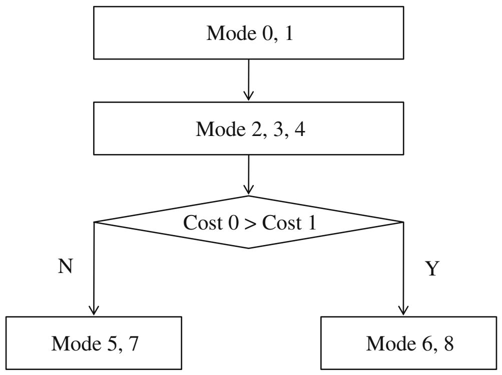


**Figure 4** A modified three step algorithm for intra prediction [3].

of I4 is smaller than the cost of I16. In general, the I4 mode is chosen more frequently than the I16 mode. Therefore, the early termination in the pipeline as shown in Fig. 2(b) achieves limited speed-up. In order to overcome this limitation, it is necessary to have a scheme that allows the frequently-chosen mode is performed first so that the resulting cost is used for the early termination of the seldom-chosen mode. Section 3.1 presents the outline of the algorithm whereas the details of the algorithm are presented from Sections 3.2 to 3.5.

### 3.1 Flow of the Proposed Fast Intra Prediction

Figure 5 shows the sketch of the proposed algorithm. The first step predicts the mode between I4 and I16 that is likely to be chosen as the final mode. In this step, the prediction is made by using the spatial correlation with neighboring macroblocks (see Section 3.2 for details). If I4 is chosen over I16, then I4 is performed before I16. I16 is terminated early by using the cost of I4 as the stop criterion. On the other hand, if I16 is chosen in the first step, then I16 is performed before I4. Then, I4 is terminated early with the cost of I16 as the stop criterion. In this way, the effect of early termination on speed-up is maximized.

The advantage of the proposed schedule over the previous schedule in Fig. 2(b) is that the selection between I4 and I16 for early termination is possible whereas the previous schedule always selects I4 for the candidate of early termination resulting in a limited speed up in the case where I16 is chosen as the final mode. Adopting 8-pixel parallel hardware implementation, the previous pipeline can process both I4 and I16 in parallel, with 4-pixel parallel hardware dedicated to each of I4 and I16. As I4 takes longer than I16, the hardware for I16 is often wasted when I4 is not terminated early. On the other hand, the proposed pipeline processes I4 with 8-pixel parallel hardware dedicated to I4 almost twice faster than the previous pipeline in Fig. 2 (b) does. This is because only 4-pixel parallel architecture is used for I4 in the previous pipeline (the other 4-pixel architecture is dedicated to I16). After I4, the 8-pixel parallel hardware is dedicated to I16, and speed up is achieved by early termination. Therefore, the proposed pipeline schedule achieves better hardware utilization than that in Fig. 2(b), leading to faster execution time than the schedule in Fig. 2(b). Similarly, the proposed pipeline uses 8-pixel parallel hardware for I16 first and then later for I4 when I16 is over. The hardware under-utilization is minimized with the proposed pipeline.

For further speed-up by reducing the number of prediction modes for I4, this paper also employs the modified three step algorithm which is proposed in [3]. Recall that the algorithm always discards two prediction modes among the nine prediction modes without a much degradation of R-D performance. The plane mode for I16 and Chroma predictions is omitted to reduce the complexity of the hardware as in [3]. In addition, this paper proposes two additional techniques: early termination among the I16 modes and additional prediction mode reduction for I4. The details of these two additional techniques are discussed in Sections 3.4 and 3.5, respectively.

### 3.2 Mode Selection Between I4 and I16

In the algorithm in Fig. 5, the first step is a selection between I4 and I16. The selection is made by observing the intra prediction modes of neighboring macroblocks. The prediction modes of the upper and left macroblocks are checked first, and I16 is selected if one of the two neighboring modes is I16. Otherwise, I4 is selected. The top-leftmost macroblock in a frame has no neighboring macroblocks. In this case, I4 is always selected because I4 is selected more frequently as the best mode than I16 does.

Table 1 shows the probability of the selected mode to be finally chosen as the best mode. Four test sequences of size 1920×1080 are used. The high accuracy implies that the selected mode is the best mode with high probability. As shown in Table 1, the simple selection scheme achieves very high accuracy, at least 79%, in these four test sequences.

```
if (I16 selected) {                    // see Section III.B
    I16 prediction;
    Derive the threshold for early termination of I4
    I4 prediction with early termination (see Section III.C)
} else
    I4 prediction;
    Derive the threshold for early termination of I16
    I16 prediction with early termination (see Section III.D)
}
```

**Figure 5** The sketch of the proposed intra prediction algorithm.

**Table 1** Accuracy of prediction between I4 and I16.

| Test sequence | Accuracy (%) |
|---|---|
| Blue sky | 92.02 |
| Tractor | 90.07 |
| Pedestrian area | 79.24 |
| Rush hour | 79.13 |

### 3.3 Early Termination of I4

When I16 is selected over I4, then I16 is performed first. In this case, the cost of I16 is used as the stop criterion for the early termination of I4. Then, the early termination algorithm presented in [10] is employed. Figure 6(a) shows the flow chart of the early termination. After the prediction of each 4×4 block, the cost is added to the accumulated costs of the prediction of previous 4×4 blocks. This addition is represented by the block with $C4_{accum}(N)=C4(N)+C4_{accum}(N-1)$, where $N$ represents the number of the current 4×4 block, $C4_{accum}(N)$ represents the cost accumulated for $N$ 4×4 blocks, and $C4(N)$ represents the cost of the 4×4 intra prediction of the N$^{th}$ block. The cost of 4×4 intra prediction is the sum of absolute transformed differences (SATDs) of each 4×4 block. In the next step, the accumulated cost is compared with the early termination threshold, $Th(N)$. The selection of the threshold is to be discussed in details in the next paragraph. If the current accumulated cost $C4_{accum}(N)$ is larger than the threshold $Th(N)$, the total cost of I4 is expected to be larger than that of I16, and I4 is early terminated. If the accumulated cost is smaller than the threshold, $N$ is incremented by one and the next iteration of the loop is performed again. If $N$ reaches 16 without being early terminated, I4 is completed.

The threshold function, $Th(N)$, determines the amount of computation saving by early termination so that the selection of $Th(N)$ is important for an effective trade-off between computation saving and compression efficiency. The most rigorous threshold that ensures no loss in compression efficiency would be the total cost of I16 which enforces I4 to simply terminate when the intermediate cost of I4 is larger than the total cost of I16. On the other end, a flexible threshold is the intermediate cost of I16 for the corresponding 4×4 blocks that enforces early termination when the I4 intermediate cost is larger than the I16 intermediate cost of the equivalent 4×4 blocks. The threshold function used in [10] is a value between the two thresholds and is defined as follows [10]:

$$Th(N) = \frac{Cost_{I16}}{16} \times (N + M(N)) \quad (1)$$

where $Cost_{I16}$ denotes the total cost of I16, $N$ is the index of the 4×4 block currently being processed, and $M(N)$ is a margin considering the cost variation of the remaining 4×4 blocks. From experimental results, $M(N)$ is defined as follows:

$$M(N) = M(0) \times (1 - N/16) \quad (2)$$

where $M(0)$ of 0.75 is experimentally chosen for the early termination of I4 from the result of I16. The computation of (1) is not very complex as it can be implemented with a table look-up operation, one addition, and one multiplication.

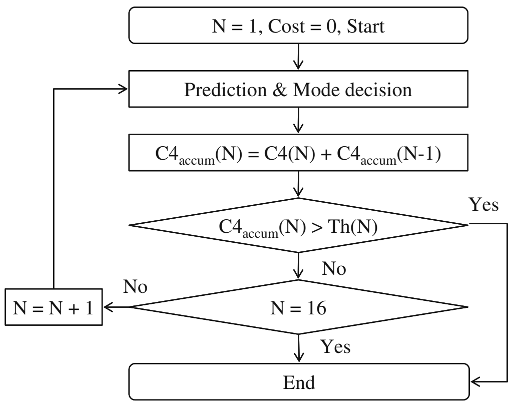
### 3.4 Early Termination of I16

When I4 is selected over I16, I16 is terminated early with the cost of I4 used as the stop criterion. The early termination of I16 is slightly different from I4 in that one prediction mode of I16 is performed for the entire macroblock and then the next prediction mode is performed. Figure 6(b) shows the early termination of I16. In order to share the hardware for both I4 and I16 efficiently, a macroblock is decomposed into 4×4 blocks and then the prediction and cost calculation are performed block by block of size 4×4. At the end of the cost calculation of each 4×4 block, the cost is added to the accumulated cost and then compared with the threshold, $Th(N)$. When the accumulated cost is greater than the threshold, the prediction mode is terminated early. Then, the next prediction mode is performed. The outer loop in Fig. 6(b) represents the iteration for performing three prediction modes. Note that the transform of DC coefficients in an I16 mode is not included in Fig. 6(b) for simplicity. Also note that the fourth prediction mode (Plane mode) is often excluded for I16 prediction [3, 10, 12]. The function given in (3) is used for the threshold

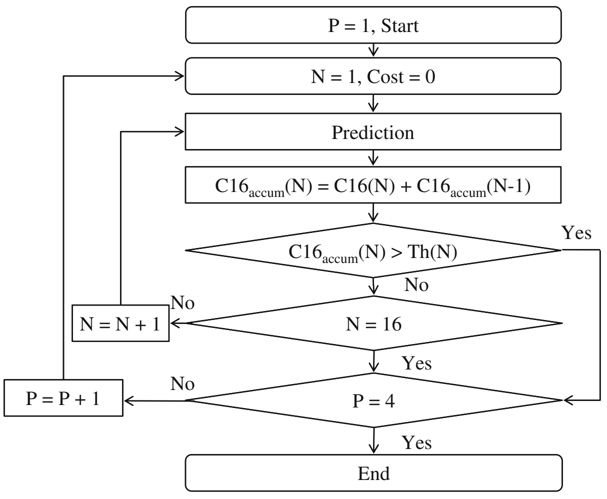
$$Th(N) = \frac{Cost_{I4}}{16} \times (N + M(N)) \quad (3)$$

where $M(N)$ is the same as (2) with the value of $M(0)$ chosen as 0.5 by experiments.

I16 is performed mode by mode, and consequently, the cost of one prediction mode is available before the start of the next mode and it can be used as the stop criterion for the early termination of the next mode. Thus, the early termination of I16 is attempted from the second prediction mode using the cost of the first prediction mode as the stop criterion. For early termination to be effective, it is important to choose the prediction mode to be performed first. The processing order of the three I16 modes is decided as follows. First, the best I16 mode of the left macroblock is selected as the first mode to be performed. If the first mode is chosen as 0, then the second and third modes are 1 and 2, respectively. If the first mode is 1, the second and third modes are 0 and 2, respectively. If the first mode is 2, the



(a) Early termination of I4 prediction



(b) Early termination of I16 prediction

**Figure 6** Flowchart of intra prediction with early termination. **a** Early termination of I4 prediction **b** Early termination of I16 prediction.

second and third modes are 0 and 1, respectively. For the threshold, the same function similar to (3) is used again and the value of *M(0)* is chosen as 0.25 in this case. Note that the reference cost (cost of I4) and *M(0)* are updated whenever the cost of one prediction mode is smaller than the cost of the previous best mode. For example, suppose that the best mode becomes I16 after evaluating the first mode which uses the cost of I4 for *Th(N)*. Then, the second I16 mode uses the cost of the first I16 mode, instead of the cost of I4, as the stop criterion. Due to the change of the termination type, *M(0)* is also changed to 0.25.

## 3.5 Prediction Mode Reduction

The modified three-step algorithm in [3] reduces the number of I4 modes down to seven with less than 1% increase in the bit rate. As this algorithm is simple enough for hardware implementation, the intra prediction in this paper also employs the modified three-step algorithm.

Further exclusion of the I4 prediction modes is achieved by making use of the fact that adjacent pixels sometimes have the same values. Recall that Fig. 1(a) shows the predictor pixels labeled from A to M in the left, upper and upper-right of a 4×4 block. If the values of the predictor pixels are all the same, then the SATDs of all nine modes are the same. Thus, it is not necessary to perform all nine modes, but only one mode is necessary to be performed. This mode is called the *"representative mode"* among the modes with the identical predictors. With only the representative mode to be predicted, the computational complexity of I4 is significantly reduced. The complexity reduction can be achieved even when all predictor pixels are not identical, but when a certain group of prediction modes are identical. For example, if predictor pixels from A to D, from I to J, and M are all equal, modes 0, 1, 2, 4, 5, 6, and 8 result in the same SATD. Thus, only one mode among the seven modes is necessary to be performed. Table 2 summarizes the relationship between the identical predictor pixels and the prediction modes that result in the same SATD. For example, the fourth row shows that identical predictor pixels from A to H lead to the same SATDs of modes 0, 3, and 7. Hereafter, the identical predictor group is denoted by IPG and the IPG-based mode selection algorithm is denoted by MS-IPG.

The modified three-step algorithm requires I4 modes 0 and 1 to be always executed whereas mode 0 or 1 may be excluded in MS-IPG. Therefore, the modified three-step algorithm cannot be performed when mode 0 or 1 is excluded by MS-IPG. In this case, it is necessary to select the algorithm between the modified 3-step algorithm and

**Table 2** Identical pixel group and I4 prediction modes with identical SATD.

| Group number | Identical predictor group | I4 modes with identical SATD |
|---|---|---|
| 0 | A, B, C, D, E, F, G, H, I, J, K, L, M | all modes |
| 1 | A, B, C, D, I, J, K, L, M | 0, 1, 2, 4, 5, 6, 8 |
| 2 | A, B, C, D, I, J, K, L | 0, 1, 2, 8 |
| 3 | A, B, C, D, E, F, G, H | 0, 3, 7 |
| 4 | I, J, K, L | 1, 8 |

the MS-IPG. As the modified 3-step algorithm excludes 2 prediction modes, the MS-IPG is more efficient only when the number of excluded modes is larger than 2. As shown in Table 2, the MS-IPG excludes more than two prediction modes when the group number is between 0 and 3. Therefore, the MS-IPG is chosen over the 3-step algorithm when the group number is less than 4. Otherwise, the modified 3-step algorithm is selected. In this manner, the number of I4 modes is always smaller than or equal to 7.

## 4 Pipelined Execution of the Proposed Intra Prediction

### 4.1 Intra 4×4 Prediction

Figure 7 shows pipelined execution of the I4 of the first three 4×4 blocks (Blocks 0, 1 and 2). The numbers at the top of the figure represent execution cycles and the numbers in the left are the processing orders of the seven I4 modes. Mode 0 is performed at the beginning of I4 whereas Mode 7 or 8 is performed at the end of I4. The execution of one mode consists of seven operations and each box in Fig. 7 represents one operation. Box St represents the start cycle in which neighboring pixels are selected and predictors are generated. Box P represents intra prediction operation whereas box D represents the calculation of the difference between the predictors and current block. T represents integer transform which requires 2 cycles for completion (T1 and T2). Box C and Box B represent cost calculation and best I4 mode decision, respectively. After I4 of Block 0 (which is the same as B0 in Fig. 3) is completed, the reconstruction of the best mode for Block 0 begins at cycle 21 with quantization represented by Boxes Q1, and Q2. Then, inverse quantization (IQ1 and IQ2) is performed at cycle 23 and 24. Note that both Q and IQ require 2 cycles for completion. As 4-pixel parallel hardware is implemented for the reconstruction, inverse transform requires 4 cycles, from IT1 to IT4. Finally, reconstruction (denoted by R) is performed in cycle 29. Because of the left dependence between Blocks 0 and 1, Block 1 can start only after the last R is performed (in cycle 32) and therefore I4 of Block 1 must wait for the completion of reconstruction of Block 0. As the last reconstruction step is performed in cycle 32, the first 'St' step of Block 1 can begin in cycle 33 (not that this schedule is not shown in the figure. Instead, the optimized schedule illustrated in the next paragraph is shown). Note that the last 'St' step of Block 0 (for prediction mode 7 or 8) is performed in cycle 14, the idle cycle of 18 (=33−14+1) is required between Block 0 and Block 1.

To reduce the bubble cycles, this paper proposes two optimizations. The first optimization takes advantage of the fact that four pixels are reconstructed in 1 cycle. The reconstruction hardware is designed in such a way that the rightmost four pixels are generated in the first cycle (cycle 29). Note that only the rightmost four pixels are necessary for the generation of predictors for Block 1. Thus, after the first reconstruction cycle, the IP of Block 1 can begin its 'St' operation. Thus, 3 cycles can be removed from the bubble cycles between Blocks 0 and 1. The second optimization is performed based on the reason that the predictors for Mode 0 are irrelevant to the reconstructed pixels in Block 0 (the predictors for Mode 0 are constructed from the pixels in the upper 4×4 block). Thus, Mode 0 of Block 1 can begin before the completion of the 'R' step of Block 0. On the other hand, Mode 1 of Block 1 depends on the reconstructed pixels of Block 0. Therefore, the 'St' operation of Mode 1 can begin after the first 'R' operation of Block 0. Thus, the 'St' operation of Mode 1 can begin at cycle 30 which implies that the 'St' operation of Mode 0 can begin at cycle 28, 2 cycles earlier than Mode 1. With the two optimizations, five bubble cycles can be reduced. However, just 4 cycles are removed from the bubble deliberately and the 'St' operation of Mode 0 begins at cycle 29 as shown in Fig. 7. The reason of the deliberate removal of only 4 cycles is for a simple control of Chroma scheduling which includes normal 4×4 block prediction for 2 cycles and 2×2 DC Hadamard for 1 cycle.

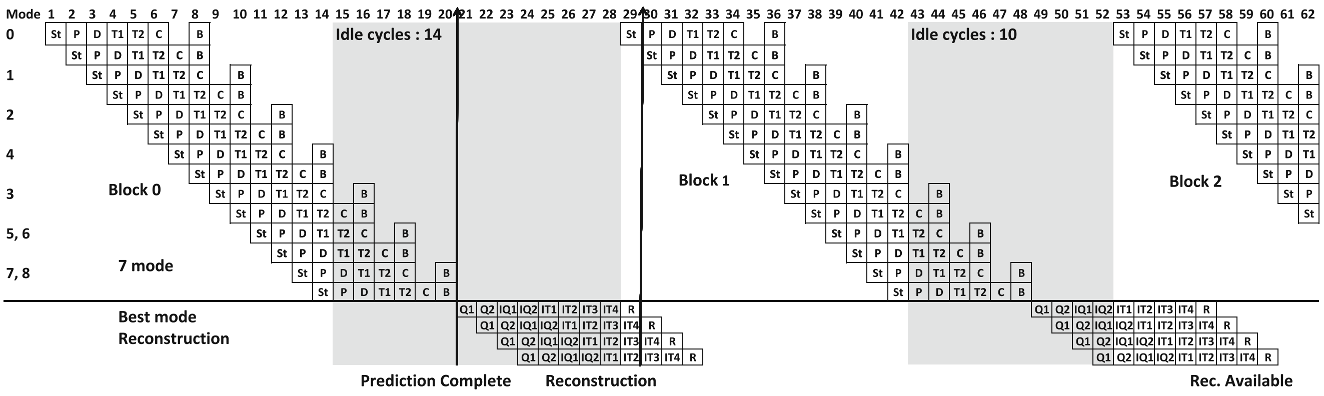


**Figure 7** Pipelined execution of intra 4×4 predictions for left and upper-right dependencies.

Therefore, bubble cycles need to be an odd number when DC Hadamard is performed during the bubble whereas bubble cycles need to be an even number otherwise. Ignoring these rules causes additional buffer to store temporal result of a 4×4 block to be used at the beginning of the next bubble. As a result, 14 bubble cycles are inserted between Block 0 and 1 as shown in Fig. 7 (from cycle 15 to cycle 28 represented by the gray area). Note that Block 15 is also dependent on its left block (Block 14). Thus, the pipeline execution is almost the same as that shown in Fig. 7. The only difference is that it is not necessary to take into consideration of the Chroma scheduling. Hence, 13 bubble cycles are inserted between Block 14 and Block 15.

Due to the upper-right dependence between Blocks 1 and 2, a bubble is also inserted between the two blocks. For the up-right dependence, Modes 0, 1, 2, 4, 5, 6, and 8 are irrelevant with its previous block. Due to the restriction in the processing order by the 3-step algorithm, Modes 5, 6, and 8 cannot be scheduled first. Thus, Modes 0, 1, 2, and 4 can be scheduled before the reconstruction of the previous block. As a result, eight bubble cycles are removed. The first optimization applied for Blocks 0 and 1 cannot be adopted in this case because Mode 0 needs the bottom four pixels of Block 1 which are available only after the last reconstruction step (Cycle 60). Thus, ten bubble cycles (from 43 to 52) are inserted between Blocks 1 and 2 as shown in Fig. 7. Note that there exist another two cases (between Blocks 7 and 8 and also between Blocks 13 and 14) that have upper-right dependence. Thus, ten bubble cycles are also inserted for these two cases.

Figure 8 shows the execution of Blocks 2, 3, and 4 which do not need bubbles caused by dependence. Therefore, Block 3 begins its 'St' operation at cycle 15 which is the next cycle of the last 'St' operation of Block 2. In this perfectly pipelined execution without any bubble, the computation of each mode is completed in every 2 cycles. Note that there is left dependence between Blocks 2 and 4 so that Block 4 needs to wait for the reconstruction of Block 2. This requirement does not cause additional slow-down because the computation for Block 3 provides enough delay between Blocks 2 and 4.

## 4.2 Pipeline Schedule of the Proposed Intra Prediction

Figure 9(a) and (b) show the two pipeline schedules in the proposed intra prediction. In both schedules, the executions of I4 and I16 are serialized. The first schedule shown in Fig. 9(a) performs I4 first followed by I16 whereas the second schedule in Fig. 9(b) performs I16 followed by I4. Then, one of the two schedules is chosen depending on the predicted modes of neighboring macroblocks as illustrated in Section 3.2.

In Fig. 9(a), B0 and B1 represent the I4 predictions of Blocks 0 and 1, respectively. The numbers inside parentheses of boxes represent the execution cycles. Note that the execution time is twice faster than that in Fig. 2(b). This is because the design in Fig. 9 employs 8-pixel parallel data path entirely dedicated to I4 whereas the design in Fig. 2(b) shares the data path by both I4 and I16, and consequently, only 4-pixel parallel data path is used for I4. B2-7 represents the consecutive execution of I4 predictions from B2 and B7 whereas B8-13 represents I4 predictions from B8 and B13. During the execution of I4, five bubbles are generated between B0 and B15. These bubbles are 14 cycles between B0 and B1, 10 cycles between B1 and B2, 10 cycles between B7 and B8, 10 cycles between B13 and B14, and 13 cycles between B15 and B16 (see details about these bubble cycles in the previous subsection). To avoid a waste of hardware resources, these bubbles are interleaved with Chroma and I16 predictions. For example, the bubble between B0 and B1 is utilized with the execution of Mode

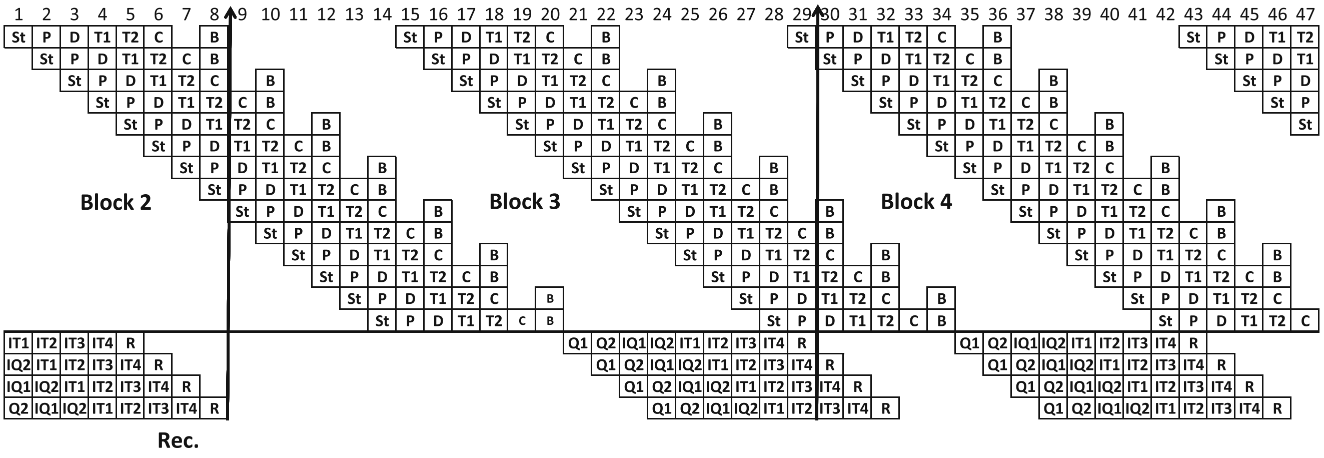
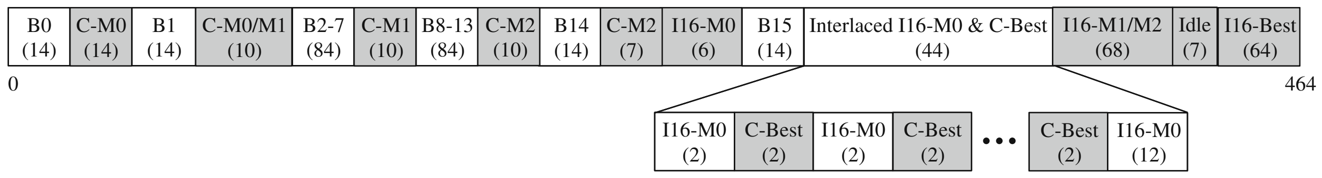
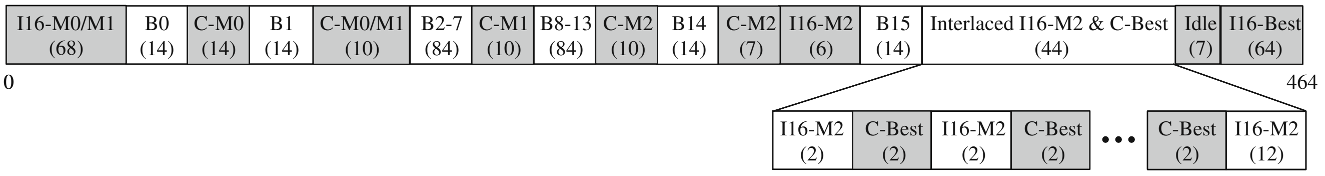


**Figure 8** Normal pipelined execution of intra 4×4 predictions.



(a) Pipeline schedule when I4 processed before I16



(b) Pipeline schedule when I16 processed before I4

**Figure 9** Proposed pipeline schedule. **a** Pipeline schedule when I4 processed before I16. **b** Pipeline schedule when I16 processed before I4.

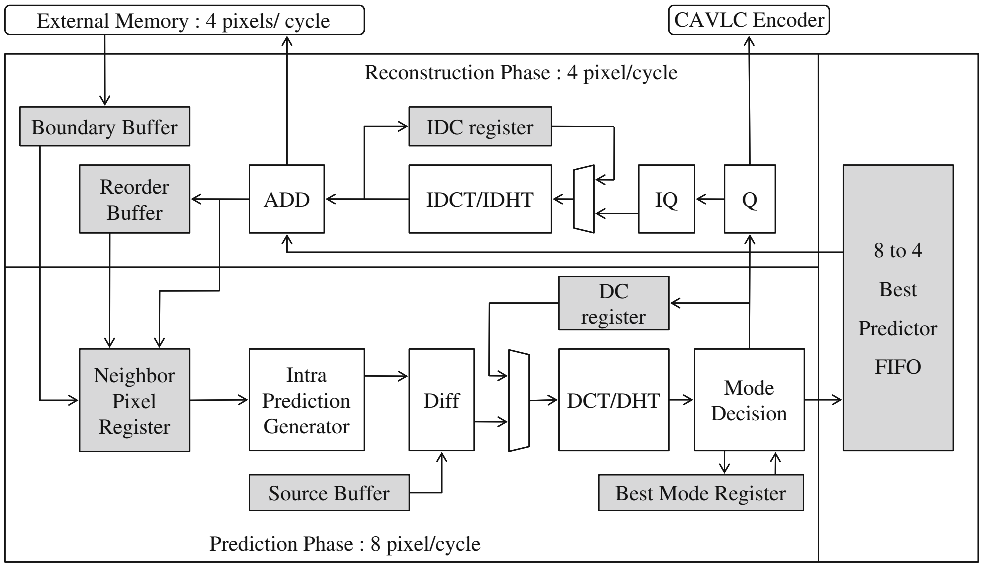
0 of the Chroma prediction (denoted by C-M0). In Fig. 9(a) and (b), C-M0, C-M1 and C-M2 represent intra predictions of Mode 0, 1, and 2 for Chroma data, respectively. It takes 17 cycles to process one mode of Chroma prediction (Each of Chroma U and V has 4 4×4 blocks (64 pixels) and, with eight pixel parallel implementation, 16 cycles are necessary to process 128 Chroma pixels (64 pixels for each of U and V). It takes additional 1 cycle to 2×2 transform the DC coefficients of U and V, making 17 cycles to process each mode). Recall that the bubble between B0 and B1 is only 14 cycles. Thus, C-M0 is not completed during the first bubble cycles and the remaining computation for the C-M0 is performed in the second bubble between B1 and B2. The fourth box from the left denoted by C-M0/M1 represents the sequential execution of C-M0 (for 3 cycles) followed by C-M1 (for 7 cycles). The remaining execution of C-M1 is performed after the execution of B2-7. C-M2 is performed for 10 cycles during the bubble between B13 and B14. Then, the remaining 7 cycles of C-M2 are performed during the bubble between B14 and B15. Recall that the bubble cycles between B14 and B15 are 13 cycles. Thus, the bubble includes 6 cycles after the completion of C-M2. These remaining 6 cycles are consumed by the execution of I16-M0 (the first mode of I16 prediction). The execution order of the modes of I16 may vary (see Section 4.3). Thus, I16-M0, I16-M1, and I16-M2 do not represent Mode 0, 1 and 2, of I16. Instead, they represent the I16 mode that is executed first, second, and third, respectively.

Each mode of I16 takes 34 cycles (I16 prediction has 16 4×4 blocks which take 32 cycles to predict the 256 Luma pixels and additional 2 cycles are required to generate the 4×4 transform of the I16 DC coefficients). Thus, the remaining part of I16-M0 is performed after B15. Then, the other modes of I16 predictions and DCTQ/reconstuction for the best Chroma mode (C-Best) are performed. The execution of C-Best requires 32 cycles to re-predict and reconstruct 8 4×4 blocks. However, the prediction hardware must remain idle for 16 cycles during C-Best computation because 4-pixel parallel hardware is applied for reconstruction. To avoid these idle cycles, C-Best and I16 are performed in an interleaved manner (Interleaved I16 and C-Best in Fig. 9). Then, I16-Best in Fig. 9 represents the execution cycles for the decision of the best I16 mode. This step is necessary only when I16 is selected as the best macroblock mode. It requires 64 cycles for prediction. Note that the 64 cycles for I16 selection are not needed when I4 is selected.

In the pipeline schedule shown in Fig. 9(b), the first two modes of I16 (denoted by I16-M0/M1) are performed first. Then, I4 and Chroma prediction are performed in the interleaved manner as the schedule presented in Fig. 9(b). This implies the third mode of I16 is not completed before the start of I4. Thus, the stop criterion for I4 early termination is chosen as the smaller cost of only the first two modes. It is possible for the cost of the third mode is less than the smaller cost of the first two modes. Thus, the early termination rate of I4 may be slightly decreased by this delayed evaluation of the third mode although experimental results show that this decrease is not significant. During the execution of I4, the execution order from B0 to B15 is the same as that shown in Fig. 9(a). Then, I16-M2 and C-Best are performed in an interlaced manner. The last two steps are same as those in Fig. 9(a).

The main advantage of the serialized execution of I4 and I16 over the previous schedules is that an early mode decision between I4 and I16 can be effectively used to speed up the computation time of the unselected mode with early termination. If I4 is selected over I16, then the first schedule (Fig. 9(a)) is adopted so that I4 is performed first. Then, I16 is terminated early by using the cost of I4 as the stop criterion. On the other hand, if I16 is selected over I4, then I16 is performed first and I4 may be terminated early by using the result of I16. In a software-based implementation of an early mode decision for H.264 intra prediction, it is often the case that only the selected mode is executed. In this case,

**Figure 10** Block diagram of the intra prediction hardware.



the unselected mode is simply discarded. The performance drop by this discard is often not very significant because a software-based implementation often uses a sophisticated algorithm to select the better mode. In the hardware-based implementation as in this paper, a complicated algorithm is not easy to design so that only a simple algorithm is allowed for the selection of the better mode. As a result, the discard of the unselected mode may often substantially degrade the compression efficiency. To avoid such degradation, it is desirable to use an early termination scheme which avoids the discard of the unselected mode from the beginning. Instead, the unselected mode is discarded by comparing the cost of the selected mode with the estimated cost of unselected mode and terminating the execution of the unselected mode only when the estimated cost is greater than the cost of the selected mode.

In both schedules, the optimal I4 order as shown in Fig. 3 is adopted for the further speed-up for I14 execution. One change made by the new pipeline schedules in comparison with that in Fig. 2(b) is that the new schedules do not exclude the two I4 modes (modes 3 and 7) for blocks 2, 8, and 14. This is because of the observation that the exclusion of the two I4 modes (modes 3 and 7) for blocks 2, 8 and 14 as in Fig. 2(b) incurs a bitrate increase of 0.49%. The proposed schedules attempt to improve the compression efficiency by avoiding this performance loss and placing bubbles between up-right dependencies as well as between left dependencies. As a result, the new schedules place five bubbles instead of two as in Fig. 2(b).

## 4.3 Pipeline Schedule for MS-IPG

This subsection revisits the pipeline executions shown in Figs. 7 and 8 and explains issues when MS-IPG is employed. A decision whether to perform MS-IPG or the modified three-step is made right after R (at cycle 30 in Fig. 7). Since mode 0 is already performed at cycle 30 and mode 0 is always a candidate for the representative mode, mode 0 is always chosen as the representative mode.

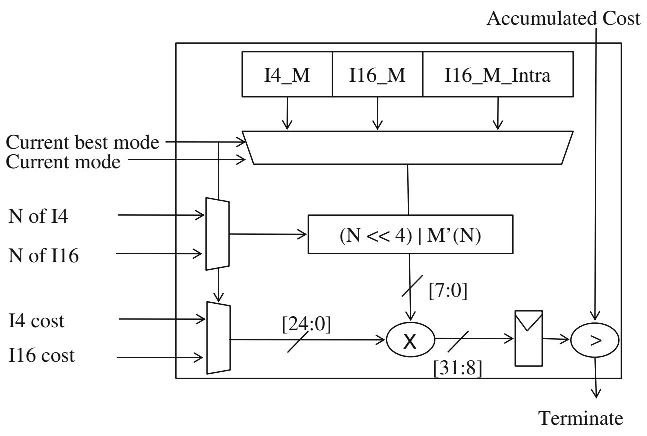


**Figure 11** Early termination hardware.

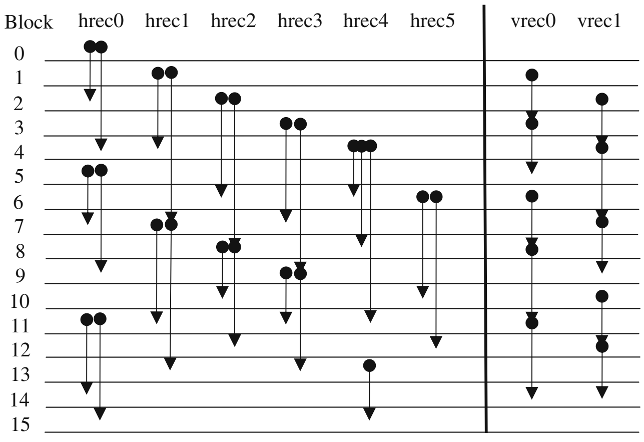


**Figure 12** Reorder buffer and lifetime of reconstructed pixels.

**Table 3** Gate counts of hardware modules.

| Components | Gate count |
|---|---|
| Boundary buffers | 11720 |
| Intra prediction | 4185 |
| Transform | 10150 |
| Mode decision | 14207 |
| FIFO | 4165 |
| Q and IQ | 14414 |
| Inverse transform | 7640 |
| Reconstruction controller | 4488 |
| Scheduler | 5792 |
| Early termination | 806 |
| Reorder buffers | 1745 |
| Total | 79313 |

For the up-right dependence as shown in Fig.7, 4 modes are processed before R. Therefore three bubbles may be generated after mode 0 if MS-IPG is selected after R. To reduce these bubbles, MS-IPG is performed in a two-step manner. In the first step, predictors of IPG 2 are compared at the beginning of Block 2. This is possible because predictors are available from the beginning of Block 2. If predictors of IPG 2 are all identical, the bubbles can be replaced by modes 4, 5, and 6. Otherwise, the first four modes (mode 0, 1, 2, and 4) are performed. In the second step predictors of IPG 3 are compared after R. Then, modes 3 and 7 are excluded if up and up-right predictors are all identical. If the predictors differ, the three-step algorithm is selected. Note that the predictors of IPG 1 are also compared in the first step. However, the bubble replacement scheme cannot be applied in this case. For IPG 1, modes 3 and 7 are the candidate modes for the replacement. However, these modes have to be done after R because they require up-right predictors. As no I4 mode can be scheduled in these bubbles, I16 blocks are performed during the bubbles.

MS-IPG can also generate bubbles in the normal pipeline schedule (Fig. 8). A block may depend on predictors from two blocks ahead. For example, reconstructed pixels of Block 2 are used as left predictors of Block4 as shown in Fig. 8. Even if Block 3 ends early by MS-IPG, Block 4 cannot start immediately after Block 3 because Block 4 has to wait for R of Block 2. Therefore, bubble generation is unavoidable between Block 3 and Block 4. This bubble is also filled by I16 blocks.

**Table 5** Performance improvement by identical pixel group.

| Video type | Number of excluded modes/MB |
|---|---|
| 1920×1080 | 15.137 |
| 1280×720 | 3.484 |
| 352×288 | 8.679 |

## 5 Hardware Implementation

The proposed intra prediction is implemented by programming with Hardware Description Language. The block diagram is shown in Fig 10. Similar to the organization in [3], the prediction phase adopts eight pixel parallel datapath which is double the width of the four pixel parallel datapath for the reconstruction phase. The parallelism of datapath is designed to alleviate the imbalance of prediction and reconstruction hardware utilization [3, 11, 12]. For I4 prediction, the predicted pixels and transformed residuals are buffered until all seven modes have been compared and the best mode is decided. The mode decision algorithm is modified to use DCT instead of the Hadamard transform to reuse the transform coefficients for quantization and to avoid regeneration of the best mode as proposed in [11].

Additional control logics are necessary to select the processing order of I16 modes and also to determine early termination which requires the computation of Eqs. 1, 2 and 3. Equation 1 or 3 is implemented with a 1 adder and a 1 multiplier. Equation 2 is implemented with a look-up table. As there are three kinds of early terminations, three look-up tables are constructed. The processing order in Fig. 3 requires additional buffers to store neighboring pixels of 4×4 blocks. For example, Block 4 needs pixels from Blocks 0, 1, and 2. Thus, the results of Blocks 0, 1, and 2 must be stored until the intra prediction of Block 4 begins. Reconstructed pixels are stored in different buffers (reorder buffers) in a manner that minimizes the number of buffers.

The controller for the pipeline schedule is slightly more complicated than that in other intra prediction implementations such as [3] because of the control overhead. The

**Table 4** Early termination rates for various image sizes.

| Video type | I16 selection (%) | I4 termination (%) | I16 termination (%) |
|---|---|---|---|
| 1920×1080 | 30.515 | 19.815 | 66.809 |
| 1280×720 | 9.479 | 4.607 | 75.823 |
| 352×288 | 9.659 | 6.055 | 72.656 |

**Table 6** Comparison with previous designs.

| Design feature | This work | [3] | [10] | [11] |
|---|---|---|---|---|
| Gate count | 79 K[a] | 94 K | 126 K | 85 K |
| Target size | HD1080p | HD1080p | HD720p | 720x480 |
| Maximum cycles | 464 cycles | 441 cycles | 624 cycles | 1060 cycles |
| Average cycles for | | | | |
| 1920×1080 | 334 cycles | 417 cycles | 475 cycles | 1017 cycles |
| 1280×720 | 342 cycles | 409 cycles | 587 cycles | 1002 cycles |
| 352×288 | 343 cycles | 407 cycles | 574 cycles | 1001 cycles |

[a] VLC is excluded in gate count for this work

controller is responsible for the switch of the pipeline schedule between Fig. 9(a) and (b), comparison of neighbor pixels for IPG, and control of early termination. The hardware for early termination is designed as shown in Fig. 11. The *M(0) x (16−N)* in Eq. 2 is denoted as *M'(N)* in Fig. 11. *M'(N)* is pre-computed for all *N* and stored in a table. I4_M, I16_M, I16_M_Intra are the three tables for I4 termination, I16 termination according to the I4 cost, I16 termination according to the I16 cost, respectively. The table, *N* and reference cost are selected according to the mode to be terminated (current mode) and the mode whose cost is used as the reference cost (current best mode). The remaining parts calculate the threshold for early termination. Figure 12 shows the reorder buffers (hrec0-5, vrec0-1) explained in the previous paragraph. The hrec stores bottommost pixels in a 4×4 block while vrec stores leftmost pixels. The numbers in the left are the block number in Fig. 3. The arrow represents lifetime of reconstructed pixels. To minimize the number of buffers, reconstructed pixels are stored as shown in Fig. 12.

The list of gate counts, synthesized using 0.13 μm technology is shown in Table 3. The total gate count is about 79 K. Note that VLC is not included in Table 3. Extra modules required to implement the proposed methods are Scheduler, Early termination, and Reorder Buffers. As shown in the table, the hardware overhead of the proposed intra prediction is less than eight thousand gates which are very small when compared with the total gate counts. Other hardware components including boundary buffers and a reconstruction controller are same as [3].

## 6 Comparison with Previous Works

Computation reduction of I4 and I16 by the proposed early termination schedule is shown in Table 4. The test sequences are Foreman, Mobile, Stefan, Weather for CIF-size (352×288), Parkrun, Shields, Stockholm, Mobcal for HD-size (1280×720), and Blue sky, Station, Rush hour, Tractor for Full-HD-size (1920×1080) videos. Four QPs, 16, 20, 24, and 28 are used and the results have been averaged over these four QPs. Table 4 shows that the I16 mode selection rate ranges from 9.5% to 30.5%. For the macroblocks with I4 selected as the first mode, the early termination rate of I16 varies from 66.8% to 75.8%. In the other case, the early termination rate of I4 varies from 4.6% to 19.8%. For CIF-size videos, a high rate of I16 early termination is achieved whereas the I16 early termination rate decreases as the video size increases. This is an expected result because smaller images are denser, and the prediction accuracy of I4 is higher for smaller images. As the video size increases, the selection rate of I16 over I4 increases. Furthermore, the prediction accuracy of I16 also increases resulting in the increase of I4 early termination rate. Therefore, I4 computation time is reduced for large videos. Experimental results also show that I16 selection decreases as QP decreases although they are not presented in this paper. This implies that the speed-up by I16 early termination increases for high quality videos, and the speed-up by I4 early termination increases for low quality videos. With the effective selection of the mode between I4 and I16 for early termination, the proposed early termination schedule in Fig. 9(a) and (b)

**Table 7** PSNR and bitrate changes.

| | PSNR (dB) | Bitrate (%) |
|---|---|---|
| Plane mode skip | −0.0071 | 0.0974 |
| DCT SATD | 0.0008 | 0.0331 |
| 3-Step | −0.0463 | 0.5547 |
| MS-IPG | 0.0231 | −0.3991 |
| Early terminations | −0.0324 | 0.5561 |
| Final result | −0.0619 | 0.8422 |

**Table 8** PSNR and bitrate comparison with [3].

| | PSNR (dB) | | Bitrate (%) | |
|---|---|---|---|---|
| | This work | [3] | This work | [3] |
| 1920×1080 | −0.0577 | −0.0446 | 1.2179 | 0.9063 |
| 1280×720 | −0.0505 | −0.0450 | 0.5526 | 0.5174 |
| 352×288 | −0.0774 | −0.0665 | 0.7559 | 0.6318 |

always achieves speed-up for various video quality or video sizes.

Table 5 shows the speed up achieved by taking advantage of MS-IPG presented in Section 3.5. The second column shows the average number of the excluded modes per a macroblock. Large images include a large number of pixels with identical values and fifteen I4 modes are excluded on average in 1920×1080 size videos. The number of predictors with identical values increases as QP increases although the results are not presented in this paper. This is because pixels with similar magnitudes may become the same values after quantization with large QP. In these experiments, HD-size videos achieve less speed up than CIF-size videos. This is because HD-size test videos used in this paper include a number of detailed objects, making their complexity almost as large as that of CIF-size videos.

Comparisons with previous designs are provided in Table 6. The architectures in [3], [10] and [11] require the maximum clock cycles of 441, 624 and 1060, respectively whereas the proposed design requires 464 cycles. As the execution time of the proposed hardware reduces significantly because of early termination and the mode selection according to identical predictors, the average cycles for various videos are smaller than the maximum cycle. For Full-HD size videos, the proposed design requires 334 cycles on average which is only 80% of the required cycles in [3].

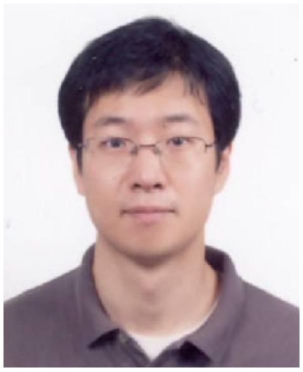
The PSNR and bit rate changes of the proposed design compared to H.264/AVC JM8.6 [13] are shown in Table 7. The results obtained from three video sizes are averaged. The performance drop is due to the aggregate influences by (1) plane mode skip, (2) DCT based SATD mode decision, (3) 3-step algorithm, (4) and early termination of I4 and I16. Compared to [3], PSNR is degraded by optimization (4). However, this degradation is compensated partly by the mode selection based on MS-IPG. The PSNR is enhanced because the mode selection by IPG sometimes substitutes the 3-step algorithm. Note that the mode selection by MS-IPG is lossless whereas that by the 3-step algorithm is lossy. The proposed design suffers 0.0619 dB PSNR loss, and 0.842% bit rate increase. Table 8 compares the PSNR and bit rate changes with the previous work [3]. As shown in the table, the difference is negligible.

## 7 Conclusion

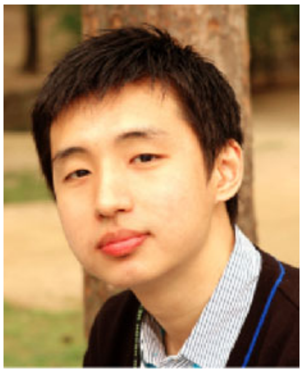
About 20% of the execution cycles for H.264 intra prediction are saved by the proposed pipeline schedule, early termination, and the mode selection based on IPG. Experimental results show that the proposed schedule with early termination is effective for various video sizes and quality. The mode selection based on IPG also provides substantial computation saving in large videos with large QPs. In spite of the significant reduction in computation time, PSNR drop is 0.0619 dB and the bit rate increase is less than 0.842%. Although this paper is mainly for a specific hardware, the proposed methodology can be applied to a wide range of platforms.

## References

1. *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification* (ITU-T Rec. H.264/ISO/IEC 11496-10 AVC), Mar. 2003.
2. Puri, A., Chen, X., & Luthra, A. (2004). Video coding using the H.264/MPEG-4 AVC compression standard. *Signal Processing: Image Communication, 19*, 793–849.
3. Lin, Y. K., Ku, C. W., Li, D. W., & Chang, T. S. (2009). A 140-MHz 94 K Gates HD1080p 30-Frames/s Intra-Only profile H.264 encoder. In *IEEE Trans Circuits and Syst Video Technol, 19*(3), Mar.
4. Pan, F., Lin, X., Rahardja, S., et al. (2005). Fast mode decision algorithm for intraprediction in H.264/AVC video coding. *IEEE Transactions on Circuits and Systems for Video Technology, 15* (7), 813–822.
5. Tsai, A. C., Wang, J. F., Lin, W. G., & Yang, J. F. (2007). A simple and robust direction detection algorithm for fast H.264 intra prediction. In *Proc. IEEE Int. Conf. Multimedia and Expo*, (pp. 1587–1590), Beijing, China, Jul.
6. Wei, Z., Li, H., & Ngan, K. N. (2007). An efficient intra mode selection algorithm for H.264 based on fast edge classification. In *Proc IEEE Int Symp Circuits and Syst* (pp. 3630–3633). New Orleans, La, USA, May.
7. Kun, Z., Chun, Y., Qiang, L., & Yuzhou, Z. (2007). A fast block type decision method for H.264/AVC intra prediction. In *Proc. Int. Conf. Advanced Communication Technology* (vol. 1, pp. 673–676). Gangwon-Do, Korea, Februrary.
8. Song, J. B., Li, B., Li, W., & Jiang, L. (2006). A novel fast intra prediction algorithm applied in H.264/AVC. In *Proc. Int. Conf. Signal Process.* (vol. 1, pp. 16–20). Beijing, China, November.
9. Yang, C. L., Lai-Man, P., & Lam, W. H. (2004). A fast H.264 intra prediction algorithm using macroblock properties. In *Proc. IEEE Int. Conf. Image Process*, (vol. 1, pp. 461–464). Singapore, October.
10. Jung, J. S., Jin, G. H., & Lee, H.-J. (2008). Early termination and pipelining for hardware implementation of fast H.264 intra prediction targeting mobile HD applications. *EURASIP Journal on Advances in Signal Process*, volume, Article ID 542735.
11. Huang, Y. W., Hsieh, B. Y., Chen, T. C., & Chen, L. G. (2005). Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder. *IEEE Transactions on Circuits and Systems for Video Technology, 15*(3), 378–401.
12. Cheng, C. C., Ku, C. W., & Chang, T. S. (2006). A 1280×720 pixels 30 frames/s H.264/MPEG-4 AVC intra encoder. In *Proc IEEE Int Symp on Circuits and Syst*, May.
13. *Joint Video Team Reference Software* JM8.6.



**Jin-Su Jung** received the M.S. degrees in The School of Electronic and Electrical Engineering from Inha University, Incheon, Korea, in 2003. He is currently working toward the Ph.D. degree in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea. His research interests are in the area of computer architecture and SoC design for multimedia applications.



**Young-Joon Jo** received the B.S. degree in Electrical Engineering from Seoul National University, Korea, in 2009. He is now a Ph.D. student in Electrical and Computer Engineering at Purdue University. His research interests include programming models, compilers and runtimes for optimizing and parallelizing irregular programs.

**Hyuk-Jae Lee** received the B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Korea, in 1987 and 1989, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Purdue University at West Lafayette, Indiana, in 1996. From 1998 to 2001, he worked at the Sever and Workstation Chipset Division of Intel Corporation in Hillsboro, Oregon as a senior component design engineer. From 1996 to 1998, he was on the faculty of the Department of Computer Science of Louisiana Tech University at Ruston, Louisiana. In 2001, he joined the School of Electrical Engineering and Computer Science at Seoul National University, Korea, where he is currently working as an Associate Processor. He is a founder of Mamurian Design, Inc., a fabless SoC design house for mobile multimedia applications. His research interests are in the areas of computer architecture and SoC design for multimedia applications.