135-MHz 258-K Gates VLSI Design for All-Intra H.264/AVC Scalable Video Encoder

Gwo-Long Li, Tzu-Yu Chen, Meng-Wei Shen, Meng-Hsun Wen, and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*— To satisfy the video application diversities, an extension of H.264/advanced video coding (AVC), called scalable video coding (SVC), is designed to provide multiple demanded video data via a single video encoder. However, constructed on the fundamental of H.264/AVC, the complexity of SVC is much higher than that of H.264/AVC. In this paper, a VLSI design for all-intra scalable video encoder is proposed to aim at efficient scalable video encoding. First, the memory bandwidth requirements for several encoding methods are analyzed to find out the best encoding method which can achieve best tradeoff between internal memory usage and external memory access. Afterward, an all-intra SVC encoder combined with several advanced techniques, including fast intra prediction algorithm, efficient syntax element encoding approach in context-adaptive variable-length coding, and hardware-efficient techniques, are implemented in a macroblock (MB)-level pipeline to increase data throughput. Implementation results demonstrate that our proposed SVC encoder can process more than 594-k MBs per second, which is equivalent to the summation of 60 high-definition, 1080-p, SD 480-p, and common intermediate format frames under 135-MHz working frequency. The proposed design consumes 258-K gate counts when synthesized by 90-nm CMOS technology.**

***Index Terms*— All-intra, scalable video coding (SVC), VLSI architecture design.**

I. Introduction

RECENTLY, with the prosperity of the Internet video, digital television, and portable devices, the demands for digital video have become more and more diverse. To deal with these diversified video applications, scalable video coding (SVC), the latest video coding standard inherited from the state-of-the-art H.264/advanced video coding (AVC), is formed to provide different scalabilities (temporal, spatial, and quality) in a single bitstream [1]. Fig. 1 shows an SVC encoder architecture with two spatial layers. To generate the scalable bitstream, the input images are first downsampled to lower spatial resolution and encoded by H.264/AVC compatible video encoder. Afterward, the higher spatial resolution images are encoded by H.264/AVC encoder with advanced interlayer prediction techniques to fully utilize the relationship between consecutive spatial layers and consequently improve the coding performance. In addition, the quality and temporal scalability are achieved in each spatial layer by the coarse granular scalability (CGS) and hierarchical B structure approaches, respectively. Finally, all generated bitstreams corresponding to different quality scalabilities are grouped into single SVC bitstream. However, in addition to the primitive coding complexities of H.264/AVC, the extra scalabilities of SVC also contribute significant computational complexity and memory requirement in hardware realization.

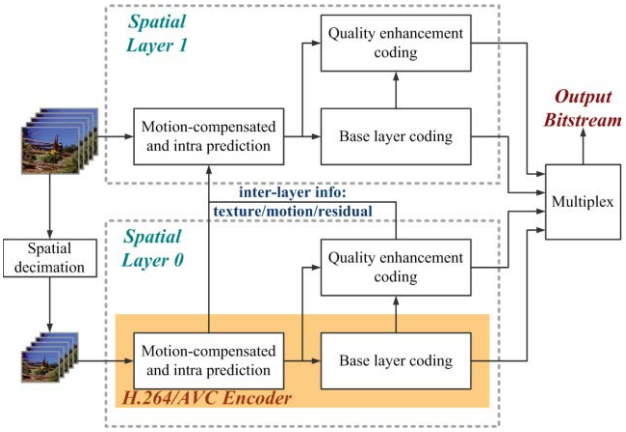


Fig. 1. Architecture of an SVC encoder with two spatial layers.

This paper focuses on the all-intra SVC encoder that provides the following features:

1) H.264/AVC scalable high 10 intra profile;
2) at most three spatial layers from quarter common intermediate format (QCIF) to high-definition (HD) 1080 p, CIF + SD 480 p + HD 1080 p at 60 f/s (equivalent to 4096 × 2160 at 51 f/s) for high visual quality application;
3) at most three quality layers for any quantization parameter (QP) value setting;
4) 8 × 8 intra prediction and transform size.

The major challenges for such design are the high throughput and related interlayer dependency that cause complex scheduling and data access. To solve the above problems, we first analyze the effect of different coding methods in spatial scalability by formulating the requirement of internal memory storage and external memory access. Based on

Manuscript received June 1, 2011; revised October 28, 2011 and February 17, 2012; accepted March 3, 2012. Date of publication April 3, 2012; date of current version March 18, 2013.

G.-L. Li is with the Industrial Technology Research Institute, Hsinchu 30074, Taiwan (e-mail: glli@itri.org.tw).

T.-Y. Chen and M.-H. Wen are with PixArt Imaging Inc., Hsinchu 30074, Taiwan (e-mail: tychen@dragons.ee.nctu.edu.tw; mhwen@dragons.ee.nctu.edu.tw).

M.-W. Shen and T.-S. Chang are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 30074, Taiwan (e-mail: mwshen@dragons.ee.nctu.edu.tw; tschang@dragons.ee.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2012.2190536

the results of the analysis, the best tradeoff is selected as the basis of our all-intra SVC encoder architecture design. Furthermore, to meet the high-throughput requirement, a fast intra prediction algorithm, parallel encoding architectures, and fast context-adaptive variable-length coding (CAVLC) entropy coding method are proposed to realize our SVC encoder. Implementation results show that our design can provide the capability of processing 60 frames per second for 1080-p, 480-p, and CIF resolution video sequences when running at 135-MHz operating frequency.

The rest of this paper is organized as follows. In Section II, the system-level memory analysis for different all-intra SVC encoding methods is presented in detail to quantify the internal memory usage and external memory access for the purpose of selecting the memory-efficient encoding method. Section III introduces our proposed fast intra prediction algorithm in detail. The complete introduction for our SVC encoder design in terms of algorithmic and architectural level is shown in Section IV. Several simulation results and implementation outcomes are exhibited in Section V to demonstrate the performance of our SVC encoder design. Finally, the conclusions are given in Section VI.

II. Analysis of All-Intra SVC Encoder

SVC supports three scalabilities: spatial, temporal, and quality scalability. To support spatial scalability, SVC adopts pyramid coding structure in which the frame resolution of each spatial layer is different from the other layers. As a result, such coding structure results in the high correlation between spatial layers, and thus motivating the adoption of interlayer predictions [1] to fully utilize the similarities between spatial layers. However, the adoption of interlayer prediction also complicates the hardware design. To analyze the complexity, this paper first defines three encoding methods called frame-, row-, and macroblock (MB)-level encoding methods, and proposes several theoretical analyses in the following subsections to obtain the internal memory usage and external memory access requirement of different encoding flows. Afterward, the coding method which achieves best tradeoff between internal memory usage and external memory access requirement will be selected as the best coding method in our SVC encoder design. In addition, all constants listed in the following analysis tables are derived by the approximation and simulation approach.

A. Frame-Level Coding Method

In this coding method, the spatial enhancement layer would be encoded after the entire reference layer is encoded as shown in Fig. 2. To examine the internal storage under this coding method, Table I shows the internal storage requirement for this method. The term *pre-deblock* (PDB) data refers to the pixels needed for the deblocking filter. In the table, PDB data are assumed to be stored in internal memory and they dominate the internal memory cost, especially for high-resolution applications. If PDB data are stored into external memory, the corresponding storage size in internal memory would be a constant. The additional costs from PDB data and other terms in external bandwidth are listed in Table II.

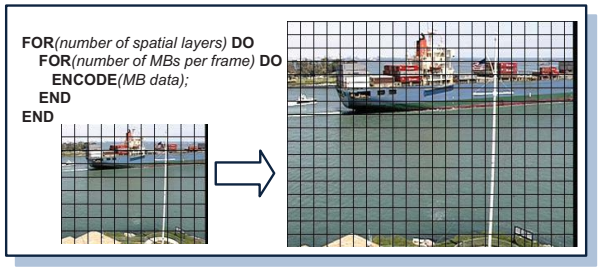


Fig. 2. Frame-level encoding method: graphical illustration and the pseudo code.

TABLE I
Internal Memory Usage for the Frame-Level Coding Method

Name	Bytes
PDB coefficient (row and column data)	$256 \times m \times (W_{\mathrm{mb,max}} + 1)$
CGS coefficient	$288 \times m$
MB type	$5.5 \times (W_{\mathrm{mb,max}} + 2)$
Other neighboring info*	$5.5 \times (W_{\mathrm{mb,max}} + 9)$

$W_{\mathrm{mb,max}}$: the width of the largest frame in the unit of MB; m: number of quality layers.
*includes coded block pattern, nonzero transform coefficient flags for the deblocking process, and neighboring intra prediction modes.

TABLE II
External Memory Access for the Frame-Level Coding Method

Name	Bytes
Input pixel	$1.5 \times \sum_{i=0}^{n-1} W_i \times H_i$
Reconstructed pixel	$1.5 \times m \times \sum_{i=0}^{n-1} W_i \times H_i$
Output bitstream*	$0.5 \times m \times \sum_{i=0}^{n-1} W_i \times H_i$
Pre-deblocking coefficient	$256 \times m \times \sum_{i=0}^{n-1} W_{mb,i} \times (H_{mb,i} - 1)$

n: number of spatial layers; m: number of quality layers; W_i: the frame width of the ith spatial layer; H_i: the frame height of the ith spatial layer; $W_{\mathrm{mb},i}$: the frame width of the ith spatial layer in the unit of MB; $H_{\mathrm{mb},i}$: the frame height of the ith spatial layer in the unit of MB.
*the output bitstream of each quality layer is conservatively estimated as 10% of the original input.

B. Row-Level Method

The second method is called the "row-level" method, which is shown in Fig. 3. In this coding method, a row of MBs would be encoded after a corresponding row of MBs in the reference layer has been encoded. If the PDB data are stored in internal memory, the corresponding internal memory cost will be changed as listed in Table III. The external data bandwidth requirement of the row-level method is the same as that of the frame-level method.

C. MB-Level Method

The third one is called the "MB-level" method, which is shown in Fig. 4. In this coding method, an MB in the reference layer would be encoded first, and then the corresponding MBs

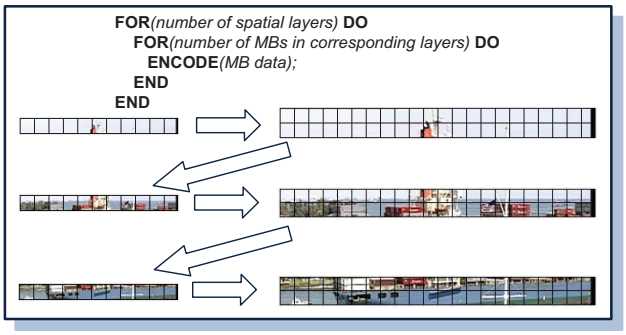


Fig. 3. Row-level method: graphical illustration and pseudo code.

TABLE III
INTERNAL MEMORY USAGE UNDER THE ROW-LEVEL CODING METHOD

Name	Bytes
PDB coefficient (row data)	$256 \times m \times \sum_{i=0}^{n-1} (W_{\mathrm{mb},i} + 1)$
CGS coefficient	$288 \times m$
mb_type	$5.5 \times \sum_{i=0}^{n-1} (W_{\mathrm{mb},i} + 2)$
Other neighboring info*	$\sum_{i=0}^{n-1} (5.5 \times W_{\mathrm{mb},i} + 9)$

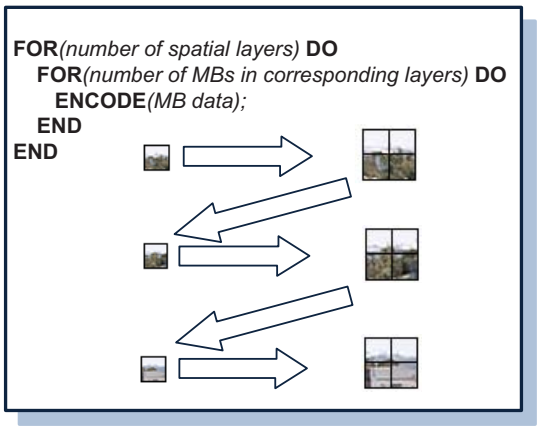


Fig. 4. MB-level method: graphical illustration and pseudo code.

in the enhancement layer will be encoded. The internal storage requirements and external memory access remain the same as in the row-level method.

D. Analysis Results

All the previous analyses are applied for coding conditions of three spatial layers (CIF, SD 480 p, and HD 1080 p) and three quality layers with YUV 420 format in 30-Hz frame rate as exhibited in Fig. 5. In our simulation, the memory access latency will not be considered since the data access of intra prediction mode is very simple and regular. Table IV summarizes the analysis result comparisons and improvements of different coding methods. From this table, we can observe that the external memory accesses of all coding methods are the same. However, for the internal storage, the frame-level coding method has less internal storage requirements

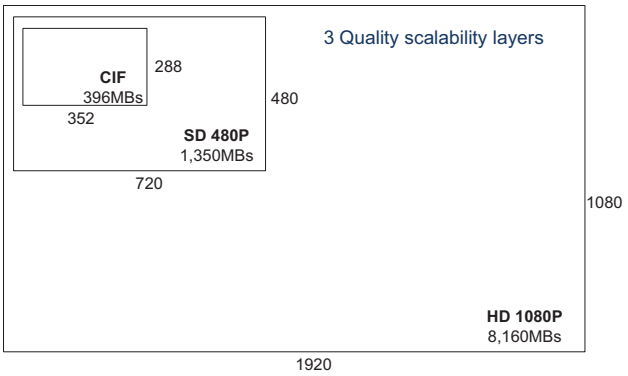


Fig. 5. Proposed input specification.

TABLE IV
MEMORY DEMAND COMPARISON OF EACH CODING METHOD UNDER THE GIVEN SPECIFICATION

Memory usage	(A) PDB data stored in internal memory		(B) PDB data stored in external memory	
	Internal storage (KB)	External access (MB)	Internal storage (KB)	External access (MB)
Frame-level	95.13	48.138	2.20	48.28
Row-level	149.03	48.138	3.10	48.28
MB-level	149.03	48.138	3.10	48.28

than the other two coding methods. This situation can be explained as follows. In MB- and row-level coding methods, some information such as the prediction data of neighboring MBs have to be stored temporary inside the internal memory due to spatial layer switching. Although the prediction data of neighboring MBs can be stored into external memory, the frequent external memory access will lead to the overall video coding system performance drop. Therefore, it is more reasonable to store such data inside internal memory. However, for the frame-level coding method, the problem of internal memory space caused by spatial layer switching would no longer exist and would thus result in less internal memory space requirements. Therefore, from this table, we understand that the frame-level coding method is the best one because it can achieve best tradeoff between internal memory usage and external memory access. Thus, we choose the frame-level coding method for our design based on the cost and memory bandwidth tradeoff. A much more complete analysis for SVC encoder with temporal scalability can be found in our previous work [2].

III. PROPOSED FAST INTRA MODE DECISION ALGORITHM

Intra prediction explores the redundancy of highly correlated intensity values between neighboring pixels. Fig. 6 shows the intra prediction modes adopted in H.264/AVC, in which the intra 16×16 supports four prediction modes [shown in Fig. 6(a)] and intra 8×8 and 4×4 provide nine prediction modes [shown in Fig. 6(b) and (c)]. Traditionally, intra prediction uses exhaustive search, which means all

TABLE V
INTENSITY CALCULATION FOR DIFFERENT CATEGORIES

DC	AC	V	H
$I_{DC} = \lvert t_{00} \rvert$	$I_{AC} = \sum_{i=0}^{m} \sum_{j=0}^{n} \lvert t_{ij} \rvert_{(i=j)si}$	$I_V = \sum_{i=1}^{n} \lvert t_{0i} \rvert$	$I_H = \sum_{i=1}^{n} \lvert t_{i0} \rvert$
	$m = n = 3$ for 4×4 $m = n = 7$ for 8×8	$n = 3$ for 4×4 $n = 5$ for 8×8	$n = 3$ for 4×4 $n = 5$ for 8×8
D	**DV**	**DH**	
$I_{DV} = \sum_{i=1}^{n} \lvert t_{ii} \rvert$	$I_{DV} = \sum_{i=1}^{m} \sum_{j=2}^{n} \lvert t_{ij} \rvert_{ij2}$	$I_{DH} = \sum_{i=2}^{m} \sum_{j=1}^{n} \lvert t_{ij} \rvert_{ij1}$	
$n = 3$ for 4×4 $n = 5$ for 8×8	$m = 2, n = 3$ for 4×4 $m = 3, n = 4$ for 8×8	$m = 3, n = 2$ for 4×4 $m = 4, n = 3$ for 8×8	

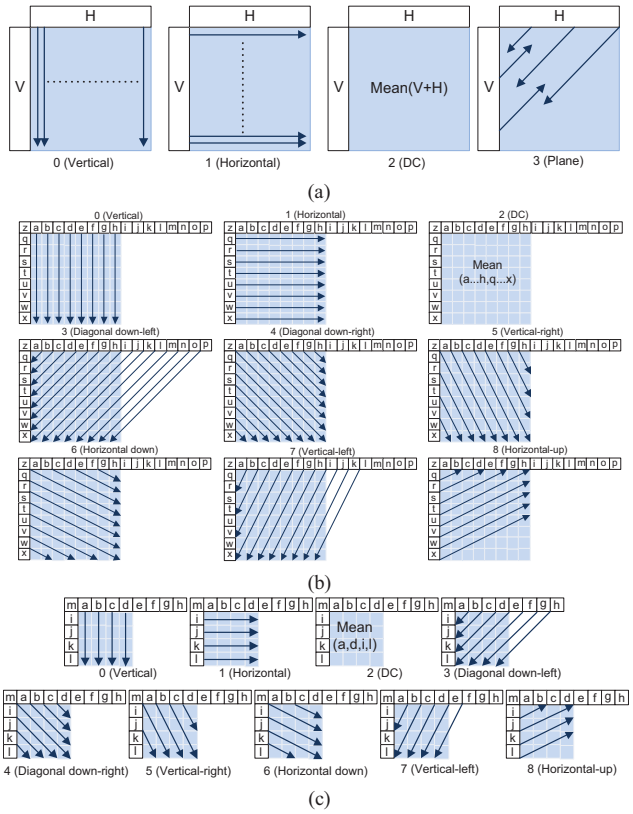


Fig. 6. Illustration of intra prediction modes in H.264. (a) Intra 16×16. (b) Intra 8 × 8. (c) Intra 4 × 4.

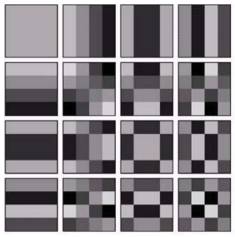


Fig. 7. Basis pattern of DCT.

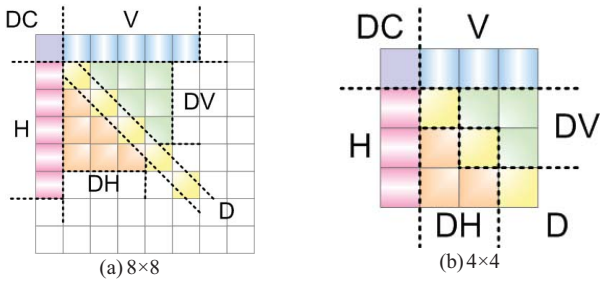


Fig. 8. Categories of coefficients for different transform sizes. (a) 8 × 8. (b) 4 × 4.

available modes in each kind of block size would be examined one by one. However, full search is not time efficient. Thus, many fast algorithms have been developed to attain acceptable performance with checking the fewest number of mode candidates. They used the approaches such as early termination [3], [4], selected candidates based on the result of each decision step [5], [6], or edge information [7], [8]. Besides, intra prediction can be also combined with the transform process to use the transform results for mode decision [9] or block size selection [10].

In this section, a fast intra prediction algorithm is proposed based on the algorithm in [11] with modified edge detection in transform domain, DC-dominant condition, and fast intra 8×8 prediction to achieve higher system throughput and better quality than the other approaches.

A. Proposed Fast Intra Prediction Algorithm With Transform Domain Edge Detection (TraDED)

1) Intra 4 × 4 *and* 8 × 8 *Predictions:* Considering the basis patterns of discrete cosine transform (DCT) in Fig. 7, we can classify these patterns into six categories: DC, vertical (V), horizontal (H), diagonal (D), diagonal–vertical (DV), and diagonal–horizontal (DH), as shown in Fig. 8. In the proposed algorithm, the intensities of each category are calculated as shown in Table V. In this table, t indicates the transform coefficient.

Fig. 9 shows the mode decision algorithm which is decided from the magnitudes of these intensities. For DC mode, it must be turned on if the block is on the boundary in case there are no mode candidates. Otherwise, the DC mode would be turned off if the ratio (I_{DC}/I_{AC}) is smaller than a given threshold, which means the DC intensity is weak. On the other hand, in AC mode decision, if the ratio (I_{DC}/I_{AC}) is larger than another given threshold, DC intensity dominates the block edge direction, and most of the AC modes would be turned off except vertical or horizontal mode. Otherwise, AC candidate modes are chosen by the comparison of intensities

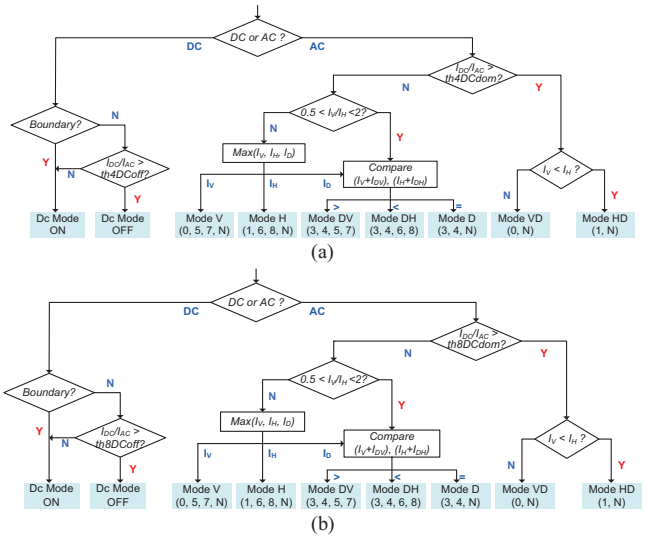


Fig. 9. Proposed TraDED algorithm. (a) 4 × 4. (b) 8 × 8.

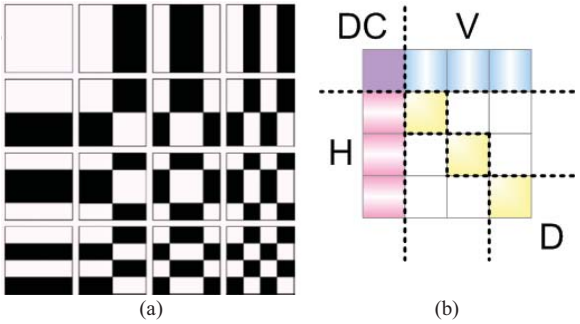


Fig. 10. 4 × 4 DHT. (a) Basis patterns. (b) Categories of coefficients.

I_V, I_H, I_D, I_{DV}, and I_{DH}. In this case, if I_V is the maximum in $\{I_V, I_H, I_D\}$ and the ratio (I_V/I_H) is larger than 2, then vertical intensity dominates. Hence, vertical mode (mode 0) and two adjacent modes (modes 5 and 7) would be chosen as the mode candidates. Similarly, if I_H is the maximum and the ratio (I_V/I_H) is less than 0.5, horizontal mode (mode 1) and two adjacent modes (modes 6 and 8) are selected. Otherwise, neither vertical nor horizontal edge is regarded as the major direction, and diagonal modes (modes 3 and 4) would be considered as the mode candidates according to the relation between $(I_V + I_{\mathrm{DV}})$ and $(I_H + I_{\mathrm{DH}})$.

Besides, the most probable mode (mode N: a specific prediction mode in intra prediction) would be selected for intra prediction if the number of AC mode candidates is smaller than four. Our proposal not only increases the coding efficiency but also makes the total number of mode candidates to be restricted within five.

2) Intra 16×16 Prediction: In the prediction of 16×16 block size, pixels tend to be highly correlated with each other. It means that DC coefficients would be much larger than AC coefficients. Hence, DC coefficients in Hadamard transform (DHT) can be taken into consideration in edge detection. The basis patterns of 4×4 DHT are shown in Fig. 10(a), which can be divided into four categories for the four available modes as shown in Fig. 10(b). Their intensities are calculated by the equations listed in Table VI.

TABLE VI
INTENSITY CALCULATION FOR DIFFERENT CATEGORIES

DC	V	H
$I_{\mathrm{DC}} = \lvert t_{00}\rvert$	$I_V = \sum_{i=1}^{3} \lvert t_{0i}\rvert$	$I_H = \sum_{i=1}^{3} \lvert t_{i0}\rvert$
D	**AC**	
$I_D = \sum_{i=1}^{3} \lvert t_{ii}\rvert$	$I_{\mathrm{AC}} = \sum_{i=0}^{3}\sum_{j=0}^{3} \lvert t_{ij}\rvert_{(i=j)si}$	

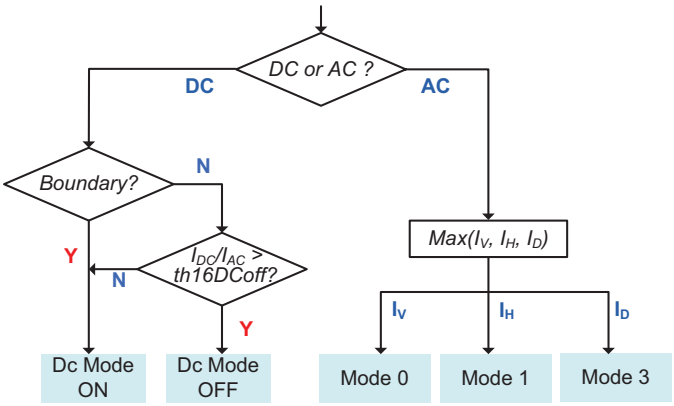


Fig. 11. Flowchart of intra 16 × 16 mode decision.

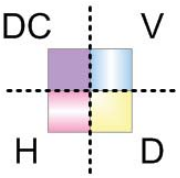


Fig. 12. Categories of coefficients in DHT 2 × 2.

Fig. 11 shows the mode decision of intra 16 × 16 prediction. The DC mode is always on unless that the current block is not at the boundary and the ratio $(I_{\mathrm{DC}}/I_{\mathrm{AC}})$ is smaller than a given threshold *th16DCoff*. For AC, the mode candidate is selected from the maximum intensity among $\{I_V, I_H, I_D\}$. In worst case, two modes would be chosen by our algorithm.

3) Intra Chroma Prediction: For chroma prediction, we simplify the intensity calculation as the sum of two components (Cb and Cr) with the corresponding coefficients as shown in Fig. 12. The calculation is listed below

$$I_{\mathrm{DC}} = |t00|_{\mathrm{Cb}} + |t00|_{\mathrm{Cr}}, \quad I_V = |t01|_{\mathrm{Cb}} + |t01|_{\mathrm{Cr}} \quad (1)$$

$$I_H = |t10|_{\mathrm{Cb}} + |t10|_{\mathrm{Cr}}, \quad I_D = |t11|_{\mathrm{Cb}} + |t11|_{\mathrm{Cr}}. \quad (2)$$

Fig. 13 shows the flowchart of mode decision. DC mode is always on to ensure there are two mode candidates in the nonboundary MBs. In addition, other intensities are compared together to decide another mode candidate from the largest intensity (I_H for mode 1, I_V for mode 2, and I_D for mode 3).

B. Simulation Results

In the following simulation, 100 I-frames of eight sequences with different resolutions and four QPs of 8, 16, 24, and 32 are undertested. The thresholds of *th4DCdom (set to 16)*,

TABLE VII
RATE DISTORTION PERFORMANCE COMPARISON FOR OUR PROPOSED TRADED WITH JM 12.4

		QP = 8		QP = 16	
		ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)
QCIF	Akiyo	−0.02	1.42	−0.02	1.64
	Foreman	+0.02	0.26	+0.03	0.97
CIF	Mobile	+0.03	0.51	+0.05	0.77
	Stefan	+0.02	0.55	+0.03	1.03
720 p	Shields	+0.05	0.70	+0.06	1.13
	Stockholm	+0.07	0.80	+0.07	1.38
1080 p	Sunflower	−0.01	0.16	+0.01	1.71
	Tractor	+0.01	0.32	+0.04	1.03
Average		**+0.02**	**0.59**	**+0.03**	**1.21**
		QP = 24		**QP = 32**	
		ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)
QCIF	Akiyo	−0.04	1.50	−0.18	1.20
	Foreman	+0.06	1.64	+0.32	2.53
CIF	Mobile	+0.11	1.20	+0.10	2.14
	Stefan	+0.09	1.56	+0.05	0.99
720 p	Shields	−0.02	1.93	−0.13	3.93
	Stockholm	−0.06	1.25	−0.20	4.15
1080 p	Sunflower	−0.08	4.06	−0.33	3.04
	Tractor	+0.02	1.08	−0.41	3.01
Average		**+0.01**	**1.78**	**−0.10**	**2.62**

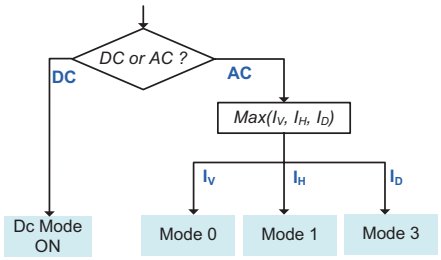


Fig. 13. Flowchart of intra chroma mode decision.

th4DCoff (set to 2), *th8DCdom (set to 64)*, *th8DCoff (set to 2)*, and *th16DCoff (set to 2)* are derived either from [11] or experimentally with considering the simplicity of hardware implementation. The comparison of bitrate (BR) and peak signal-to-noise ratio (PSNR) between the original result in JM12.4 [12] (with the low-complexity mode) and the proposed algorithm is listed in Table VII. Furthermore, the average mode candidates for each sequence are summarized in Table VIII.

From the results we understand that the performance of *TraDED* pretty approaches the performance of the full search algorithm in JM12.4. For the rate distortion comparison, our proposed *TraDED* algorithm can achieve 0.02-dB PSNR increase with 1.54% negligible BR increase. However, for low BR points, especially for high-resolution sequences, the performance of the proposed algorithm is not as good as expected. This phenomenon can be explained below. For low-BR (also called low-quality) video coding, the larger QPs are usually used to result in less BR requirements. However, the larger QP also significantly degrades the video quality and thus blurs the object boundary and texture. As a result, the blurred object boundary and texture lead to the prediction failure of our proposed algorithm as well as other fast prediction algorithms. However, such rate distortion performance losses are negligible in low BR video coding. For the average mode candidate reductions, we can find that our proposed *TraDED* can save at least 48.43% and 53.28% of intra mode checking for chroma and luma components, respectively, when compared to JM 12.4. As a result, through our proposed fast intra prediction algorithm, more than half intra mode candidates can be reduced, and this increases the hardware throughput.

TABLE VIII
AVERAGE MODE CANDIDATES IN ENCODING

Algo. / Seq.	JM12.4 (Full search)				Proposed (*TraDED*)			
	4×4	8×8	16×16	Chro.	4×4	8×8	16×16	Chro.
Akiyo	8.72	8.45	3.61	3.61	3.60	3.96	1.67	1.92
Foreman	8.72	8.45	3.61	3.61	3.74	4.02	1.83	1.94
Mobile	8.86	8.72	3.80	3.80	3.70	3.83	1.62	1.95
Stefan	8.86	8.72	3.80	3.80	3.48	3.86	1.83	1.96
Shields	8.92	8.85	3.89	3.89	4.27	4.10	1.67	1.98
Stockholm	8.92	8.85	3.89	3.89	4.16	4.08	1.82	1.99
Sunflower	8.97	8.94	3.95	3.95	4.17	4.49	1.92	1.99
Tractor	8.97	8.94	3.95	3.95	4.20	4.24	1.89	2.00
Average	**8.87**	**8.74**	**3.81**	**3.81**	**3.92**	**4.07**	**1.78**	**1.97**
Savings (%)					**55.85**	**53.40**	**53.28**	**48.43**

TABLE IX
RATE DISTORTION PERFORMANCE COMPARISON BETWEEN OUR PROPOSED TRADED AND [11] (REFER TO JM 12.4)

QP		8				24			
		[11]		*TraDED*		[11]		*TraDED*	
		ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)	ΔPSNR (dB)	ΔBR (%)
QCIF	Akiyo	−0.16	0.27	−0.02	1.42	−0.04	0.75	−0.04	1.50
	Foreman	−0.15	0.15	+0.02	0.26	−0.04	0.74	+0.06	1.64
CIF	Mobile	−0.20	0.36	+0.03	0.51	−0.07	0.74	+0.11	1.20
	Stefan	−0.18	0.30	+0.02	0.55	−0.07	0.81	+0.09	1.56
	Stockholm	−0.23	0.20	+0.07	0.80	−0.05	0.95	−0.06	1.25
1080 p	Tractor	−0.16	0.04	+0.01	0.32	−0.03	1.13	+0.02	1.08
Average		−0.19	0.23	+0.02	0.65	−0.05	0.86	+0.02	1.45

Tables IX and X show the comparison of rate distortion performance and average number of candidate modes between our proposed *TraDED* and [11], respectively. For rate distortion comparison, our proposed algorithm can aim at 0.03-dB PSNR increase with only 1.07% additional BRoverhead when compared to JM 12.4, while [11] results in 0.28-dB PSNR degradation with 0.54% BR increase. In addition, 15.63% and 9.35% mode reduction can be achieved for luma and chroma intra prediction when compared to [11]. However, although our proposed fast intra prediction algorithm has a little bit of rate distortion losses when compared to [11], the overall reduced intra prediction modes lead our proposed SVC encoder to be able to finish the encoding of an MB as fast as possible, which is helpful for high-throughput SVC encoder.

TABLE X
COMPARISON BETWEEN [11] AND TRA DED FOR AVERAGE MODE CANDIDATES

		[11]			*TraDED*			
		4×4	**16×16**	**Chro.**	**4×4**	**8×8**	**16×16**	**Chro.**
QCIF	*Coastguard*	4.11	2.00	2.16	3.87	3.62	1.69	1.89
	Container	4.33	2.00	2.09	3.52	3.75	1.73	1.91
	Foreman	4.67	2.00	2.18	3.74	4.02	1.83	1.96
	News	4.40	2.00	2.12	3.82	3.81	1.55	1.95
	Silent	4.66	2.00	2.08	4.10	4.05	1.80	1.95
	Average	**4.43**	**2.00**	**2.13**	**3.81**	**3.85**	**1.72**	**1.93**
CIF	Mobile	4.43	2.00	2.16	3.70	3.83	1.62	1.95
	Paris	4.33	2.00	2.13	3.85	3.80	1.66	1.94
	Stefan	4.57	2.00	2.12	3.48	3.86	1.83	1.96
	Tempete	4.37	2.00	2.17	3.83	3.70	1.41	1.95
	Average	**4.43**	**2.00**	**2.15**	**3.72**	**3.80**	**1.63**	**1.95**

TABLE XI
COMPARISON WITH OTHER CAVLC ENCODER DESIGNS

Design features	**[13]**	**[14]**	**[15]**	**This paper**
Technology	0.18 μm	90 nm	0.18 μm	90 nm
Max frequency	125 MHz	200 MHz	140 MHz	135 MHz
Gate count	12125	66559	12276	8454
Avg. cycles/MB	266.5 (QP = 10)	254 (QP = NA)	126.6 (QP = 40)	293 (QP = 10) 273 (QP = 12) 80 (QP = 40)

IV. ALL-INTRA SVC ENCODER ARCHITECTURE DESIGN

The implementation of an SVC intra encoder with the adoption of previous analysis and fast algorithm is introduced in this section. Our design flow is stated as follows. At the beginning, we define our target specification which specifies how many spatial and quality layers and what kind of features that our SVC encoder will support. After defining the target specification, we analyze how many clock cycles that can be used to finish the encoding of an MB under our defined operating frequency. Afterward, we design and optimize individual components to make sure that our designed SVC encoder can meet the constraint of available clock cycles. The detailed design flow is described in the following subsections.

A. System Analysis

The target system specification is an SVC encoder working under 135-MHz clock frequency with three quality layers, three spatial layers (CIF, SD 480 p, and HD 1080 p), and 60 frames per second. To fulfill such throughput, we adopt the "frame-parallel" encoding approach which encodes two frames at the same time with doubled hardware resources. Furthermore, we adopt mixed pixel parallelism for the high throughput and low area cost. The required parallelism is derived as follows. In the above assumption, the total number of processed MBs is

$$396 + 1350 + 8160 = 9906. \tag{3}$$

TABLE XII
LIST OF GATE COUNT FOR [16] AND OUR PROPOSED DESIGN

Module	**[16]***	**This Paper**
Transform	**19 868**	**45 265**
- for 4×4 and DC	N/A	14 558
- for 4×4 and 8×8	N/A	30 710
Cost calculation and mode decision	**12 923**	**25 974**
- for 4×4, 16×16, and chroma	N/A	11 981
- for 4×4, 8×8, and 16×16	N/A	13 993
Prediction and residual generator	**6646**	**35 938**
- for 4×4, 16×16, and chroma modes	N/A	5047
- for 4×4, 8×8, and 16×16 modes	N/A	14 828
- DC mode	N/A	2172
- plane mode	Not support	11 338
- residual	N/A	2553
Quantization and reconstructed path	**70 104**	**73 558**
- forward/inverse quantization	31 908	44 885
- inverse transform and reconstruction	38 196	28 673
Entropy coding (CAVLC)	**7474**	**29 642**
Deblocking	**Not support**	**36 892**
Quality enhancement	**Not support**	**10 352**
Total	**141 549**	**257 618**

*synthesizing in 145-MHz, 0.13-μm technology.

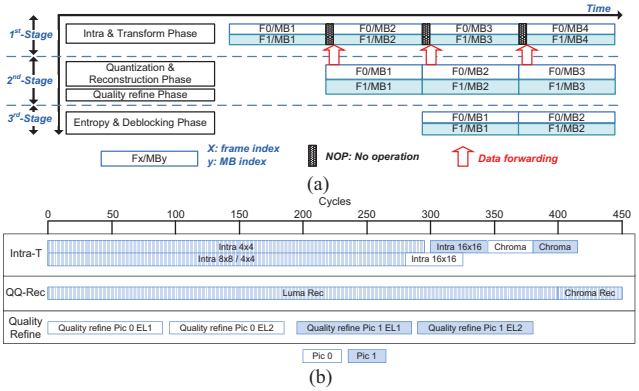


Fig. 14. Scheduling diagram of our proposed SVC encoder. (a) MB by MB-based scheduling diagram. (b) Detailed scheduling diagram of an MB.

For 135-MHz working frequency, the available operation time per MB would be

$$135\text{M}/((60/2) \times 9906) = 454 \text{ cycles}. \tag{4}$$

In (4), the operation of "/2" means that two MBs have been processed simultaneously due to the adoption of frame-parallel coding approach. And then, just considering the prediction data under fast algorithm, the total pixel counts per MB would be

$$16 \times 16 \times (5 + 5 + 2) + 8 \times 8 \times 2 = 3328. \tag{5}$$

Thus, the best parallelism can be calculated by

$$\frac{3328}{454} = 7.33 \Rightarrow 8 \text{ pixels (for double hardware resource)}. \tag{6}$$

To reduce design complexity, two equivalent components can be replaced by two components with different functions. Therefore, the parallelism of this paper is set to mixed

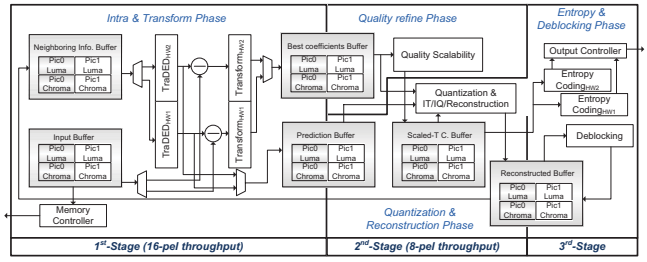


Fig. 15. Proposed all-intra SVC encoder architecture design with three pipeline stages.

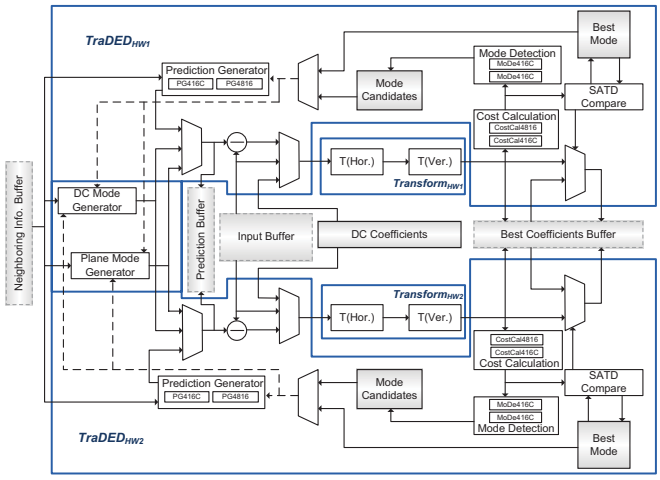


Fig. 16. Architecture design of first pipeline stage.

16/8 pixels. To do so, the intra prediction and transform have doubled hardware with 16-pixel throughput, the quantization/inverse quantization and reconstruction have single hardware with 8-pixel throughput, and the quality refine has single hardware with 8-pixel throughput.

B. Interlaced Frame-Parallel Scheduling

The overall scheduling of our SVC encoder adopts the interlaced frame parallel three-stage scheduling method as illustrated in Fig. 14. Most of the computations are interlaced for two MBs from parallel frames to increase the hardware utilization. At the beginning, the two first MBs of the two frames will enter into *Intra and Transform Phase* for intra prediction and transformation. Afterward, the two second MBs of the two frames will be inputted into *Intra and Transform Phase* and the two first MBs go into *Quantization and Reconstruction Phase*. However, due to the data dependency between the first stage and second stage, the NOP operations will be held in the first stage temporarily for the two second MBs in order to wait for the reconstruction of the two first MBs. Once the necessary pixels of the two first MBs have been successfully reconstructed, the reconstructed pixels, accompanied with a valid signal, will be forwarded to the first stage from the second stage to start up the intra prediction and transformation process for the two second MBs. Finally, the two first MBs go into the third stage to generate the SVC bitstream and remove the blocking effects. At the same time, the two second and third MBs of the two frames will, respectively, enter into the second stage and first stage for encoding.

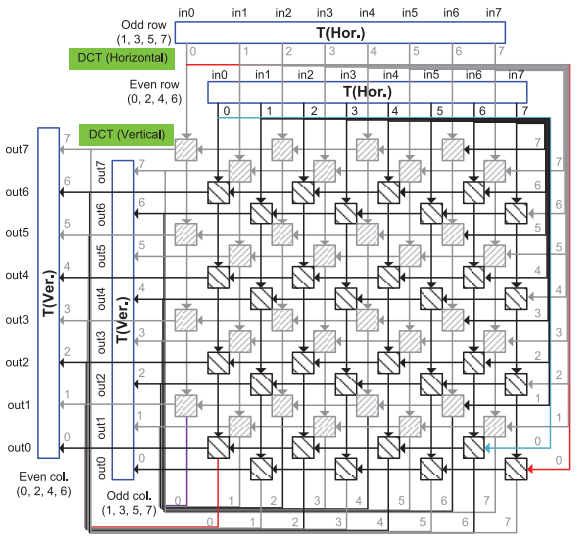


Fig. 17. Architecture of module *Tran48*.

C. Three Pipeline Stage Architecture Design

Fig. 15 shows our proposed all-intra SVC encoder architecture design with three pipeline stages. In our architecture, all operations are grouped into four phases including *Intra and Transform Phase*, *Quality Refine Phase*, *Quantization and Reconstruction Phase*, and *Entropy and Deblocking Phase*. First, in *Intra and Transform Phase*, we process the input by the proposed *TraDED* intra prediction to obtain the best prediction mode and apply the transform operation to the residuals to obtain the transform coefficients for the following encoding usage. The function of quality scalability can be done in *Quality Refine Phase*. Then, the data are further quantized and reconstructed in the *Quantization and Reconstruction Phase*, along with the quality layer computation. Finally, the *Entropy and Deblocking Phase* executes the entropy coding for all coefficients that are needed to be encoded into bitstream and removes the blocking effect of reconstructed pixels.

1) Architecture Design of First Stage: Fig. 16 shows the detailed architecture design of the first pipeline stage. It basically follows the *TraDED* algorithm to compute cost first for mode selection and then compute the intra prediction output. This stage is formed by two parallel *TraDED* and *Transform* modules so that two MBs are processed at the same time. In *TraDED* design, two buffers called *Best Mode* and *Mode Candidates* are used to store, respectively, the best intra prediction mode and mode information temporally resulted during the intra prediction process. The *DC Mode Generator* and *Plane Mode Generator* are shared by two *TraDED*s to generate the predictions of DC and plane mode in intra prediction. The other components will be introduced in the following subsections.

a) Parallel transform modules with dynamic task assignment: Since *TraDED* algorithm, mentioned in Section III, is adopted in the proposed design, a transform module not only changes the domain of residues but also plays a role in basis construction for mode candidate decision. Thus, the two transform modules in the encoder are not identical since more 4×4 DCT computations are needed for mode decision. So

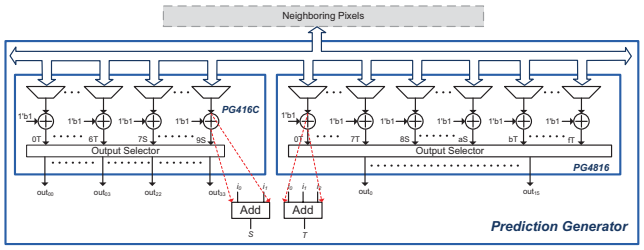


Fig. 18. Architecture of prediction generator.

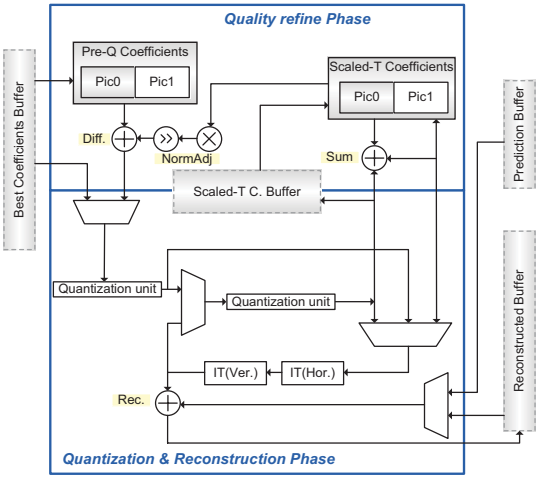


Fig. 19. Architecture design of second pipeline stage.

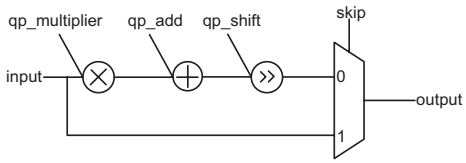


Fig. 20. Architecture of a quantization unit.

we design two types, one (*Tran4DC*) for 4×4 DCT and DHT processing, and another one (*Tran48*) deals with 4×4 and 8×8 DCT. The module *Tran4DC* is similar to the traditional 4×4 DCT butterfly architecture [17]. The module *Tran48* extends our previous even/odd parallelism approach [17] to support 16-pixel parallelism without extra cost in the transpose registers. As shown in Fig. 17, four 1-D 8-point transform units are included (two for horizontal and two for vertical operations) and connected to an 8×8 block-size coefficient register. This transform unit can also support 1-D 4×4 transform for two rows by using a multiplexer for better hardware utilization.

The task assignment of the two transform modules is dynamically determined. In the transformation stage, our design simultaneously executes the transformation for 8×8 and 4×4 block sizes through the *Tran4DC* and *Tran48* modules, respectively. Once all the 8×8 subblocks have been done, the remaining unprocessed subblocks of 4×4 would be immediately assigned to the *Trans48* module for transformation to increase the hardware utilization.

b) Mode decision modules: Two *Mode Decision* modules (each module is composed of two submodules *MoDe416C* and *MoDe48*) read intensity information from *Cost Calculation* modules. The concept of fast intra prediction algorithm *TraDED* is implemented in these two modules. Module *MoDe416C* processes intra 4×4, intra 16×16, and intra chroma predictions, module *MoDe48* deals with intra 4×4 and intra 8×8 predictions. The output length of a mode decision module is 9 bit, where each of them represents the enable signal for the related mode. This 9-bit-length data will be decoded by mode candidate generator and it outputs one mode candidate for each cycle.

c) Intra prediction generator modules: The selected modes by *TraDED* are computed in this part. This part allocates all prediction modes into four modules to generate intra prediction information according to their utilization rate, one for DC mode (*DC Mode Generator*), one for plane mode (*Plane Mode Generator*), and two *Prediction Generator* modules (each *Prediction Generator* module is formed by two submodules of *PG416C* and *PG4816* as shown in Fig. 18). The two *Prediction Generator* modules are not identical due to unequal mode usage. In module *PG416C*, predictions for intra 4×4, intra 16×16, and intra chroma mode are generated. Furthermore, the more complicated module *PG4816* not only supports 4×4, 8×8, and intra 16×16 prediction data generation but also deals with neighboring data prefiltering in intra 8×8 prediction. The prediction generator decomposes all prediction generations into basic two or three input adder types (S or T in Fig. 18) so that they can be selected by regular selector for different mode computations. The DC and plane mode are separated into a dedicated module since it is not resource-efficient to complete DC or plane mode prediction within one cycle as for other modes. For DC mode, module *DC Mode Generator* calculates the mean values of neighboring pixels for all different block sizes since the utilization of DC mode is not frequent (around 20%).

2) Architecture Design of Second Stage: Fig. 19 exhibits the architecture design of our second pipeline stage in which the operations of quantization, reconstruction, and quality scalability (*Quality Refine Phase*) are executed. In forward and inverse quantization, the parallelism is set to 8 pixels instead of 16 pixels to save resource. The architecture of a quantization unit (note: inverse quantization has the same structure) is shown in Fig. 20. In addition, the QP-dependent multipliers are implemented by lookup tables for low cost consideration. The quantization module also supports quality refinement processes to share the hardware cost since most of the operations in this coding technique are very similar to the quantization process except the normalization and coefficient subtractions as shown in Fig. 21. For reconstruction, the reconstructing path includes inverse transform module and a set of adders. Furthermore, the inverse transform module integrates two transform units and transposition registers to support all kinds of inverse transforms with 8-pixel throughput.

3) Architecture Design of Third Stage: In this pipeline stage, two modules called *Deblocking* and *Entropy Coding* are implemented. The *Deblocking* module is used to remove the blocking effects of reconstructed pixels for further prediction usage. To support high-throughput requirements of our SVC encoder, we adopt [18], [19] to construct our *Deblocking* mod-

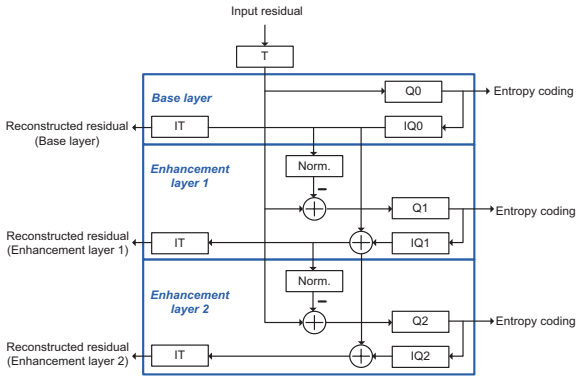


Fig. 21. Block diagram of quality enhancement coding in SVC.

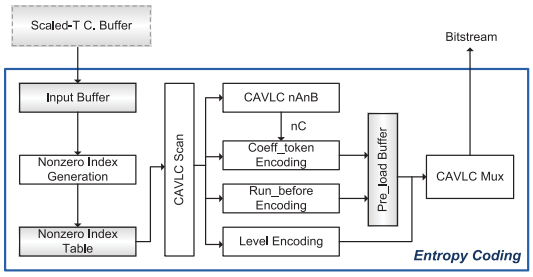


Fig. 22. Architecture design of entropy coding.

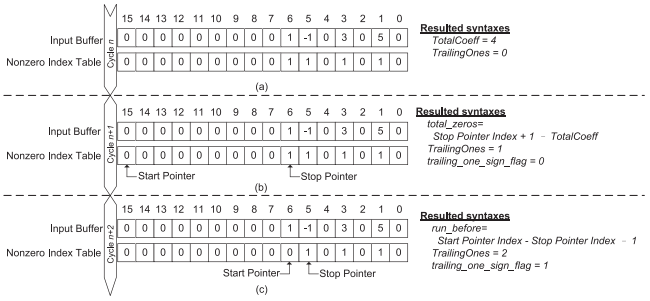


Fig. 23. (a) Example of input buffer and nonzero index table contents. (b) Nonzero index table with start pointer and stop pointer. (c) Illustration of nonzero index table updating.

ule. To achieve high-throughput entropy coding, an improved CAVLC design [20] is adopted in our design.

Fig. 22 shows the architecture design of our *Entropy Coding* module which processes 4×4 subblock at a time. The 16 coefficients are stored in an *Input Buffer*, and the *Nonzero Index Generation* module generates a *Nonzero Index Table* which marks nonzero coefficients as "1" and zero coefficients as "0" for all coefficients. After the *Nonzero Index Table* has been generated, the "0" and "1" inside the *Nonzero Index Table* will be used to speed up the CAVLC encoding process. By using the *Nonzero Index Table*, we can directly encode the nonzero coefficients and skip all the zero coefficients by *CAVLC Scan*. That is, we only spend cycles on encoding nonzero coefficients and save cycles on encoding zero coefficients. Besides, *CAVLC nAnB*, *Coeff_token Encoding*, *Level Encoding*, and *Run_before Encoding* are used for encode CAVLC symbols. Finally, we use *CAVLC Mux* to select what CAVLC symbols will be encoded into bitstream.

Fig. 23 exhibits a cycle-by-cycle example to demonstrate how the *CAVLC Scan* module operates and what syntaxes would be generated with the aid of *Nonzero Index Table*. The coding flow of our proposed fast CAVLC is described as follows. First, in cycle n, the 16 coefficients are loaded into *Input Buffer* in zigzag scan order and we check which coefficients are nonzero at the same time. If the coefficient is nonzero, we mark "1" to the corresponding *Nonzero Index Table* as shown in Fig. 23(a). After that, we use an adder to calculate the total number of 1s in the *Nonzero Index Table* and the sum is the syntax element of *TotalCoeff*. In cycle $n+1$, two pointers called *Start Pointer* and *Stop Pointer*, generated and controlled by *CAVLC Scan*, are used to scan *Nonzero Index Table* as shown in Fig. 23(b). The *Start Pointer* index is initiated to 15 and the *Stop Pointer* index indicates the first "1" in the *Nonzero Index Table*. After obtaining the position of the stop pointer index and *TotalCoeff*, we can calculate *total_zeros*, which is one of the CAVLC encoding symbols, by the following equation:

$$\text{total\_zeros} = \text{Stop Pointer Index} + 1 - \text{Total Coeff} \quad (7)$$

where *Stop Pointer Index* is the index that the *Stop Pointer* indicated. At the cycle $n+2$, we update the *Nonzero Index Table* as shown in Fig. 23(c). The bit at the position of the *Stop Pointer* is set to "0" and *Start Pointer* moves to the new zero bit. Moreover, the *Stop Pointer* points to the next nonzero coefficient. Afterward, the *run_before*, which is one of the CAVLC encoding symbols, can be calculated by the following equation:

$$\text{run\_before} = \text{Start Pointer Index} - \text{Stop Pointer Index} - 1 \quad (8)$$

where *Start Pointer Index* stands for the index that the *Start Pointer* pointed. At the following cycles, we repeatedly update the *Start Pointer* and *Stop Pointer* in the same way. Besides, we check whether the coefficient pointed by the *Stop Pointer* is a trailing one or not. If the coefficient pointed by the *Stop Pointer* is a trailing one, we add one to *TrailingOnes* and set the value of *trailing_one_sign_flag*, which is one of the CAVLC encoding symbols. After all the *trailing_one_sign_flag* are encoded, the rest of the coefficients pointed by the *Stop Pointer* are *level*s. After each *level* is encoded, we start to encode another CAVLC symbol, *total_zeros*. Finally, we encode the last CAVLC symbol, *run_before*. However, the traditional method only encodes one *run_before* at each cycle. The main difference between our fast CAVLC encoding and traditional method is that we use an additional buffer called *Pre_load Buffer* to store each corresponding *run_before* when we process the levels temporally. In the final step, we write all *run_before* information which have been encoded previously and temporally stored in *Pre_load Buffer* into bitstream to finish the CAVLC encoding for a given block. Therefore, through the adoption of *Pre_load Buffer*, we only spend one cycle to encode *run_before* after *total_zeros* is encoded.

Fig. 24 shows the timing schedule for encoding a 4×4 subblock. The x means the number of trailing 1s and y means the number of levels. Besides, y equals *TotalCoeff* minus the

TABLE XIII
COMPARISON BETWEEN THE DESIGNS IN PREVIOUS WORKS AND THIS PAPER

Design feature	[17]	[16]*	[21]	[22]	[23]	[24]	[25]	This paper
CMOS technology	0.13 μm	0.13 μm	0.13 μm	90 nm	0.18 μm	90 nm	90 nm	90 nm
Max operation frequency	140 MHz	145 MHz	114 MHz	152 MHz	150 MHz	210 MHz	166 MHz	135 MHz
Gate count	94.7 K	141 K	265.3 K	91 K	201.8 K	414.28 K	2079 k***	258 K
On-chip memory usage	1 KB	1 KB	8.4 KB	2.9 KB	5.39 KB	8.99 KB	81.7 KB	24 KB
Standard	Intra H.264	Intra H.264	Intra H.264	Intra H.264	Intra H.264	H.264, SVC, and MVC	H.264 and SVC	All-intra H.264 and SVC
Number of quality layers						4		3
Frame rate	30 f/s	30 f/s	30 f/s	30 f/s	61 f/s	30 f/s**	30 f/s	60 f/s
Max target resolution	HD 1080p	HD 1080p	HD 1080p	HD 1080p	HD 1080p	HD 1080p**	HD 1080p	HD 1080p + SD 480p +CIF
Throughput (MB/s)	244 800	244 800	244 800	244 800	497 760	244 800**	244 800	594 360

*: intra part only; **: only for SVC configuration; ***:logic gates.

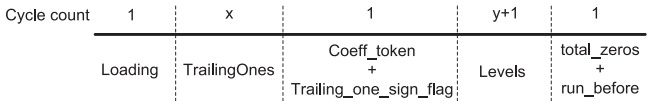


Fig. 24. Timing schedule for encoding a 4 × 4 block.

number of trailing 1s. Therefore, the total cycle counts for encoding a 4×4 subblock is *TotalCoeff* + 4.

Table XI shows the comparisons of implementation results for different CAVLC designs. From this table, we can observe that our design has lowest gate count when compared to that of other works. For average cycles per MB, although the design in [15] has the lowest average cycles per MB, these results were only derived at the very high QP condition that has less residual to be encoded by the CAVLC encoder.

V. IMPLEMENTATION RESULTS

The proposed variable-pixel parallel SVC intra encoder with *TraDED* fast algorithm is designed by Verilog-HDL and implemented by using 90-nm CMOS technology. The critical path of this design is in 8×8 transform because the complex architecture and the hardware utilization of our proposed SVC encoder is 70.5% on average. Table XII shows the gate count and comparisons with the intra part of our previous H.264/AVC encoder [16]. The hardware increase of the intra SVC encoder mainly comes from the doubled hardware requirements such as *Intra Prediction*, *Transform*, and *Entropy Coding* modules for higher throughput.

Table XIII demonstrates the detailed comparisons of our SVC encoder when compared to previous works with intra-only H.264/AVC [17], [16], [21]–[23]. Comparison with full SVC encoders [24], [25] is also included in Table XIII. Compared to other H.264/AVC intra-only designs, the proposed design can support both video coding standards, H.264/AVC and SVC, by only adjusting some parameters to result in corresponding bitstreams. For the high visual experience applications, our proposed design can provide 60-Hz frame rate encoding capability. However, [17] and [16], [21], [22] only process 30 frames per second for 1080-p resolution. The gate count of our design is around 258 K, with 135-MHz working frequency, implemented in 90-nm 1P9M CMOS technology.

Compared with the SVC design in [24], the four quality layers are provided in a single spatial layer configuration in our design. For three spatial layer configuration, designs in [24] and [25] only support 16 CIF +4 CIF + CIF at 30-Hz frame rate, while our proposed design can support SVC standard with a target spec of three quality layers in three resolutions (CIF, SD 480 p, and HD 1080 p) with 60-Hz frame rate, which is strongly demanded to increase better visual experience.

VI. CONCLUSION

In this paper, a high-performance all-intra SVC encoder VLSI design was proposed. To support significant amount of coding operations of SVC, the system level memory analyses are introduced first in this paper to find out the best coding method which can achieve best tradeoff between internal memory usage and external memory requirements. Furthermore, several advanced techniques are proposed as well to improve the coding performance of the SVC encoder. Implementation results show that our proposed SVC encoder can process 60 frames per second with an image resolution of 1080 p + 480 p + CIF at the cost of 258-K synthesized gate count under the operating frequency of 135 MHz.

REFERENCES

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[2] T.-Y. Chen, G.-L. Li, and T.-S. Chang, "Memory analysis for H.264/AVC scalable extension encoder," in *Proc. IEEE Symp. Circuits Syst.*, Taipei, Taiwan, May 2009, pp. 361–364.

[3] B. Meng, O. C. Au, C.-W. Wong, and H.-K. Lam, "Efficient intra-prediction algorithm in H.264," in *Proc. IEEE Int. Conf. Signal Process.*, vol. 3. Sep. 2003, pp. 837–840.

[4] F. Fu, X. Lin, and L. Xu, "Fast intra prediction algorithm in H.264-AVC," in *Proc. IEEE Int. Conf. Signal Process.*, vol. 2. Aug. 2004, pp. 1191–1194.

[5] C.-C. Cheng and T.-S. Chang, "Fast three step intra prediction algorithm for 4×4 blocks in H.264," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2. May 2005, pp. 1509–1512.

[6] C.-H. Hsia, J.-S. Chiang, Y.-H. Wang, and T.-Y. Teng, "Fast intra prediction mode decision algorithm for H.264/AVC video coding standard," in *Proc. IEEE Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, vol. 2. Nov. 2007, pp. 535–538.

[7] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu, "Fast mode decision algorithm for intraprediction in H.264/AVC video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 813–822, Jul. 2005.

[8] J.-C. Wang, J.-F. Wang, J.-F. Yang, and J.-T. Chen, "A fast mode decision algorithm and its VLSI design for H.264/AVC intra-prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 10, pp. 1412–1422, Oct. 2007.

[9] Y.-C. Wei and J.-F. Yang, "Transformed-domain intra mode decision in H.264/AVC encoder," in *Proc. IEEE Region 10 Conf.*, Nov. 2006, pp. 1–4.

[10] T. Zhang, G. Tian, and S. Goto, "A frequency-based fast block type decision algorithm for intra prediction in H.264/AVC high profile," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Nov. 2008, pp. 1292–1295.

[11] H.-Y. Lin, K.-H. Wu, B.-D. Liu, and J.-F. Yang, "An efficient VLSI architecture for transform-based intra prediction in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 6, pp. 894–904, Jun. 2010.

[12] *H.264/MPEG-4 AVC Reference Software, JM 12.4* [Online]. Available: http://iphome.hhi.de/suehring/tml/

[13] T.-H. Tsai, S.-P. Chang, and T.-L. Fang, "Highly efficient CAVLC encoder for MPEG-4 AVC/H.264," *IET Circuits Devices Syst.*, vol. 3, no. 3, pp. 116–124, Jun. 2009.

[14] Y. Yi and B. C. Song, "High-speed CAVLC encoder for 1080p 60-Hz H.264 codec," *IEEE Signal Process. Lett.*, vol. 15, pp. 891–894, 2008.

[15] Y.-J. Kim, K.-Y. Wang, S.-S. Lee, B.-S. Kim, B.-K. Choi, and D.-J. Chung, "Implementation of high efficient CAVLC encoder for H.264/AVC," in *Proc. IEEE Int. Conf. Pervas. Comput., Signal Process. Appl.*, Sep. 2010, pp. 912–915.

[16] Y.-K. Lin, D.-W. Li, C.-C. Lin, T.-Y. Kuo, S.-J. Wu, W.-C. Tai, W.-C. Chang, and T.-S. Chang, "A 242 mW 10 mm^2 1080p H.264/AVC high-profile encoder chip," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2008, pp. 314–615.

[17] Y.-K. Lin, C.-W. Ku, D.-W. Li, and T.-S. Chang, "A 140-MHz 94 K gates HD1080p 30-Frames/s intra-only profile H.264 encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 3, pp. 432–436, Mar. 2009.

[18] F. Tobajas, G. M. Callico, P. A. Perez, V. de Armas, and R. Sarmiento, "An efficient double-filter hardware architecture for H.264/AVC deblocking filtering," *IEEE Trans. Consumer Electron.*, vol. 54, no. 1, pp. 131–139, Feb. 2008.

[19] K. Xu and C.-S. Choy, "A five-stage pipeline, 204 cycles/MB, single-port SRAM-based deblocking filter for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 3, pp. 363–374, Mar. 2008.

[20] M.-C. Tsai and T.-S. Chang, "High performance context adaptive variable length coding encoder for MPEG-4 AVC/H.264 video coding," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2006, pp. 586–589.

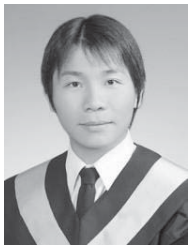
[21] H.-C. Kuo, L.-C. Wu, H.-T. Huang, S.-T. Hsu, and Y.-L. Lin, "A low-power high-performance H.264/AVC intra-frame encoder for 1080pHD video," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 925–938, Jun. 2011.

[22] J.-W. Chen, H.-C. Chang, J.-S. Wang, and J.-I. Guo, "A dynamic quality-adjustable H.264 intra coder," *IEEE Trans. Consumer Electron.*, vol. 57, no. 3, pp. 1203–1211, Aug. 2011.

[23] D. Claudio, Z. Bruno, T. Cristiano, S. Altamiro, B. Sergio, S. Felipe, P. Daniel, and A. Luciano, "A high throughput H.264/AVC intra-frame encoding loop architecture for HD1080p," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 579–582.

[24] T.-D. Chuang, P.-K. Tsung, P.-C. Lin, L.-M. Chang, T.-C. Ma, Y.-H. Chen, and L.-G. Chen, "A 59.5 mW scalable/multi-view video decoder chip for quad/3-D full HDTV and video streaming applications," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2010, pp. 330–331.

[25] Y.-H. Chen, T.-D. Chuang, Y.-J. Chen, C.-T. Li, C.-J. Hsu, S.-Y. Chien, and L.-G. Chen, "An H.264/AVC scalable extension and high profile HDTV 1080p encoder chip," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2008, pp. 104–105.



Gwo-Long Li received the B.S. degree from the Department of Computer Science and Information Engineering, Shu-Te University, Kaohsiung, Taiwan, the M.S. degree from the Department of Electrical Engineering, National Dong-Hwa University, Hualien, Taiwan, and the Ph.D. degree from the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan, in 2004, 2006, and 2011, respectively.

He is currently an Engineer with the Industrial Technology Research Institute, Hsinchu. His current research interests include video signal processing and its VLSI architecture designs.

Dr. Li was a recipient of the Excellent Master Thesis Award from the Institute of Information and Computer Machinery in 2006.

Tzu-Yu Chen received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2007 and 2009, respectively.

He joined the PixArt Imaging Inc., Hsinchu, in 2009. His current research interests include digital signal processing, scalable video coding, and associated VLSI architectures.

Meng-Wei Shen received the B.S. and M.S. degrees from the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan, in 2007 and 2009, respectively.

His current research interests include digital signal processing, scalable video coding, and associated VLSI architectures.

Meng-Hsun Wen received the B.S. and M.S. degrees in electrical engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2009 and 2011, respectively.

He joined the PixArt Imaging Inc., Hsinchu. His current research interests include H.264/advanced video coding and associated VLSI architecture designs.

Tian-Sheuan Chang (S'93–M'06–SM'07) received the B.S., M.S., and Ph.D. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 1999, respectively.

He is currently a Professor with the Department of Electronics Engineering, NCTU. From 2000 to 2004, he was a Deputy Manager with Global Unichip Corporation, Hsinchu. His current research interests include silicon intellectual properties, system-on-a-chip designs, VLSI processing, and computer architectures.