VLSI ARCHITECTURE OF HEVC INTRA PREDICTION FOR 8K UHDTV APPLICATIONS

Jianbin Zhou, Dajiang Zhou, Heming Sun, Satoshi Goto

Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Kitakyushu 808-0135, Japan.
E-mail: khshu@fuji.waseda.jp

ABSTRACT

This paper presents an efficient VLSI architecture of intra prediction for 8Kx4K HEVC decoder. It supports all 35 intra prediction modes and prediction sizes ranging from 4x4 to 64x64. This works proposed a Cyclic SRAM Banks based Parallel Reference Sample Fetching (CSB-PRSF), which guarantees enough reference samples for prediction and reduces the number of registers used for storing reference samples. To guarantee high throughput, 16 pixels are predicted by 4x4 Block Based Pipelining, and dependency between neighboring blocks is eliminated by Hybrid Data Forwarding and Block Reordering.

This architecture is synthesized using 90nm technology and the maximum working frequency is 469 MHz, with 72.1K gates area. Running at 397MHz, the architecture can support 4320p@120fps HEVC intra decoding, with full modes and full sizes.

Index Terms— HEVC decoder, intra prediction, VLSI architecture, 8K UDTV

1. INTRODUCTION

High Efficiency Video Coding (HEVC) [1][2] is the emerging video coding standard developed by Joint Collaborative Team on Video Coding (JCT-VC). It is aimed at 50% bit rate reduction compared with H.264 standard. Intra prediction plays an important role in HEVC, saving about 22%-36% bitrate. Meanwhile, the computational complexity is increased by some critical changes. Firstly, the maximum block size for intra prediction is 64x64 in HEVC instead of 16x16 in H.264. Secondly, the number of intra prediction modes is 35 in HEVC instead of 10 in H.264.

In HEVC, frames are divided into Coding Units (CU) and each root CU can be recursively divided into 1 or 4 smaller CUs. Each leaf CU will be processed by Prediction Units (PU) and Transform Units (TU). PU ranges from 64x64 to 4x4 while TU from 4x4 to 32x32. If a CU is encoded in intra mode, each TU corresponds to an intra prediction block with the corresponding PU's prediction mode. Therefore, the blocks size is from 4x4 to 32x32 and there're 35 prediction modes in intra prediction.

Several designs have already been proposed for HEVC intra prediction hardware design. Li [4] proposed VLSI architecture for 4x4 intra prediction in HEVC was proposed. Huang [5] proposed architecture for 4k Ultra HD HEVC decoder which has a low circuit area. The architecture proposed by Palomino [6] only supports several modes in 4x4 and 64x64 PU, and the throughput is low. The architecture proposed by Liu [7] supports all modes and PU sizes for 1080p@30fps HEVC encoder. Another work supports full HD is proposed by Zhou [8]. The works [6] [7] and [8] are for encoder instead of decoder. In the state of art, as there is no architecture for 8k Ultra HD HEVC decoder, this work would be the first one.

To design intra prediction architecture for an 8K Ultra HD HEVC decoder, there're 2 key challenges. The first one is that large number of reference pixels for prediction leads to a large area of control circuit and of registers. The second one is that data dependency between processed TU and unprocessed TU reduces the throughput. In this work, the issues above are solved by 3 techniques: 1) Cyclic SRAM Banks based Parallel Reference Sample Fetching(CSB-PRSF); 2) 4x4 Block Based Pipelining; 3) Hybrid Data Forwarding and Processing Order Rearranging.

The rest of the paper would be organized as follows. Section 2 will briefly introduces intra prediction in HEVC. Section 3 will show the details of our proposals. Section 4 will discuss the implementation result and Section 5 would be the conclusion.

2. INTRA PREDICTION IN HEVC

To process each TU by intra-prediction, firstly we get the reference samples, which come from the adjacent TUs. And then we calculate the predicted samples according to reference samples and intra prediction mode.

Before the reference samples are used for prediction, reference substitution and smoothing are applied on them in some situations. Reference samples are filtered in some modes if block size is 8x8, 16x16 and 32x32. There're 2 kinds of filtering method used in HEVC intra-prediction, three-tap [1 2 1]/4 FIR filtering and bilinear filtering. The bilinear filtering is used in 32x32 block when discontinuity is detected.

35 intra prediction modes can be classified into 3 classes, which are Planar, DC and angular mode. Each of

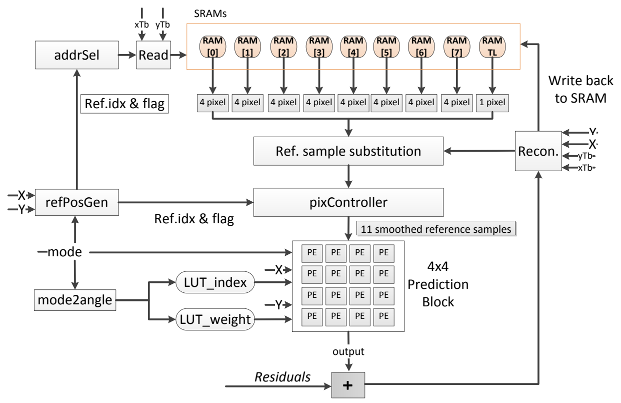


Fig. 1 The architecture of intra prediction engine

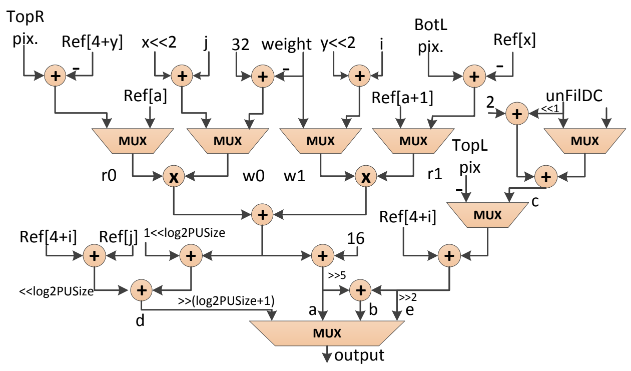


Fig. 2 The structure of one PE, designed for Planar, DC and angular prediction.

them has a formula shown below. For angular mode, each predicted sample is calculated according to the equation below:

$$\text{pred.} = ((32 - w) * ref[a] + w * ref[a+1] + 16) \gg 5 \quad (1)$$

For DC mode, the average of the top and left reference samples' value (dcVal) is used as the value of whole PU. Additionally, the first column, first row and the top-left pixel are filtered when PU's size is less than 32 in luma.

For Planar mode, each predicted sample is calculated according to the equation below:

$$\text{pred}[x,y] = \begin{pmatrix} (x+1) * \left(p[N][-1] - ref_{left}[y]\right) + \\ (y+1) * \left(p[-1][N] - ref_{above}[x]\right) + \\ N * \left(ref_{left}[y] + ref_{above}[x]\right) + N \end{pmatrix} \gg (Log_2 N + 1)$$

, where N is the size of PU, with $x, y = 0 \ \dots N-1$ (2)

3. PROPOSED VLSI ARCHITECTURE

For intra prediction of 8K HEVC decoding, high computational complexity and high throughput are 2 key challenges. According to our analysis, nearly 16 pixels processed per cycle must be required, if system's frequency is 397MHz and luma, chroma samples are processed in serial. To meet this requirement, three techniques are used in this work. The technique described in 3.1 and 3.3 guarantees that the throughput of 15 pixels/cycle can be met. The technique described in 3.2 guarantees necessary reference samples can be fetched for prediction in an efficient way.

3.1. 4x4 Block Based Pipelining

We choose 4x4 block as a prediction block (PB), because it has the same size as minimum TU. Large TUs can be predicted by processing the 4x4 PBs inside the TU one by one.

The architecture of intra prediction engine is shown in Fig. 1. For each 4x4 prediction block, our engine takes 4

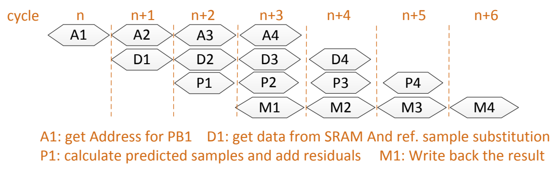
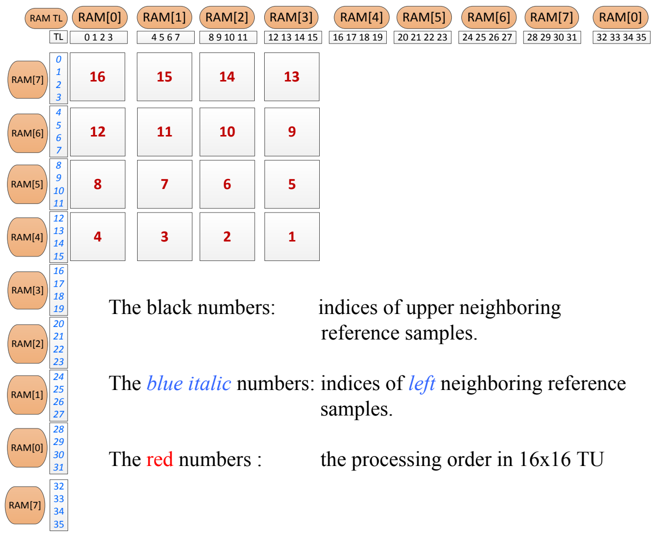

cycle
n
n+1
n+2
n+3
n+4
n+5
n+6
A1
A2
A3
A4
D1
D2
D3
D4
P1
P2
P3
P4
M1
M2
M3
M4
A1: get Address for PB1
D1: get data from SRAM And ref. sample substitution
P1: calculate predicted samples and add residuals
M1: Write back the result

Fig. 3 a 4-stage pipeline in Intra prediction engine

stages to process. The whole process is shown in Fig. 3. At the 1st stage, according to position of 4x4 block and prediction mode, the addresses are generated to get the samples for prediction; at 2nd stage, according to addresses, reference samples are read from SRAMs and reference sample substitution is done before being stored into registers; at 3rd stage, reference samples are filtered and used for calculating the 4x4 predicted samples. After that samples are added to the residuals; at 4th stage, the sum of predicted values and residuals are written back to SRAMs. Our engine takes 4 stages in pipeline to process a 4x4 prediction block. Each 4x4 prediction block is processed per cycle. For 4/8/16/32/64 PU, it requires 1/4/17/68/272 cycles to complete processing.

In the 4x4 prediction block we proposed, there're 16 Prediction Elements (PE) which process 16 samples in parallel per cycle. The detailed design of each PE is shown in Fig. 2. The inputs of each PE inside the 4x4 block are prediction mode, weight and reference index. We generate 2 LUTs to get the weight and reference index instead of using formula to calculate in hardware, in order to reduce the path delay. In each predictor, there're 2 multipliers, 10 adders, and 5 multiplexers. For the predictor in 1^{st} row and column, 4 additional adders and 2 multiplexers are needed for each predictor for filtering. "a" is the output of angular prediction mode; "d" is the output of Planar; "e" and "b" are the output for boundary samples smoothing in DC mode and horizontal, vertical modes correspondingly when TU is 4x4 to 16x16. The hardware used for boundary samples smoothing are only in the first row and first column in the prediction block. We make a transformation to the Planar mode's formula so that Planar mode and angular modes can share 2 multipliers. By this way 16*2 multipliers (5bit by 8bits) are saved.



The black numbers: indices of upper neighboring reference samples.

The *blue italic* numbers: indices of *left* neighboring reference samples.

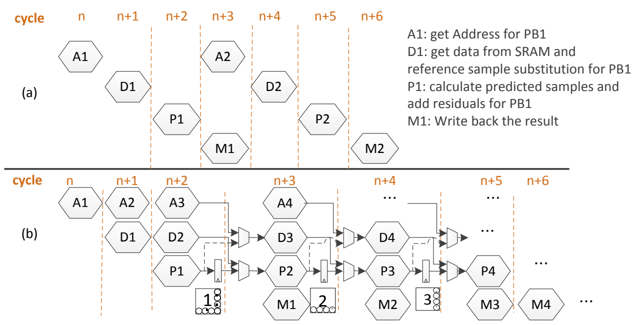
The red numbers : the processing order in 16x16 TU

Fig. 4 Arrangement of reference samples in 8 Cyclic SRAM Banks and processing order of 4x4 block in 16x16 TU

3.2. Cyclic SRAM Banks based Parallel Reference Sample Fetching

In some situations, reference samples substitution and smoothing are applied before it can be used for calculation. In most cases, 2 smoothed reference samples are necessary for calculating a predicted sample. We found that among all prediction modes and sizes of TU, 5~11 smoothed reference samples are needed for 4x4 block's prediction. There're three issues that makes it difficult to fetch reference samples from SRAMs. Firstly, among angular prediction mode 11 to 25 in 16, 32 TU, part of the reference samples are discontinuous. Secondly, Planar mode needs additional top-right and bottom left samples for prediction. Thirdly, [1 2 1] FIR filtering needs neighboring left and right samples to the specific sample to do filtering; while bilinear filtering needs 4 additional samples. These unsmoothed reference samples have to be fetched each cycle for later prediction. As a SRAM bank allows only one of its cell's data to be read by a specific address per cycle, collision may occur if some reference samples to be used are stored in the same SRAM bank but different cell.

To reduce the probability that collision happens, we developed a Parallel Reference Sample Fetching scheme based on 8 Cyclic SRAMs Banks to store the reference samples. The arrangement of reference samples store in SRAM banks is shown in Fig. 4. The width of each SRAM banks is 32 bits (4 samples). The cyclic order guarantees any neighboring 32 samples can be fetched in the same cycle, which reduces the chances of collision. As increment of SRAMs will increase the chip's area and power consumption, we found that using 8 SRAM banks is more suitable than any other numbers. The top reference samples are always stored in even address while the blue italic one in odd address. As SRAMs banks are in cyclic order, the address and position of reference sample in a specific SRAM cell can be calculated directly by shift and mod operation.



A1: get Address for PB1
D1: get data from SRAM and reference sample substitution for PB1
P1: calculate predicted samples and add residuals for PB1
M1: Write back the result

Fig. 5 (a) stall 2 cycles to solve dependency problem in 4x4 TU (b) Use Data forwarding to solve dependency problem in 4,8 TU

We develop a look-up table (LUT) based Reference Sample Position Generator (RSPG) to indicate which reference samples are needed for prediction. It includes a LUT and the output depends on the position of the 4x4 prediction block and the prediction mode. The position of reference samples can be found in the LUT, no matter continuous or discontinuous reference samples. It includes the indices and a flag which indicates whether they're from top neighboring samples or left ones. By this way we simplify the control circuit and reduce the area of circuit. For example, when TU's size is 16x16, mode is 23, and current processing 4x4 block is in bottom-left corner inside 16x16PU, the smoothed reference samples needed are left 13,10,6, 3, top-left samples, top 0. As [1 2 1] FIR filtering is needed in 16x16 PU, the neighboring left and right samples of 13, 12 and 14 need to be fetched from SRAMs. Totally, in this cycle reference samples in RAM 4, 5, 6, 7, TL, and 0 are used.

In some cases, additional cycles are needed for data preparation. Such as in 16x16 TU, 16+1 cycles are needed for Planar mode and DC mode. It is used for getting the bottom-left and top-right pixels needed in planar mode and calculating the un-filtered DC value is needed in DC mode. For 32x32, 4 additional cycles are needed for data preparation.

3.3. Hybrid Data Forwarding and Block Reordering

After each PU is intra predicted, the values are added by the residuals coming from IT/IQ process, and then part of them are written back to SRAM as reference samples. For each 4x4 Prediction block, it takes 4 stages in the pipeline to finish.

The dependency occurs when the next TU needs the result of current TU and previous as reference samples. We classified it into 2 cases to discuss, one is 4,8 TU and the other is 16 and 32 TU. The 4x4 Prediction block processes a 4x4 TU in a cycle. As is shown in Fig. 3, PB_1 is being processed at cycle n+2, and PB_2 in Z-order is processed at cycle n+3. At cycle n+3, PB_2 may need the reference samples located in PB_1, while the result of PB_1 cannot be

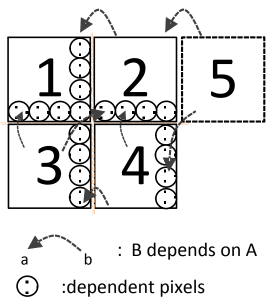


Fig. 6(a)
Dependency of 4x4 TUs

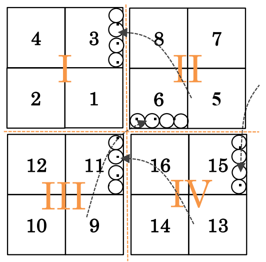


Fig. 6(b)
Dependency of 8x8 TUs

used by PB_2 yet, because it is being written back to SRAM. If such reference samples read from SRAM, and used by PB_2, wrong result occurs because reference samples fetched from SRAMs are not the reconstructed samples of PB_2 we expects. Data hazard also occurs when PB_k writes back reconstructed samples in M_k stage while PB_{k+2} read the same data in D_{k+2} stage. For dual ports SRAMs, the read address and write address cannot be the same when write and read operate at the same cycle. As shown in Fig. 6 (b), Because TU_{II} dependents on TU_I, the PB_5 in TU_{II} may use the result of PB_1 and PB_3 in TU_I as reference samples. PB_3 is processed 2 stages earlier than PB_5 in TU_{II}, so when PB_3 is in P stage, PB_5 is in A stage. PB_5 will read the data from SRAM where PB_3 is to write the reconstructed result. Thus data hazard occurs. Such problems also happen when TU III's block PB_9 depends on PU_{II}'s PB_6, PU IV's PB_{13} depends on PU_{III}'s PB_{11} and PU_V's PB_{17} depends on PU_{IV}'s PB_{15}.

To solve the dependency problem illustrated above, one method is to stall the pipeline for 2 cycles, illustrated in Fig. 5(a). However, it reduces the throughput greatly. Our solution, as shown in Fig. 6(a), is to send forward the result of "P" stage to registers, and at next cycle, next TU reads the reference samples from registers correspondingly, instead of waiting for the predicted samples to be written into SRAMs. Two data forwarding paths are built, one from stage P_k to P_{k+1}, and the other from stage P_k to D_{k+2}, as shown in Fig. 5(b).

Though Data Forwarding can solve the dependency problem of all other sizes besides 4x4 and 8x8, we do not apply it to the others for 2 reasons. Firstly, data forwarding makes the pipeline design and implementation more complicated and critical delay become longer. Secondly, as the processing order inside a TU is not restricted, we have alternative to use processing reordering to solve this problem, which is easier comparing with Data Forwarding. By taking the right-bottom 4x4 block as the start position, from right to left, from down to up, and the top-left block as the last processed one. The processing order can be referred to red numbers in Fig. 4. Block reordering is applied to 8~32 TU. This method guarantees that the 4x4 block located at the right or bottom of TU can be processed earlier, so that the data in SRAM could be ready for the use of next TU.

Table 1 Comparison of HEVC intra prediction architecture for video decoding

	This work	TVLSI'13 [5]	ASICON [7]	ICIP'11 [4]
Platform	90nm	40nm	65nm	130nm
Area	72.1K	27.0K	77K	36.7K
SRAM	21.0kB	4.9kB	--	--
Pred. mode	All	All	All	17
PU Sizes	All	All	All	4x4
Max. Freq. (MHz)	469	--	600	150
Specification (decoding)	4320p, 120fps @397MHz	2160p, 30fps @200MHz	--	100M samples/s @150MHz
TP(samples/cycle)	15	2	16	0.67
Norm. TP (samples/cycle/k-gate)	0.21	0.07	0.21	0.07

4. IMPLEMENTATION RESULT

The proposed VLSI design is implemented by Verilog HDL, and synthesized using TSMC 90nm Technology. The maximum performance is 469MHz. The size of SRAM equals to (1+0.5+0.5)*(number of pixels in width+256) Bytes**.** If luma and chroma samples are processed in serial, pixels to be processed per second equals to 7680* 4320*120*1.5. The throughput's requirement can be met when system's frequency is above 397MHz, when 16 samples are processed per cycle with 1/17 cycles used for data preparation.

Comparing with the work in [5], we process 16 samples per cycle, and use a 4-stage pipeline and data forwarding to increase the throughput. Also every cycle we only fetch several necessary reference samples from SRAM, instead of fetching all reference samples at a time and storing them all in registers. A high normalized throughput is achieved in this work. Firstly, we achieve a high throughput and hardware utilization about 94%(16/17). Secondly, more reference samples can be reused by 4x4 Block and more neighboring predicted samples are probable to share same reference samples. At last, reference samples fetching scheme help reduce the number of registers for storing reference samples and complexity in calculation, so that circuit area is reduced.

5. CONCLUSION

In this paper, we present intra prediction hardware architecture for HEVC 4320p@120fps decoding. We process a 4x4 Prediction Block per cycle. By CSB-PRSF, necessary reference samples are fetched from SRAMs at each cycle, with a relative low complexity and circuit area. By Hybrid Data Forwarding and Block Reordering, we eliminate the dependency between PUs to increase throughput. Hardware utilization of 94% is achieved and only 272 cycles are used to process a CTU in the worst case.

6. REFERENCE

[1] "H.265: High efficiency video coding", ITU-T Rec. Apr, 2013.

[2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[3] J. Lainema, F. Bossen, W.-J. Han, J. Min, and K. Ugur, "Intra coding of the HEVC standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1792–1801, Dec. 2012.

[4] F. Li, G. Shi, and F. Wu, "An efficient VLSI architecture for 4x4 intra prediction in the high efficiency video coding (HEVC) standard," in Proc. 18th IEEE Int. Conf. Image Process., Sep. 2011, pp. 373–376.

[5] C.-T. Huang, M. Tikekar, A.P. Chandrakasan, "Memory-Hierarchical and Mode-Adaptive HEVC Intra Prediction Architecture for Quad Full HD Video Decoding," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.PP, no.99, pp.1,1, 0

[6] D. Palomino, F. Sampaio, L. Agostini, S. Bampi, & A. Susin (2012, September). "A memory aware and multiplierless VLSI architecture for the complete Intra Prediction of the HEVC emerging standard." In Image Processing (ICIP), 2012 19th IEEE International Conference on (pp. 201-204). IEEE.

[7] C. Liu, W. Shen, T. Ma, Y. Fan, X. Zeng "A highly pipelined VLSI architecture for all modes and block sizes intra prediction in HEVC encoder," ASIC (ASICON), 2013, vol., no., pp.1,4, 28-31 Oct. 2013

[8] N. Zhou, D. Ding, L. Yu. "On hardware architecture and processing order of hevc intra prediction module." 30th Picture Coding Symposium (PCS), 2013,vol., no., pp.101,104, 8-11 Dec. 2013