A Memory Aware VLSI Architecture for the Complete Intra-Frame Prediction of the Emerging HEVC Standard

[1]Daniel Paomino, [1]Felipe Sampaio, [2]Luciano Agostini, [1]Sergio Bampi, [1]Altamiro Susin
{dmvpalomino,fmsampio,bampi}@inf.ufrgs.br, agostini@inf.ufpel.edu.br,
altamiro.susin@ufrgs.br
[1]Federal University of Rio Grande do Sul
[2]Federal University of Pelotas

Abstract

This work proposes a hardware architecture for the Intra Frame Prediction of the emerging High Efficiency Video Coding (HEVC) standard. The architecture was designed considering all innovative features of the Intra Prediction included in the HEVC, i.e. all modes and all Prediction Units (PU) sizes. Performance and memory accesses are a problem in the HEVC intra prediction and hardware architecture designs are good alternative to solve these issues, especially when energy-efficient solutions are targeted. Buffers and internal memories were used in the designed architecture to decrease the number of external memory accesses. Two independent datapaths processing eight samples in parallel and with a deep and multiplierless pipeline were designed to increase the throughput. The architecture was synthesized using an IBM 65nm CMOS technology. The results have shown that the architecture is able to process 30 HD720p frames per second and 13 HD1080p frames per second when running at 500 MHz, reducing in 95% the accesses to the external memory.

1. Introduction

The High Efficiency Video Coding (HEVC) [1] is the emerging video coding standard that is being developed aiming to double the compression rates when compared to the latest consolidated standard, the H.264/AVC [2]. This claim for even more coding efficiency is happening due the recent technology advances, which are enabling many improvements regarding multimedia systems.

All the tests and evaluations to define the coding tools that will be inserted in the final draft of the standard are being performed using the HM test model [3].

The HEVC defines data structures that can be pointed as the main reasons of its high complexity. Frames are divided into quad-trees called Coding Units (CU). Each CU could be divided into Prediction Units (PU). At this level the prediction steps are applied. The CU division can be represented recursively in a quad-tree structure.

The Intra Prediction module is responsible to exploit the spatial data redundancies inside a frame. By using the already coded samples as reference, the Intra Prediction generates candidate blocks using different copy modes. The HEVC increases the computational complexity of the Intra Prediction in order to achieve gains in the coding efficiency, when compared with the H.264/AVC. Instead of the two block size partitions of the H.264/AVC (4x4 and 16x16) [4], the HEVC allows five different PU sizes for the Intra Prediction (4x4, 8x8, 16x16, 32x32 and 64x64). Besides, the number of prediction modes is much higher, 34 in the worst case against 9 in the H.264/AVC. This high complexity stimulates the design of dedicated hardware solutions in order to design energy-efficient results.

There is no published work in the literature that implements the complete Intra Prediction for all CU treeblock possibilities. The work [5] implements a simplified architecture that is able to perform the Intra Prediction only for 4x4 PU sizes of the image.

The goal of this work is to design an Intra Prediction hardware architecture that processes all prediction modes for all PU sizes in the quad-tree approach. The architectural design was planned to allow real time processing for high resolution videos. Besides, another target of this work is to reduce the number of memory accesses which is required in the reference implementation presented in the HM test model, since the HEVC requires the use of huge block sizes, like 64x64 samples.

The architectural design presented in this work considers: (i) five different PU sizes (64x64, 32x32, 16x16, 8x8 and 4x4) and (ii) 34 different prediction modes (33 angular predictions plus one DC prediction). In order to handle with the memory accesses to fetch, in the worst case, the entire 64x64 treeblock, on-chip memories and internal buffers were used. Two identical data paths were designed to process, each one, 17 prediction modes. This way, due the angular predictions regularity, the same control word is sent for the two execution paths.

2. Intra Prediction

The Intra Prediction process in the current HM software can be performed in several ways. Depending on the PU size, there are until 33 directional possible modes plus the DC mode for luminance components. This great amount of possible modes has been useful to improve the coding efficiency in the HEVC encoders, when compared with H.264/AVC compliant encoders [1].Table 1 shows the number of available modes considering each possible PU size.

The number of block sizes possibilities and available prediction modes have increased a lot in comparison with H.264/AVC. For this reason, the computational complexity of the intra prediction mode has increased as well, complicating some implementations issues, like the minimum requirements of throughput, energy budgets, hardware cost and so on.The 33 possible intra prediction direction modes considering an 8x8 PU size are illustrated in Fig. 1., with the vertical and horizontal angles +/-[0, 2, 5, 9, 13, 17, 21, 26 and 32] [1].

Table 1. Number of intra prediction modes.

PU Size	Number of Intra Modes
64x64	3
32x32	34
16x16	34
8x8	34
4x4	17

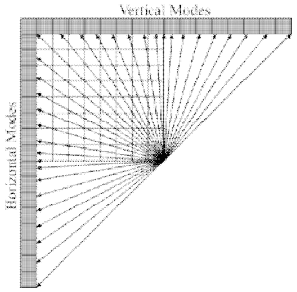

Fig. 1. 33 possible modes directions for HEVC Intra Prediction.

The predicted pixels are generated in according with the Equations (1) and (2). The integer pixels ***PInt*** are generated only by copying the according neighbor. The fractional pixels ***PFract*** are generated by an interpolation of two neighbors as shown in Equation (2). The constants k_0 and k_1 are calculated based on the prediction angle and further details are in the current HEVC working draft [1].There is also the DC mode, where each predicted pixel is calculated based on an average of all available neighbors.

(1)

(2)

3. Designed Architecture

The main goals of the designed architecture were: (a) perform the entire intra prediction process (all modes and all PU sizes), (b) reduce the number of external memory accesses as much as possible and (c) provide a throughput enough to process high resolution videos in real time. Next sections detail the designed features.

3.1. Architecture Data Paths

The first decision to increase the throughput of the designed architecture was the data path division in two parts (data path above and data path left). The architecture is divided according to the directions of the available modes. The data path above is responsible to perform all vertical modes while the data path left performs all horizontal modes (17 modes each). The mode 3 is performed in both data paths to maintain the architecture regularity, and only one result is used. Fig. 2 shows how the angular modes are attributed to each data path.

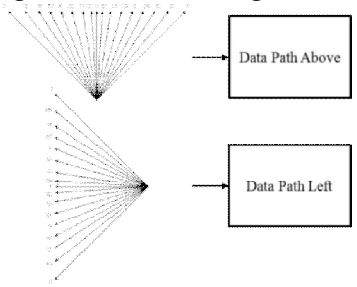

Fig. 2. Data path division.

Besides the throughput increase, this division in two data paths brings some other benefits to the architecture. For two modes in each data path with equal prediction angles, the constants used in the sample prediction and the addresses to access the neighbor pixels are equals, which simplifies the architecture control.

Other alternative used to increase the throughput of this architecture was to increase the number of samples that will be computed in parallel. In this work the parallelism level is eight samples, which means that each data path will process four samples per clock cycle. This parallelism was defined considering the communication with external memory and the required throughput to reach real time processing. This way, each data path is composed by four sample predictors.Fig. 3 shows the simplified block diagram of the designed architecture.

The operative part is composed only by adders and subtractors. The multiplication presented in the Equation (2) was decomposed in shifts and adders, since the $k_0$and k_1 constants are always values between 1 and 31. Each adder was included in one independent pipeline stage, to reach a critical path as small as possible. This way, the architecture data path (sample prediction and SAD calculation) is composed by 9 pipeline stages.

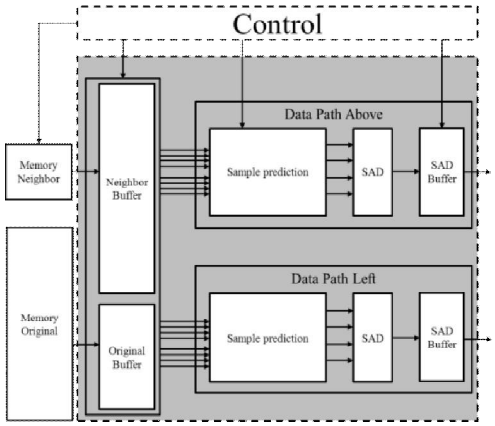

Fig. 3. Block diagram of the designed architecture.

3.2. Buffers and Internal Memories

In order to locally store the necessary samples for the Intra Prediction execution, two different on-chip memories were designed: *Neighbor Memory* and *Original Memory*. The neighbor memory uses 129 words of eight bits and delivers one sample per clock cycle. The original memory has 1024 words of 32 bits each (one 64x64 treeblock). The samples were organized to be aligned in the memory. It means that each read loads 4 samples per clock cycle.

In HM software the intra prediction process over a PU is performed in two steps. First, one predicted block is generated according to one of the available modes. Then, the residual between the original block and the predicted block is generated. It means that the original block is accessed as many times as the number of available modes, increasing the number of redundant memory accesses. In this work, the prediction is performed in a different way to save memory accesses.

Initially, all necessary neighbor samples (when available) are loaded from the memory of neighbors to an internal register bank (*Neighbor Buffer*). This register bank is composed by 129 registers of 8 bits, since in the worst case (when the prediction is over 64x64 PUs) there are 129 neighbors to be used by the prediction. The neighbor buffer was used since the data paths need four or five samples (integer or fractional prediction) per clock cycle while the memory delivers only one sample per cycle. Then four original samples are loaded from the memory of originals samples to another buffer (*Original Buffer*).

When the load step is complete, the prediction of each mode starts. Instead of performing one mode prediction using all original samples and then reload the original samples to perform the next mode, all modes are applied to those four samples. This way, the original samples are loaded only once from the external memory, decreasing the number of memory accesses. Since only four samples (in each above and left data path) are calculated in parallel, another buffer (*SAD Buffer*) with 17 registers was used for each data path to store the partial values.Fig. 4 shows a comparison in terms of memory accesses per treeblock considering the corner cases of the design.

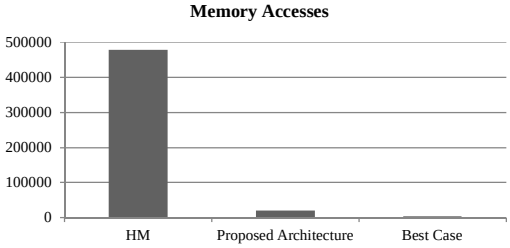

Fig. 4. Comparison of memory accesses.

In the HM software each original sample is accessed for each mode prediction of one PU while the proposed architecture access the sample only once for each PU size. This way, the designed architecture saves 95% in memory accesses when compared to the HM software. Besides, comparing with the best case (when the original samples are accessed only once for a whole treeblock prediction) the proposed architecture uses five times more access. The best case was not considered as target, since it implies in a great amount of internal registers, increasing the architecture resources.

3.3. Architecture Schedule

As all samples (neighbor and original) are available after the load step, the pipeline technique was also used to increase the architecture throughput. Besides, it improves the architecture usage, since after the latency of loading the neighbor samples the architecture will always have valid input data.

The first step is to load all available neighbor samples from the external memory. This step is performed only once for each PU size and the time needed to load these neighbor samples varies (from 9 to 129) according

to the target PU size and the available neighbors. The neighbors will be used for both data paths. Then the loading of the four original samples for each data path takes only one clock cycle. After that, the computing of all possible modes for the given PU is performed. Finally the partial SAD values are storage.

The architecture working flow is based on the PU size. For each PU size the architecture takes different number of clock cycles to process all the prediction modes. Table 2 shows how many clock cycles are spent to perform each possible PU size. The 33 direction modes are applied only to 32x32, 16x16 and 8x8 PU sizes. For 4x4 PU size only 17 direction modes are applied and for 64x64 PU size only two direction modes (horizontal and vertical with angle 0) are applied. The DC mode can be applied for all PU sizes.Considering a 64x64 treeblock and the number of clock cycles presented in Tab. 2, the architecture takes 73.682 clock cycles to process a whole treeblock, i.e. all possible modes for all possible PUs.

Table 2. Number clock cycles to perform each PU size.

PU Size	# Clock Cycles
64x64	1162
32x32	4490
16x16	1162
8x8	314
4x4	62

4. Synthesis Results

The architecture was described in VHDL and synthesized to IBM 65nm standard cell library [6]. Table 3 presents the synthesis results.

Table 3. Synthesis Results

Work	This work	Li [5]
Technology	IBM 65nm	TSMC 130nm
Gate Count	36.734	9.020
Max Frequency (MHz)	500	150
PU size supported	All	Only 4x4

The designed architecture can work at a high operation frequency (500 MHz). This is possible since the critical path of the architecture is only one adder of 29 bits. Considering this operation frequency and the number of clock cycles necessary to process all modes for all PUs sizes in a treeblock, the designed architecture can process 30 HD720p frames per second and 13 HD1080p frames per second.

There is only one work in the literature [5] that presents a hardware design for the HEVC intra prediction module. The work proposes a simplified architecture to process the intra prediction only over 4x4 PUs while our proposed architecture is able to process all PU sizes. Besides, their technology target was TSMC 130nm. This way, it is hard to do a more detailed comparison. An intra prediction architecture that processes all modes for all PU sizes was not found in the literature.

5. Conclusions

This work presented the design of a hardware architecture for the intra-frame prediction of the emerging HEVC video coding standard. This is the first work in the literature relating an HEVC intra prediction architecture that performs all intra prediction modes for all possible PU sizes. The architecture was designed to decrease the number of memory accesses and to provide a high throughput. The data reuse allowed a reduction in 95% of the external memory accesses when compared to the HM strategy. The efficient and parallel data path allowed the architecture to run at 500 MHz, being able to process 30 HD720p frames per second and 13 HD1080p frames per second.

6. References

[1] JCT. Working Draft 3 of High-Efficiency Video Coding. JCTVC-E603, 2011.

[2] JVT (T. Wiegand, G. Sullivan, A. Luthra), Draft ITU-T Rec. and final draft int. stand. of joint video spec. (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC), 2003.

[3] ISO/IEC-JTC1/SC29/WG11, "HEVC Reference Software Manual," ed. Geneva, Switzerland, 2011.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE TCSVT, vol. 13, pp. 560-576, 2003.

[5] Li, F. and Shi, Guangming “An Efficient VLSI Architecture for 4x4 Intra Prediction in the High Efficiency Video Coding (HEVC) Standard” in IEEE ICIP, pp.381, 384, 2011.

[6] Virage Logic. “High Density Tapless Standar Cell Logic Library for Common Platform 65nm LPe LowK Standard Vt Process”, 2009.