The Demonstration of Low-Power High-Performance H.264 Decoder with Rapid SoC Prototyping Platform

Sangkwon Na, Woong Hwangbo, Jaemoon Kim, Sheunghan Lee and Chong-Min Kyung

{skna, woonghb, jmkim, shlee04}@vslab.kaist.ac.kr, kyung@ee.kaist.ac.kr

VLSI Systems Lab. - KAIST - South Korea

http://vswww.kaist.ac.kr

Abstract

We introduce 4x4 sub-macroblock pipelined H.264 decoder, and data-reuse mechanism to reduce memory access and power consumption. The proposed H.264 decoder can support CIF 30fps videos at 8MHz with 25,426 LUTs and 4KB SRAM in Xilinx Virtext4.

1. Introduction

Nowadays hand-held devices have provided multimedia service, such as streaming and/or playback. Among many applications the multimedia CODEC, such as H.264, requires lots of computing power. Especially, H.264 adapted redundancy reduction algorithms to minimize the bit-rate of coded sequence and support high quality video. Because of these complex algorithms, the need has arisen for hardware implementation.

Our project aims at the development of low-power and high performance H.264 decoder for mobile terminals. After H.264 standard publication, many design/implementation results have been reported. [1] explains how to implement H.264 decoder based on a SoC platform design methodology. Because intra/inter prediction is executed on ARM processor, its performance does not exceed 20.15 fps at most. [2] proposes hybrid task pipelining architecture. Because the sequence parameter set (SPS), picture parameter set (PPS), and slice header are parsed by RISC processor, it distorts the pipeline advance. In other words, the hybrid part can be bottleneck among decoding processes.

In this paper, we report low-power and high-performance H.264 decoder. The most important design point is 4x4 sub-macroblock pipeline architecture. We implemented each algorithm element which can process samples as 4x4 sub-macroblock unit. Hence, our design guarantees high throughput maintaining low power dissipation. In addition, we applied data reuse mechanism to MC and DF, and reduce the memory access and power consumption.

This paper is organized as follows. Section 2 will present the proposed 4x4 sub-macroblock pipeline architecture. In section 3, we will describe how we implement and verify the design. Section 4 will show the experimental results and the decoder performance. In section 5, we will conclude our work.

2. 4x4 Sub-Macroblock Pipeline Architecture

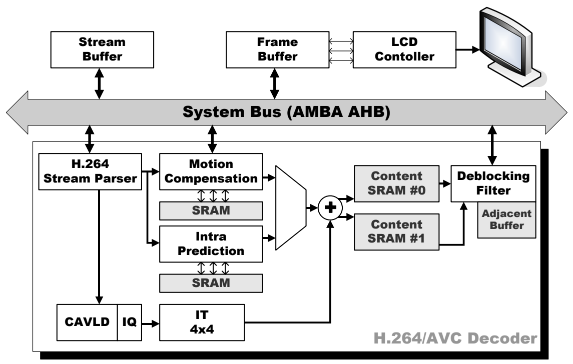

Figure 1: H.264 Decoder System Diagram

Our H.264 decoder has 4x4 sub-macroblock pipeline architecture. The processing unit of this pipeline architecture is a 4x4 sub-macroblock, and each functional block transfers sample data through private channel. It is described in figure 1.

"H.264 *Stream Parser (SP)*" reads coded data from stream buffer, and parses them to get header information, sample types, and sample data. After parsing, SP transfers data to and controls other functional blocks. Actually, SP works as a central controller. Because one macroblock is predicted as inter-frame type or intra-frame type, the results of "*Motion Compensation (MC)*" and "*Intra Prediction (IP)*" pass through the multiplexer, and one of them is added up to the residual sample. The residual sample is made by "*Inverse Transform (IT)*" and "*Inverse Quantization (IQ)*." Final decoded samples are transferred to "*Deblocking Filter (DF)*," and it executes filtering process and stores the results at the frame buffer.

For the performance improvement, we devised and applied some design techniques. With formula transformation we reduced the processing element for IQ/IT, so we obtained results of IQ/IT within one cycle while consuming low power. Furthermore, we decomposed and re-assembled CAVLD and IQ. As a result, just one IQ unit was used to produce inverse-quantized samples.

We adopted a hybrid filtering sequence for DF. It satisfies the required filtering condition defined in the H.264 standard. Finally, DF needs 208 clock cycles to filter one macroblock.

3. Data Reuse Mechanism

We apply data reuse mechanism to MC and DF, and reduce the memory access. Off-chip memory access consumes large amount of power, so proposed method can reduce power consumption effectively.

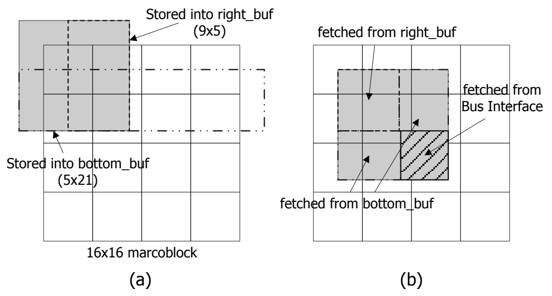

Figure 2: Reusable pixels for MC

Our MC refers to other macroblocks which are directed by motion vectors, and uses buffers for reusable pixels. As a result, it reduces memory access by up to 65-80%. If previously-used pixels are reusable for other sub-macroblock reference, they are stored in the pixel cache buffer. It is described in figure 2.

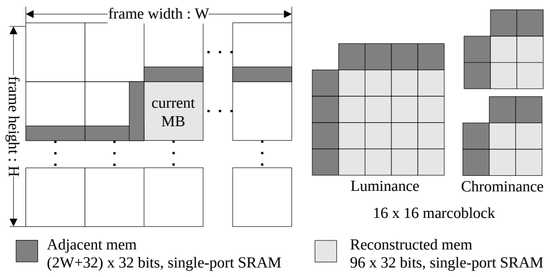

Figure 3: Memory architecture for DF

Our DF uses pixel buffers which have ready-filtered pixels, and it is called 'adjacent memory.' Adjacent pixels which are out of boundaries of current macroblock should be accessed from the frame buffer; however, the access to external memory decreases the performance and consumes a large amount of power. Hence we use adjacent memory for buffering ready-filtered pixels. Figure 3 presents memory architecture for DF, and a dark grey rectangular denotes adjacent memory.

4. Experimental Results

After the design is finished, we integrated all designs and tested them with generic sequences. There are six generic sequences named as 'akiyo', 'carphone', 'foreman', 'football', 'news', and 'stefan'. We used RTL simulator, VCS MX 7.2, and our design is described in synthesizable Verilog HDL. Our H.264 decoder is synthesized with 25,426 LUTs and 4KB SRAM in Xilinx Virtex4.

Coded bit-stream is loaded into the stream buffer, and decoded images are recorded as file. After all sequences are decoded, decoded results are verified with golden results gathered from reference software, JM v10.2 [3].

About test sequences, we measured the performance of proposed H.264 decoder. Table 1 shows the operating clock frequencies needed for real-time decoding process.

Finally we made H.264 decoding system upon SoC prototyping platform, iNTUITION [4]. Figure 4 shows H.264 demonstration system displaying decoded image at LCD display. This system has ARM1136 core-tile, ZBT SRAM, FLASH and LCD controller.

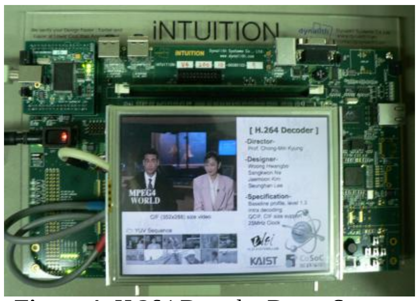

Figure 4: H.264 Decoder Demo System

Table 1: Decoding performance results

Video sequence	**size**	**frequency (MHz)**
Akiyo	QCIF/30fps	1.46
Carphone	QCIF/30fps	1.83
Foreman	QCIF/30fps	2.01
Football	CIF/30fps	7.94
Foreman	CIF/30fps	7.78
News	CIF/30fps	6.01
Stefan	CIF/30fps	7.99

iNTUITION provides high-density FPGA and built-in logic analyzer, therefore fast logic verification is available. Furthermore, verification can be performed under real chip environment with the peripherals of iNTUITION. With ARM1136 core, iNTUITION supports HW/SW co-design/verification feature.

5. Conclusion

We developed 4x4 sub-macroblock pipelined H.264 decoder targeting low power and high performance. Data reuse mechanism helps to reduce power consumption. Proposed H.264 decoder needs 8MHz as operating frequency for CIF/30fps decoding operation.

Our H.264 decoder is synthesized and mapped into Xilinx Vertex4 FPGA. We demonstrate our design with rapid SoC prototyping platform, iNTUITION.

6. References

[1] S. H. Lee, J. H. Park, S. W. Kim, S. J. Ko and S. Kim, "Implementation of H.264/AVC Decoder for Mobile Video Applications," *Proc. of ASPDAC*, 2006.
[2] T. Chen, Y.-W. Huang, T.-C. Chen, Y.-H. Chen, C.-Y. Tsai and L.-G. Chen, "Architecture Design of H.264/AVC Decoder with Hybrid Task Pipelining of High Definition Videos," *ISCAS'05*, 2005.
[3] H.264/AVC Software Coordination, Heinrich-Hertz-Institut, http://iphome.hhi.de/suehring/tml
[4] iNTUITION, http://www.dynalith.com/intuition.php

Acknowledgement

This research has been supported by grants from *SoC Initiative for Ubiquity and Mobility*. Thanks are due to Dynalith Systems Co. for assistance with the equipment and implementation.