Chapter 1

Introduction

H.264/AVC is the latest video coding standard and is recognized as the most efficient one. Several studies have shown that H.264 gains up to 50% of compression rate as compared to other previous standards [1]-[3]. Similar to previous standards, the H.264/AVC is based on a motion compensated hybrid DCT (Discrete Cosine Transform) algorithm. Where sub-pixel motion compensation is one of the main factor that makes H.264 a good coding scheme. However, sub-pixel motion compensation comes with a cost of the high computational complexity.

After profiling the H.264 decoder code (JM 8.0, [4]) on ARM development environment, we observe the percentage of runtime of each sub-procedure. Table. 1.1 shows the parameter of the benchmark. Fig. 1.1 shows the result. We can see that the interpolation procedure takes up to 22% execution time. In this thesis, we will pay attention to the interpolation procedure which is one of the most time-consuming functions.

Table 1.1: Parameters of benchmark for profiling

benchmark	foreman.qcif
number of frames	128
frame sequence	IPPPPP....
reference frame	5
QP	28

[5] discuss the optimization of the H.264/AVC sub-pixel interpolation operation. UNPACK and PACK [6] instructions were combined with {LOAD} and {MIN, MAX, STORE}, respectively. The proposed instruction set extensions result in cycle savings without much

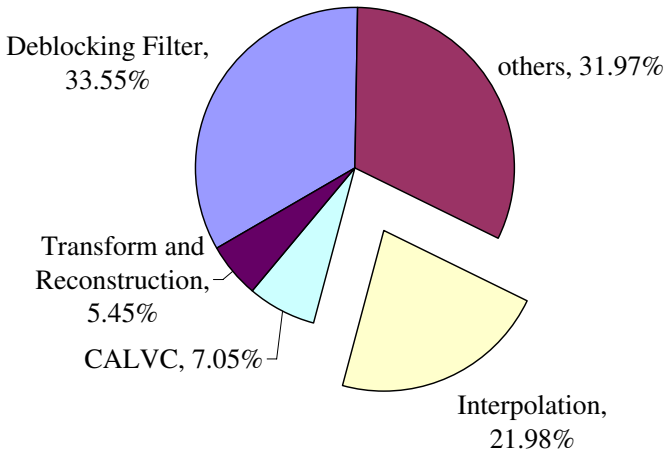

Figure 1.1: Execution time profiling of foreman.qcif on H.264 decoder

hardware overhead.

We take the architecture and the instruction set of ADI Blackfin533 as an example DSP to understand how exactly the interpolation performs. After porting the interpolation procedure code on the simulator of ADI Blackfin533, we dump out the assembly code of interpolation procedure and analyze it in detail. We notice that even if the compiler can generate the most efficient code based on current ISA, the procedure still spends much time on computing unnecessary operations. The reason is that traditional instructions are designed for general usage. Since the computation of interpolation is high, it worths to design special instructions for the procedure.

Many researches have been proposed to decrease the computational complexity of interpolation. For a pixel in a referenced block, computation of interpolation needs six multiplications. To reduce the number of multiplications, [7] proposed a new algorithm for interpolating with a 4-tap filter. The interpolation procedure loads data from memory frequently. To reduce the number of memory access, [8] proposed an efficient method to achieve the goal where a set of vector registers that hold a block of pixels to minimize the latency of memory access is designed.

In this thesis, we will define new instructions to execute the interpolation procedure in a DSP processor. We use the existing components as much as possible and add small amount

of hardware overhead to execute new instructions. Functionality of interpolation remains unchanged and the number of executing instructions are decreased.

The rest of the thesis is organized as follows. In Chapter. 2, we briefly describe the functionality of interpolation, definition of new instructions, and how new instructions are applied. Chapter 3 demonstrates the hardware design which is based on Starfish DSP developed in NCTU. Modifications of Starfish-assembler and Starfish-Simulator are also briefly described. Experimental results on performance and overhead of our design are shown in Chapter 4. Finally, Chapter 5 gives the concluding remarks.