Feasibility of Implementing an H.263+ Decoder on a TMS320C6X Digital Signal Processor

EE382C
Embedded Software Systems
Dr. Brian Evans

May 12, 1999

Kurt Nee
Chad Roesle

ABSTRACT

H.263+, version 2 of ITU-T H.263, is a specification for video compression that is especially useful in low bit-rate situations. It includes many different modes of operation to deal with many of the problems associated with low bit-rate applications. Our project consists of analyzing possible implementations of H.263+ embedded video decoders, and then determining the feasibility of implementing a decoder on a Texas Instruments TMS320C62x Digital Signal Processor chip. Finally, we present our implementation results.

INTRODUCTION

With the advent of cellular phones and digital cameras, mobile videophones are a likely progression for technology in the future. Wireless applications of video transmission, however, require special considerations. Three key design criteria are the size, power consumption and video quality of the device. Our project will involve determining the feasibility of implementing a video decoder in an embedded system, in which size, power and bandwidth are severely limited. To accommodate all of these limitations, every aspect of the implementation must be optimized. H.263+ is a standard video compression algorithm that is well suited to low bandwidth transmissions. Our project will attempt to optimize an H.263+ decoder on a Texas Instrument's (TI's) TMS320C62x DSP chip.

H.263 OVERVIEW

Approved by the ITU-T in 1995, H.263 is similar to the MPEG-1 and MPEG-2 video coding standards. Three basic steps are taken to encode video sequences in baseline H.263 without any optional modes. The first step is to reduce temporal redundancies in the picture by using motion-compensation prediction. The next step is to encode the motion-compensated frames using a discrete cosine transform (DCT)-based algorithm. Finally, the quantized DCT coefficients, motion vectors, and side information are entropy coded using variable-length codes (VLC's). These steps are shown in Figure 1. Many people have tried to optimize different portions of the baseline coding of H.263 to make encoders with higher complexity and performance, such as [1]. In addition to the baseline coding specifications, H.263 specifies 4 negotiable advanced coding modes. These modes are the Unrestricted Motion Vector Mode (Annex D), Syntax-Based Arithmetic Coding Mode (Annex E), Advanced Prediction Mode

(Annex F), and PB-Frames Mode (Annex G). The unrestricted motion vector mode allows motion vectors to reference pixels outside the boundaries of the picture, and allows a larger range of values for the vectors.

The syntax-based arithmetic coding mode reduces the variable-length entropy codes by approximately 5%. Advanced prediction mode allows four motion vectors for each macroblock of pixels. PB-frames mode virtually doubles the picture rate without a significant increase in bit rate by forward predicting P frames from an I frame, and bidirectionally predicting B frames from the I frame and P frame. The I frame is the only frame which is fully encoded. The others are encoded with motion vectors and delta vectors [2]. The order of the decoded frames would look something like IBPBPIBPBPI…

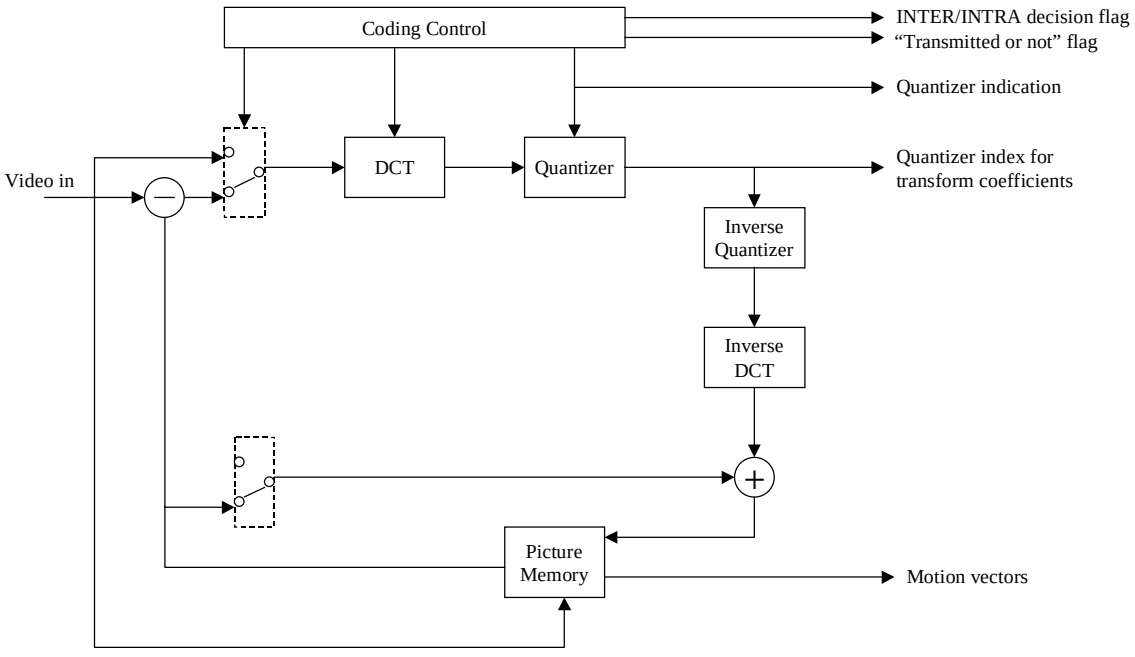

Figure 1: H.263+ video encoder block diagram [2]

H.263+ OVERVIEW

The ITU-T H.263+ standard is the second version of H.263, and is backward compatible with H.263 video streams. The objective of H.263+ is to improve compression efficiency of coded video, introduce error resiliency and allow a wider range of applications. H.263+ specifies 12 new optional coding modes in addition to the four specified in H.263, and redefines the Unrestricted Motion Vector Mode in H.263. Each of the new optional modes in H.263+ is an annex labeled from "Annex H" to "Annex T". Detailed descriptions of these modes can be found in [2] and [3].

TEXAS INSTRUMENTS DSP TMS320C62x

Texas Instruments' TMS320C62x is part of their C6000 line of DSP products [4]. All devices in this line are based on the same CPU core. The C6000 core incorporates two multipliers and six arithmetic units, making execution resources abundant. The chips fetch eight 32-bit, RISC-like instructions per cycle. The instruction packing features allow these instructions to be executed in parallel, serial or parallel/serial combinations. This design allows significant reductions in code size, number of program fetches and power consumption.

The C62x family of processors is capable of 1200-2000 MIPS with clock speeds ranging from 150-250 MHz. Program memory on the chips ranges from 512 Kilobits (Kb) to 2 Megabits (Mb). The data memory sizes range from 512 Kb to only 1 Mb. On most models, the memory areas are arranged in two blocks for improved concurrency. In addition, a 32-bit External Memory Interface (EMIF) allows access to synchronous as well as asynchronous memories. Lastly, the entire family offers at least two Multichannel Buffered Serial Ports (McBSP) for a wide variety of input/output (I/O) options. The clocks on the McBSP's run at half the clock

speed of the chips, which we consider adequate for our implementation. Typically, the chips require 3.3V for I/O and 1.8V for core operation.

Some other notable features of the C6000 series include a load-store architecture with 32 32-bit general-purpose registers, instruction packing and 100% conditional instructions. The instruction set features include byte-addressability, 32-bit address range, 8-bit overflow protection, saturation arithmetic, bit-field extract, set and clear, bit-counting, and normalization. Some of the possible applications of the C62x include wireless base stations, pooled modems, cable modems, Digital Subscriber Loop, and multimedia systems [4].

TMS320C62x COMPILER/ASSEMBLER/LINKER

The development tools provided by Texas Instruments for the TMS320C62x include an ANSI C compiler and an optimizing assembler. The assembler automatically optimizes assembly code generated from ANSI C source code. These tools were specifically designed to allow developers to compile C code into assembly code that takes advantage of the DSP's advanced features, such as the very-long-instruction-word (VLIW) architecture that utilizes parallel execution of code.

In order to create the most efficient object code, the C6000 compiler uses four levels of generic and target-specific optimizations. Some of the optimizations specific to the TMS320C6201 include software pipelining, if conversion/predicted execution, memory address cloning, and memory address dependence elimination. General optimizations for all TI chips include branch optimizations/control-flow simplifications, alias disambiguation, copy propagation, data flow optimizations, and register tracking/targeting. The Assembly Optimizer

works with the code generator to produce optimized assembly code. More detailed descriptions of these tools can be found at [5].

FORMAL MODELING

Here we present our proposal for a decoder on the TMS320C62x DSP chip for use in an embedded environment.

SYSTEM MODELING

The system we propose for a wireless video decoder would require several separate components. First, a receiver is necessary to capture the encoded bitstream. The receiver can perform minimal error correction during this stage. It will keep the encoded data in a buffer for the decoder to read. The buffer will remove some complexity from the decoder by allowing it to

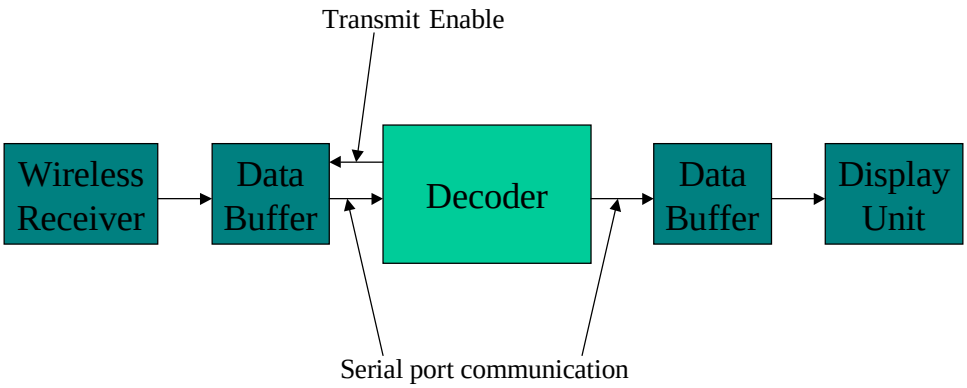

Figure 2. Wireless embedded system model

request entire frames at a time so it doesn't use cycles to keep track of the incoming bitstream – it just decodes the streams. After decoding the frame, the DSP will send the bitstream to another buffer that keeps track of the display. This buffer will actually manage two frames of data – one frame for display and one frame for incoming data. Our proposed system is shown above in Figure 2.

DATAFLOW MODELING

Figure 3 shows how we modeled a basic decoder. In the Finite State Machine (FSM) domain, we wait for the start code of the bitstream. Once the start code is obtained, we obtain which picture type and optional modes are encoded in the bitstream. We show how to decode an I frame using the Synchronous Dataflow (SDF) domain. More detail as to how the picture is decoded can be found at [6].

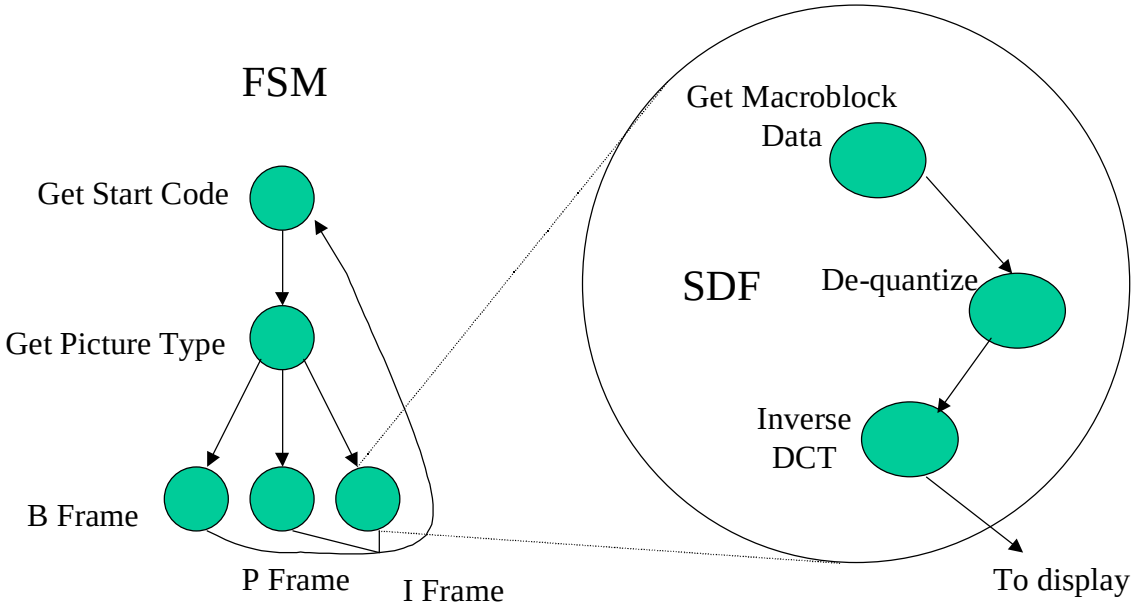

Figure 3. Modeling of decoder

IMPLEMENTATION

As a preliminary step, we estimated the amount of data and program memory required to implement an H.263+ decoder on our DSP and calculated the execution times required for a real-time implementation. Using a standard picture size of 128x96 pixels, one frame would take 18432 bytes of memory. A standard picture size of 176x144 pixels would take 38016 bytes. The amount of data memory we had to work with on the DSP chip was 1Mbits (128KBytes) of memory. We would be able to fit 6 small frames or 3 larger frames in memory on the DSP chip

at one time. Due to the memory constraints, the error correction optional modes in H.263+ could not be implemented on the DSP, since extra frames needed to be stored as references. Using the H.263+ decoder C code created by Kossentini's research group at the University of British Columbia [2] as a starting point, we condensed the C code for a minimal decoder implementation and compiled it using the Texas Instruments compiler tools. We then modified the decoder to input data from the serial ports on the DSP rather than from a file. After compiling the code, we simulated the object code using TI's C6x simulator on a K6-2 450Mhz computer. The linked object file was 300Kbytes, which was too large to fit on the C62x (256Kbytes of program memory), but could be run and debugged in the simulator.

Because of the simulator's speed (5-10 hours to decode one frame), actual timing results have not been obtained yet. We need to further optimize the code and run the decoder on hardware to get an execution time. We are confident that the DSP can decode the frames fast enough based on preliminary calculations on desired frame rate and the MIPS of the DSP. In order to decode 10 frames per second, the DSP has 1/10 seconds per frame. Based on a 4ns clock cycle, the DSP can use 25M clock ticks to decode each frame. We did not optimize any of the kernel components used in decoding, such as the inverse discrete cosine transform (IDCT) on the macroblocks of the picture. Optimization would cut down program memory and speed up decoding, but data memory usage would stay constant.

RESULTS

We modeled the decoder using formal models of computation. After estimating the amount of program and data memory required for an implementation, we compiled and simulated the decoder using the TI tools. Because the serial port emulation would not work in the simulator, we had to input data directly from a file to test the decoder. The 2Mbit constraint

for program memory was easily met with 1.1Mbit of memory necessary. We did not achieve the data memory constraint of 1Mbit since our implementation required ~4Mbits. The core design of our decoder relied on several frames being in memory at any given time to decode motion vectors and be error resilient. In order to meet the data memory requirements, we would remove almost all of the functionality of the decoder, in particular, all the optional modes in H.263+.

CONCLUSION

We expected to implement an H.263+ decoder on the TMSC62x DSP chip that would be suitable for use in wireless applications such as a wireless video phone for two way communication or a wireless videoconferencing network. We used code from the web for our decoding algorithms, and compiled, simulated and analyzed the code using TI's DSP tools. Further work needs to be done to explore options of using the external memory interface on the C6x chip to meet data memory requirements. Using the EMIF, power consumption and size become major factors in the embedded application.

REFERENCES

[1] Y. Jeong and C. Cheong, "A DCT-Based Embedded Image Coder Using Wavelet Structure of DCT for Very Low Bit Rate Video Codec," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 500-508, 1998.

[2] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video Coding at Low Bit Rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849-866, Nov. 1998.

[3] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error Resilience Support in H.263+", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 867-877, Nov. 1998.

[4] "TMS320C6000 Product Information," http://www.ti.com/sc/docs/dsps/products/c6000/index.htm, March 3, 1999.

[5] "TMS320C62x Code Generation Tools," http://www.ti.com/sc/docs/dsps/tools/c6000/comp.htm, March 3, 1999.

[6] "H.263+ C6x Decoder C Source Code," http://www.ece.utexas.edu/~roesle/evans/h263dec.zip, May 12, 1999.