AN1482
APPLICATION NOTE

MPEG2 Main Profile Video Decoding on ST100

By Maurizio COLOMBO

ABSTRACT

This document reports the results of the porting of a MainProfile@MainLevel MPEG-2 decoder on ST120. This is the standard notably used for DVD.

CONCLUSION

The optimized application needs 228.4 MCycles/s to decode 25 PAL frames/s in the case of M=3, N=12, bitrate=15 Mbps. In this evaluation we assume that all the data are into the XY internal memories (ideal DMA).

TABLE OF CONTENTS PAGE

1 - OVERVIEW ON MAIN PROFILE DECODING

Figure 1 : Main Profile decoder block diagram

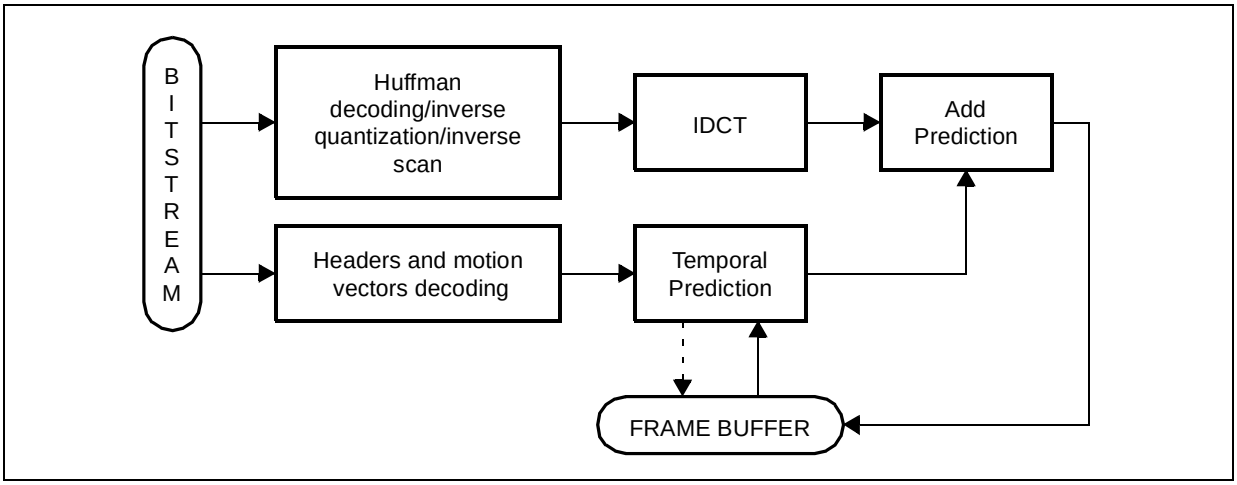

Figure 1 shows the block diagram of the Main Profile decoder [1]. The basic building blocks are the same as for Simple Profile, but some additional features are supported in Main Profile. In particular, Main Profile defines a new type of pictures, called B (bidirectionally interpolated), that are predicted from past and future I/P images. This implies an increased complexity of the *Temporal Prediction* block to support bidirectional prediction.

A Main Profile decoder must support all the prediction modes (up to six) and the two possible picture structures (frame pictures and field pictures).

Block *IDCT/Add Prediction* can be performed either in field or frame mode.

The corresponding bitstream parser, represented by the block *headers and motion vectors decoding*, must be able to support the syntax corresponding to the above features.

Huffman decoding must support MPEG-1 backward compatibility (two special routines have been added for this, since MPEG-1 syntax uses different escape codes) and two types of coefficients scan (zig-zag and alternate).

The following chapters depict all these basic blocks, showing their performance on the optimized assembly implementation for ST120. However, the reference C source is the one developed by the MPEG Software Simulation Group [2].

2 - PROFILING OF THE CODE

2.1 - Wang IDCT

This block has been taken directly from the Simple Profile implementation. Since inverse zig-zag/ alternate scan is already performed by the Huffman decoding/IQ block, there's no need to unroll the first loop of IDCT, and the code size is significantly reduced. The module **wang_idct.st1*** is made by two functions, one for intra IDCT (that includes clipping and add 128) and one for non-intra IDCT. This module uses 32 bytes of read-only data memory for constants (*IDCT_const*). The code size and performances are reported in Table 1.

Table 1 : Wang IDCT (code size and cycles)

Function	Code Size (Bytes)	Cycles/8x8 Block
Wang_idct()	1104	530
Wang_idct_inter()	996	469

This algorithm can be replaced by the high performance Huang IDCT, whose ST120 implementation is now available.

* For more information on the MPEG2 code implemented on ST120 please contact st100.marketing@st.com.

2.2 - Add_Prediction()

This function is located in the module **add_pred.st1**. Its purpose is to compute the sum of two 8x8 blocks and clip the result in the range [0,255]. In fact, one of the two blocks is the predictor computed by temporal prediction and the other is the difference block read from bitstream. The result overwrites the predictor, and this highlights a global optimization that is very effective: **Add_Prediction()** is applied only to the decoded blocks that are non-null, based on the *coded block pattern* read from bitstream. The frame/field DCT coding is dealt with at this time: the **Add_Prediction()** routine can read the predictor both in frame or field mode, and the result is written in the same fashion. Table 2 shows the number of clock cycles needed for this function in both frame/field cases and its code size.

Table 2 : Add prediction (code size and cycles)

Function	Code Size (Bytes)	Cycles/8x8 Block Frame DCT Coding	Cycles/8x8 Block Field DCT Coding
Add_Prediction()	612	117	115

2.3 - Huffman Decoding/Inv Quant/Inverse Scan

This is the function that takes from the input bitstream the Huffman variable length codes of the data coefficients for one 8x8 block and decodes them. Then, inverse quantization and reverse zig-zag/ alternate scan are applied. Another task performed by this function, is to clear the result block before starting decoding. This is needed since the decoding process affects only the non-zero coefficients. Four of these functions are needed to build a Main Profile decoder, that is MPEG-1 backward compatible. In particular, we need different functions for intra and non-intra blocks, since they use different Huffman tables. MPEG-1 backward compatibility requires to deal with a different kind of escape codes, whose peculiarity is the fact that they do not have a fixed length (there is a sort of "escape in escape" increasing the level of difficulty).

The computational cost of the four functions cannot be given as standalone, since it depends on the number of symbols that are decoded. The only interesting figure is the number of cycles/symbol, that is the same for all the four functions (Table 3).

Table 3 : Huffman decoding/Inverse quantization/Inverse scan

Function	Code Size (Bytes)	Cycles/Symbol
Decode_MPEG1_Intra_Block()	1272	24
Decode_MPEG1_Non_Intra_Block()	1040	24
Decode_MPEG2_Intra_Block()	1184	24
Decode_MPEG2_Non_Intra_Block()	896	24

All these functions are located in the module **getblk.st1**. The Huffman look-up tables needed for these functions takes 1732 bytes (read-only data memory).

2.4 - Temporal Prediction

This function is basically made of three parts: the memory read from the external frame buffer, the computation of horizontal and/or vertical interpolation and/or the average with the previously computed predictor in case of bidirectional interpolation or dual prime prediction.

The first task is left to a DMA, that will be programmed with the address of the predictor just after the macroblock header decoding, based on the prediction type information and on the motion vectors. For P skipped macroblocks, the DMA will directly copy the predictor into the current frame buffer.

So, the profiling reported here only includes the computational cost of the interpolation and/or average. An efficient implementation should envisage the use of a "powered" DMA, able to make itself this task, in particular to cope with B skipped macroblocks. For these on, a simple memory copy is not enough, since a B skipped macroblock inherit the prediction mode and the vectors from the previous one, whereas a P skipped macroblock is simply patched with the corresponding one in the previous image.

Anyway, the software complexity of this task is computed here for the sake of completeness and for an eventual implementation on a small video format. There are several possible combinations of interpolations/average. The following Table 4 gives the number of clock cycles needed for a 16x16 luminance macroblock computation, in the different cases. The function is located in the module **motion_comp.st1** and its name is **form_component_prediction().**

Table 4 : Temporal prediction : the different cases

Operation	Clock Cycles For 16x16 Luminance
Simple copy	125
Copy with average	240
Vertical interpolation	274
Vertical interpolation + average	434
Horizontal interpolation	271
Horizontal interpolation + average	431
Vert+Hor interpolation	623
Vert+Hor interpolation + average	815

The code size for this function is of 5524 bytes. A read-only table of 48 bytes is needed for the jump at the beginning (`switch` structure). The algorithm for the chroma components is the same, but the cost is reduced by a factor 4 for each component.

2.5 - Bitstream Parsing

This block represents a set of small functions that are used basically to decode the macroblock headers (containing information about the contents of the macroblock and the motion vectors). These functions are located in the modules **mb_headers.st1** (*macroblock_modes(), Get_coded_block_pattern(), Get_macroblock_address_increment()*) and **motion.st1**(*motion_vector()* and *motion_vectors()*).

The code size of these functions amounts to 2336 bytes. The data memory needed for the Huffman tables is of 584 bytes. The computational cost of these functions cannot be provided as standalone, since it depends on the bitstream to be decoded.

3 - PERFORMANCE OF THE DECODER

In this paragraph are presented and analyzed the performance measures of the Main Profile decoder in a real case. The sequence considered is made of 25 frames of *renata* (in PAL format), coded with M=3 (distance between consecutive P images) and N=12 (distance between consecutive I images). The encoder used to generate these test bitstreams uses recursive motion estimation. Several different bitrates have been considered, also in order to understand the dependency of the functions from the bitrate (Table 5).

Table 5 : Profiling of the code (Mcycles/s) for PAL format

Function \ Bitrate	2 Mbps	4 Mbps	10 Mbps	15 Mbps
Wang IDCT	29.8	46.5	85.3	94.8
Add Prediction	0.5	3.5	8.9	10.4
Temporal prediction	26.6	29.5	30.2	30.2
Huffman dec/IQ/inv scan	8.1	18.2	47.8	70.4
Others (parsing, main)	21.2	22.1	22.6	22.6
TOTAL	**86.2**	**119.8**	**194.8**	**228.4**

Once measured these data, the computational cost for the different video formats is obtained proportionally (Table 6). The bitrate indicated is referred to PAL format, and it must be scaled for the other formats.

Table 6 : Different video formats (Mcycles/s)

Resolution		PAL@2Mbps	PAL@4Mbps	PAL@10Mbps	PAL@15Mbps
PAL	(720x576)	86.2	119.8	194.8	228.4
2/3D1	(544x576)	64.65	89.85	146.1	171.3
1/2D1	(352x576)	43.1	59.9	97.4	114.2
CIF	(352x288)	21.55	29.95	48.7	57.1

Table 7 shows a summary of the code size of the different modules.

Table 7 : code size of the complete application

Module	Code Size (Bytes)	Code Type
Huffman decoding	4392	SLIW
Wang IDCT	2100	SLIW
Add prediction	612	SLIW
Temporal prediction	5524	SLIW
Parsing	2336	GP32
Headers	8736	GP32
Others	11136	GP32
TOTAL	**34836**	**SLIW/GP32**

The total size does not include the I/O libraries (printf, fread, fwrite etc.), the functions used for dma emulation (module **dma.c**), the functions used to store to file the decoded sequence (**store.c**) and the startup code (**crt0.o**-456 bytes).

Table 8 shows a summary for data memory usage.

Table 8 : Data memory usage

Data type	Memory size (bytes)
Stack	3072
Bitstream buffer	2048
Current macroblock	384
Predictor	724
Huffman tables	2316
Others (variables)	5108
TOTAL	**13652**

4 - REFERENCES

– [1] ISO/IEC 13818-2 *Recommendation ITU-T H.262*, Standard Video MPEG-2, 1995

– [2] S.Eckart, C.Fogg, *MPEG-2 Encoder/Decoder Version 1.2,* July 1996, Copyright 1996, MPEG Software Simulation Group (http://www.mpeg.org/MSSG)

5 - ACRONYMS AND DEFINITIONS

I - images : Intra-Frame encoded images
P - images : Temporally-Predicted images
B - images : Bidirectionally-Interpolated images
M : Distance between consecutive P-images
N : Distance between consecutive I-images
DMA : Direct Memory Access device
(I)DCT : (Inverse) Discrete Cosine Transform
IQ : Inverse Quantization