M.Gurunadha Babu, M.Venugopal, M.Phaniraj Kiran / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp. 399-403

Implementation of H.264 Decoder On Arm11 Mpcore

M.Gurunadha Babu,M.Venugopal, M.Phaniraj Kiran
Professor & HOD,ECE dept. Chilkur Balaji Institute of Technology,Hyderabad
Emebedded software labs,Hyderabad
Managing Director R&D, Consultant

Abstract

The power dissipation in embedded processor core is an important parameter while increasing the process performance. With the advent of VLSI technology and the latest tools to design system on chip, the complexity, design time and verification time can be reduced. To increase the total processor performance, both the multiprocessor and multi threading systems are used for any application.ARM overcomes such design issues faced in the industry for any application. In a Multiprocessor design, each processor can use the uni-processor power management techniques, such as clock gating, keeping the processor in standby mode and voltage and frequency scaling. But it also has the ability to turn entire processor off to save all their consumed power while still executing lower demanding application workloads. One of the applications considered in this paper is implementation of H.264 video decoding using ARM MP Core ARM11.In H.264 video decoding implementation, the computational standard is two or three times higher than that of H.263 and MPEG-4 standard. The complex parts of the decoder in H.264 are the operation of entropy coding and De-blocking filters in the decoder. The reduction in computational complexity an algorithm is implemented for H.264 decoder. It contains a group-based CA VLC decoding method for H.264 entropy code tables and de-blocking filter.

Keywords- H.264; CAVLC; De-blocking Filter;ARM11

I. Introduction

ARM MPCore uses ARM11 micro Architecture and it can be configured to have one to four processors and can deliver very high performance. The power consumed by the above design using multiprocessing technique is the important parameter during the application development. In a Multiprocessing design, each processor can use the uni-processor power management techniques, such as clock gating, standby mode and voltage and frequency scaling. Also the entire processors can be off to save all their consumed power. Multiprocessor design such as ARM11 MPCore further reduce the cost of system development by delivering a multiprocessor can be considered as a single, configurable macro block. This block supports standard operating systems that are able to fully utilize the processor architecture any technical / legal issues.

ARM11 MPCore can resolve a cache miss, or access to shared data and aroung 50% faster than a processor and could resolve data from a shared Level 2 cache. The scalable performance is very efficient in ARM11 MPCORE.

The ARM11 MPCore processor provides software portability across single CPU and multi-CPU designs. It provides an enhanced memory throughput of 1.3Gbytes/sec from a single CPU, and a multiprocessor solution delivers greater performance at lower frequencies than comparable single processor designs, offering significant cost savings to systems designers. The ARM 11 MPCore processor also simplifies otherwise complex multiprocessor design, reducing time-to-market and total design cost.

The ARM 11 MP Core processor supports a fully coherent data cache, the designer has a flexibility with respect to various symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP), or any of this combination. The MP Core processor increases a solution's performance via the ability to cache shared data, increases system response and allow workloads to be balanced between different processors with portable multitasked applications and allow scalability using efficient processor utilization for multithreaded applications, for example H.264 video coding.

The ARM 11 MP Core processor can be configured differently for various design requirements.

The ARM 11 MP Core processor supports the ARMv6 architecture, with SIMD media extensions for next-generation rich multimedia and convergent devices. The processor supports Adaptive Shutdown of unused processors to give dynamic power consumption as low as 0.49 mW/MHz from a generic 130nm process excluding cache. ARM Intelligent Energy Manager (IEM) can further reduce consumption to as low as 0.30mW/MHz by dynamically predicting the required performance and lowering the required voltage and frequency.

The ARM 11 MP Core enables System on chip (SOC) designers to view the core as a single

"uni-processor", simplifying SOC development and reducing time-to-market.

The Core area, frequency range and power consumption are dependent on process, libraries and optimizations.

The H.264 video coding standard provides enhanced coding efficiency for a wide range of applications. The computational complexity of the H.264 is two or three times high compare to that of H.263 and MPEG -4. This computational complexity causes problems in developing H.264 based video solutions.

The H.264 is widely implemented standard for video compression, A much simpler and less efficient Baseline Profile is used for visual communication applications today. The original H.261 standard was replaced by H.263 and followed by the latest H.264. All of these standards essentially describe how to compress video, so that it can be transmitted efficiently across real-world networks, and how to decompress it at the receiver side. A raw (uncompressed) HD video stream can be around 1 gigabit per second and clearly not appropriate for any but the few over-provisioned research networks. By applying the H.264 Baseline Profile, the 1 gigabit per second stream can be compressed to about 1 megabit per second, i.e., 1000-fold compression. However, if H.264 High Profile is used instead, the compressed video stream can be reduced to about 512 kilobits per second, i.e., 2000-fold. Figure 1 is a block diagram of the H.264 standard (encoder side).

This paper is organized as follows. In introduction part we explained the MPCore architecture features on which the H.264 video encoding application is implemented. In Section II, related work by various authors is explained. In section III, we simply review the principle of context adaptive variable length coding (CAVLC) decoding and a new group-based lookup table algorithm is introduced. In Section IV, the principle of the deblocking filter is first introduced, and multimedia frame work is explained, and a new optimized deblocking filter algorithm is then proposed. AT the end results are shown at V and conclusion will be given in Section VI.

II. RELATED WORK

Thomas Wiegand [1] proposed two entropy coding methods applied in H.264/AVS and are termed CAVLC and CABAC. Both use context-based adaptivity to improve performance relative prior standard design. Garry J.Sullivan [2] have proposed a New High profiles defined in the FREXT amendment. This technique is a better decoding method. Yen-Lin Lee, Truong Q. Nguyen [3] have proposed an analysis on efficient architecture design for VC-1 overlap smoothing and In-Loop De-blocking filter. In this paper the author analyzed the behaviour of VC-1 filters and presented several efficient methods and integrated architecture design. This paper also proposed two efficient methods. Multiple processing order and modified chrominance processing order, which greatly reduces external memory, cycles and on-chip memory size for filtered and temporal reconstructed pixels. Yen-Kuang Chen, Eric Q.Li [4] have proposed implementation of H.264 encoder and decoder on personal computers. The author proposed analysis of softare implementation of H.264 encoder and decoder on general purpose processors with media instructions and multithreading capabilities. This technique explains how to optimize the algorithm of H.264 encoder and decoders on Intel Pentium 4 processor. Chia-Cheng Lo [5] have proposed a technique to combine two entropy decoding methods defined in the H.264 standard i.e. context-based adaptive binary arithmetic coding (CABAC) and context-based adaptive variable length coding (CAVLC).

III. THE OPTIMIZED CA VLC DECODING

Most visual communication systems today use Baseline Profile. Baseline is the simplest H.264 profile and defines, for example, zigzag scanning of the picture and using 4:2:0 (YUV video formats) chrominance sampling. In Baseline Profile, the picture is split in blocks consisting of 4x4 pixels, and each block is processed separately. Another important element of the Baseline Profile is the use of Universal Variable Length Coding (UVLC) and Context Adaptive Variable Length Coding (CAVLC) entropy coding techniques.

The Extended and Main Profiles includes the functionality of the Baseline Profile and add improvements to the predictions algorithms. Since transmitting every single frame (think 30 frames per second for good quality video) is not feasible if you are trying to reduce the bit rate 1000-2000 times, temporal and motion prediction are heavily used in H.264, and allow transmitting only the difference between one frame and the previous frames. The result is spectacular efficiency gain, especially for scenes with little change and motion.

The High Profile is the most powerful profile in H.264, and it allows most efficient coding of video. For example, large coding gain achieved through the use of Context Adaptive Binary Arithmetic Coding (CABAC) encoding which is more efficient than the UVLC/CAVLC used in Baseline Profile.

The High Profile also uses adaptive transform that decides on the fly if 4x4 or 8x8-pixel blocks should be used. For example, 4x4 blocks are used for the parts of the picture that are dense with detail, while parts that have little detail are transformed using 8x8 blocks.

H.264 decoder block diagram as show in figure 1.

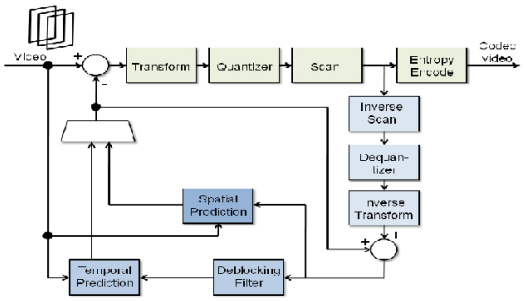

Figure 1: H.264 Encoder block diagram

As shown in figure 1 CA VLC is used to code residual data, which is obtained after transform and quantization. The residual data is usually sparse with a large number of zeros and is run length encoded. The coefficients obtained after run length coding are sent using CA VLC. In CA VLC coding, the total number of non-zero coefficients and the number of trailing ones are coded into a single variable length code. The H.264 uses the lookup tables method depending on the context for coding above mentioned elements. CA VLC decoder selects the tables depending on the number of nonzero coefficients in neighbouring blocks.

The different stages in H.264 which are implemented on Hardware are as follows.
(1) coefficient token decoding process
(2) Sign of Tis decoding process
(3) level decoding process:
(4) total zeros decoding process
(5) run_before decoding process

IV. THE OPTIMIZED DEBLOCKING FILTER METHOD

In H.264/AVC standard, the de-blocking filter is divided into 3 parts: The boundary strength of the filter depends on the coding modes of neighbour blocks in the current filtering edge. The filtering decision analyzes whether the filtering should be switched off. And the filtering operation applies on each 4 X 4 luminance and chrominance block edge on a macro block basis, because the transform coding is operated on 4 X 4 blocks.
(1). Boundary Strength (Bs) Decision: Boundary strength decides the strength of the de blocking filter operation, which is associated with each block.
(2). Filtering Decision: When Boundary strength equals to zero, no filtering takes place for edges. For non-zero Boundary strength values, a gradient like analysis is done and decides when the filtering to be switched off.
(3). Filtering Operation: The edge filtering starts to filter when the input pixels, boundary strength and threshold variables are ready.

Software Architecture:

When multi media player uses software decoder, performance problem is often an issue. CMM (Codec memory management) driver helps to improve rendering performance.
In common multimedia player, the decoded YUV data is transferred to video memory using memcpy(). It decreases much performance when the resolution of the movie is large. The CMM driver provides the interface to transfer decoded YUV data to video memory directly. At first, it allocates virtual addresses to the player. The virtual address is surely cacheable area. So software decoder can utilize cache. After decoding, the player request for CMM to flush cached area. The player request physical address of YUV buffer to CMM. With the physical address, the player calls the video drive API for rendering. YUV data is transferred to Hardware post pre-processor by DMA.

It does not only reduce memcpy() time, but also make player to decode and render at the same time. As rendering is done by only Hardware, decoding performance is not decreased. So we should make decoding and rendering as multi-threaded.

There are two methods to render YUV data. The one is using local path between Hardware post processor and LCD. It does not posses data BUS. But local path only support RGB888. The other is using DMA between post processor and LCD.

Implementation of the Multimedia Framework on ARM11
To achieve the desirable functions (playing the streamlined Video) , we need to the following work.
1) Porting of linux OS on to ARM 11 architecture and Porting some Gstreamer required open-source libraries to Linux , such as Glib, Liboil, etc.
2) According to Gstreamer framework, write two plug-ins. One is used to send the decoded original video data from Gstreamer to Linux display system.
3) Based on Gstreamer, we should construct a streaming player which can be used in Media Server guard process to supply media player service.
4) Modify assembly codes to apply Gstreamer to diversified CPU architectures, and apply some commonly used optimized technologies in embedded environment to Gstreamer.
As the porting work of Glib, Libiol and other related libraries are easy, here we introduce the rest steps.
The Data flow of this is pictorially explained in the figure 2.
The basic Data flow is described as CMM driver here is codec memory management driver. YUV Buffer is used to capture the video information from the Media file.
Post processor is basically needed for Scaling and CSC which basically populates on LCD.

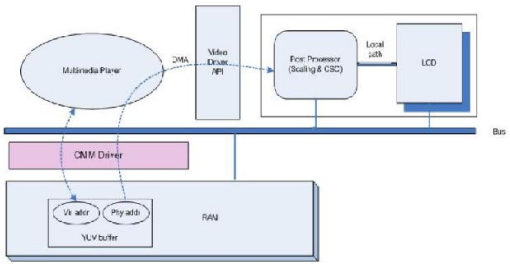

Figure:2. Data Flow diagram

Direcory Structure:

Direcory	Files	Description
/CMM_APP/	*.C,*.h	CMM test file
?CMM_DRV/	*.C,*.h	CMM Device driver file
/doc	*.doc,*.pdf	CMM document

How to test CMM:
Kernel Build
Before kernel compilation, you must setup memory layout as shown below in "include/ asm_arm/ arch_S3C2410/ reserved_mem.h"file
Default reserved memory size
MFC : 6MB
POST : 8MB
JPEG : 8MB
CMM : 8MB
Camera : 15MB

- The above sizes can be modified

#define CONFIG_RESERVED_MEM_JPEG
#define CONFIG_RESERVED_MEM_JPEG_POST
#define CONFIG_RESERVED_MEM_MFC
#define CONFIG_RESERVED_MEM_MFC_POST
#define CONFIG_RESERVED_MEM_JPEG_MFC_POST
#define CONFIG_RESERVED_MEM_JPEG_CAMERA
#define CONFIG_RESERVED_MEM_JPEG_POST_CAMERA
#define CONFIG_RESERVED_MEM_MFC_CAMERA
#define CONFIG_RESERVED_MEM_MFC_POST_CAMERA
#define CONFIG_RESERVED_MEM_JPEG_MFC_POST_CAMERA
#define CONFIG_RESERVED_MEM_CMM_MFC_POST
#define CONFIG_RESERVED_MEM_CMM_JPEG_MFC_POST_CAMERA

The following is the CMM driver module compilation procedure.
Node Name:/dev/misc?S3C_CMM
Major number: 10
Minor number: 250
1.The procedure to make device node
[root@local host CMM]#mknod/dev/misc/S3C-cmm c 10 250
2. module compilation
[root@local host cmm-drv]#make
Testing application compilation
[root@local host cmm_app]make
Now insert module and execute binary on target side.
The following commands are executed on target side.
[root@Samsung cmm_drv]insmod S3C_cmm.ko
[root@Samsung cmm_drv]cd../cmm_app/
[root@Samsung cmm_app]./cmm_test

V. Results

In our paper we have considered basic Display application which includes H.264 display, MPEG4 display, H.263 display, VC-1 display and 4 windows display (H.264,MPEG,H.263,VC-1)
The output figure 3 shows H.264 file decoder test for embedded Linux V 0.1, the decoded video output and basic display data flow diagram.

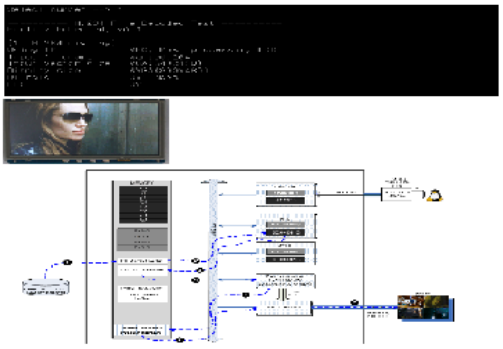

Figure: 3. H.264 file decoder test output

VI. Conclusion

In our paper, we consider several different kinds of video container formats. For each video container format, we take several different video clips with different rates, decoding standards and definitions. Initailly Embedded Linux OS has been ported to ARM , in this experiment section, we take S3C6410 ARM processor as an example. And the specific parameters are: ARM11, Linux 2.6.24 is ported, Enabling the Hardware decoder of ARM 11 S3C6410 architecture.

REFERENCES

[1] Thomas wiegand "Overview of the H.264/AVC video coding standards", IEE Transactions on circuits and systems for video technology vol. 13 No. 7 July 2003.

[2] Garry J.Sullivan "New Emerging standard H.264/AVC", SPIE Conference on applications of Digital image processing XXVII special session August 2004.

[3] Yen-lin lee, Truong Q. Nguyen "Analysis and efficient architecture design for VC-1 overlap smoothing and In-loop de-blocking filter", IEE transaction on circuit & systems for video technology-TCSV, Vol.18, No.12, PP-1786-1796.

[4] Yen-kuang chen, Eric Q.Li, "Implementation of H.264 encoder and decoder on personal computers", Journal of visual communication and Image representation – JVCIR Vol. 17, No.2. PP-509-532, 2006.

[5] Chia-Cheng Lo, Shang-Ta Tasai, Ming-Der Shien "Reconfigurable architecture for entropy decoding and inverse transform in H.264", IEE Transactions on consumer Electronics – IEE TRANS CONSUM ELETRON, Vol. 56, no. 3, PP 1670-1676, 2010.