International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967 *Volume 2 , Issue 1(Jan-Feb 2015), PP1-5*

Massively Parallel Architecture for Video Encoding

Darshan Pawar[1], Sagar Karwande[2], Kochmuriyil Jancy James[3], Renuka Joshi[4], Chandrama Thorat[5]
[1, 2, 3, 4, 5] *(Department of Computer Engineering, JSPM's Rajarshi Shahu College of Engineering, India)*

Abstract:- The fast moving world has fast moving demands and today's technologies have shifted their focus on satisfying these demands. Fast and storage efficient video streaming and downloads is one such demand. So the industry is focusing on having more resolutions in the video data to have greater quality with reduced space. To meet the storage demand of such videos, video-compression is the most needed solution. This paper proposes the concept of massively parallel architecture of CUDA for reducing the complexity involved in video compression. The complexity of video compression lies in the calculation of estimation of the motion in the video which can have a large extent of parallelism. The standard used here for this research is the most popular and widely used H.264/AVC standard.

Keywords:- CUDA, GPU, Macroblock, Motion Estimation, Motion Compensation, H.264

I. INTRODUCTION

The video compression is the process of reducing the amount of data which represents a digital video. This is done to make the video compatible for storage as well as for transmission. There have been many compression standards evolved to have a common platform to incorporate the processing of videos. H.264 is one of these and is jointly published by International Telecommunications Union (ITU) and the International Standards Organization (ISO). This encoding standard is so far the most popularly used in almost all of the video encoders. This standard is best known for its higher compression and higher quality as compared to the traditional old encoding standards. This higher compression and quality sometimes becomes a bottleneck in terms of the time taken by the encoder to compress the video. This paper proposes an approach which can effectively reduce the time taken by the encoder of H.264 standard using CUDA. The most computationally expensive process of estimation of the movement or the motion in the video can be offloaded on the GPGPU. This reduces the overall time taken for the encoding process.

II. LITERATURE SURVEY

1.1. Video Compression

The Video compression is the process of removing the redundancy of data inside the bitstream of a video signal. The redundancy can be of two forms: spatial redundancy and temporal redundancy. Spatial redundancy is the intra-frame redundancy where there is redundancy in the frame itself whereas temporal redundancy is the inter-frame redundancy which is the redundancy between the adjacent frames. The aim of this paper is to explore and remove the temporal redundancy of the video by means of the process of motion estimation.

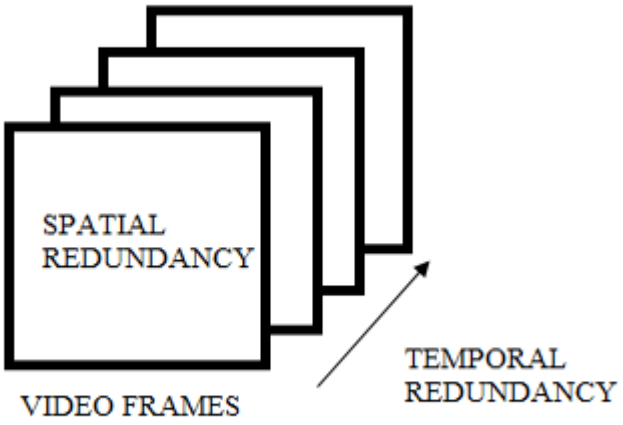

Fig. 1 Video Frames With Specification Of Spatial And Temporal Redundancy

1.2. Macroblock

Each frame is divided into two dimensional units of pixels called as macroblock. Each macroblock has a maximum dimension of 16x16 and a minimum dimension of 4x4. These macroblocks are used to remove the redundancy within them individually.

1.3. Motion Estimation

The Motion estimation of a macroblock involves finding a 16x16 sample region in a reference frame that closely matches the current macroblock [1]. Here the reference frame is the frame which is already an encoded frame. For searching the best macroblock in the reference frame a search area is defined which is centred on the current macroblock to be searched. This paper proposes an exhaustive search method which can be offloaded on the GPGPU of NVIDIA.

1.4. Motion Compensation

The motion compensation is the process of subtracting the luma and chroma samples of the selected best matching region in the reference frame from the current macroblock, which produces a residual macroblock that is encoded and transmitted together with a motion vector describing the position of the best matching region relative to the current macroblock position [1].

1.5. CUDA

The CUDA comes with a highly programmer-friendly environment which can be easily used for the development of many multi-core applications. The highly light-weighted multithreaded architecture of the GPUs and its support for multiple cores programming makes it suitable for solving a large computationally extensive problem which would take many days or months to solve in a sequential manner.

The reason behind the discrepancy in floating-point capability between the CPU and the GPU is specialized for compute intensive, highly parallel computation – exactly graphics rendering is about – and therefore designed such that more transistors are devoted to data processing rather than data caching and flow control [2].

Because of its scalable architecture, CUDA has driven the market of multi-core programming towards itself in such a way that it has been integrated in most of today's parallel programmed applications.

II. PROPOSED SYSTEM ARCHITECTURE

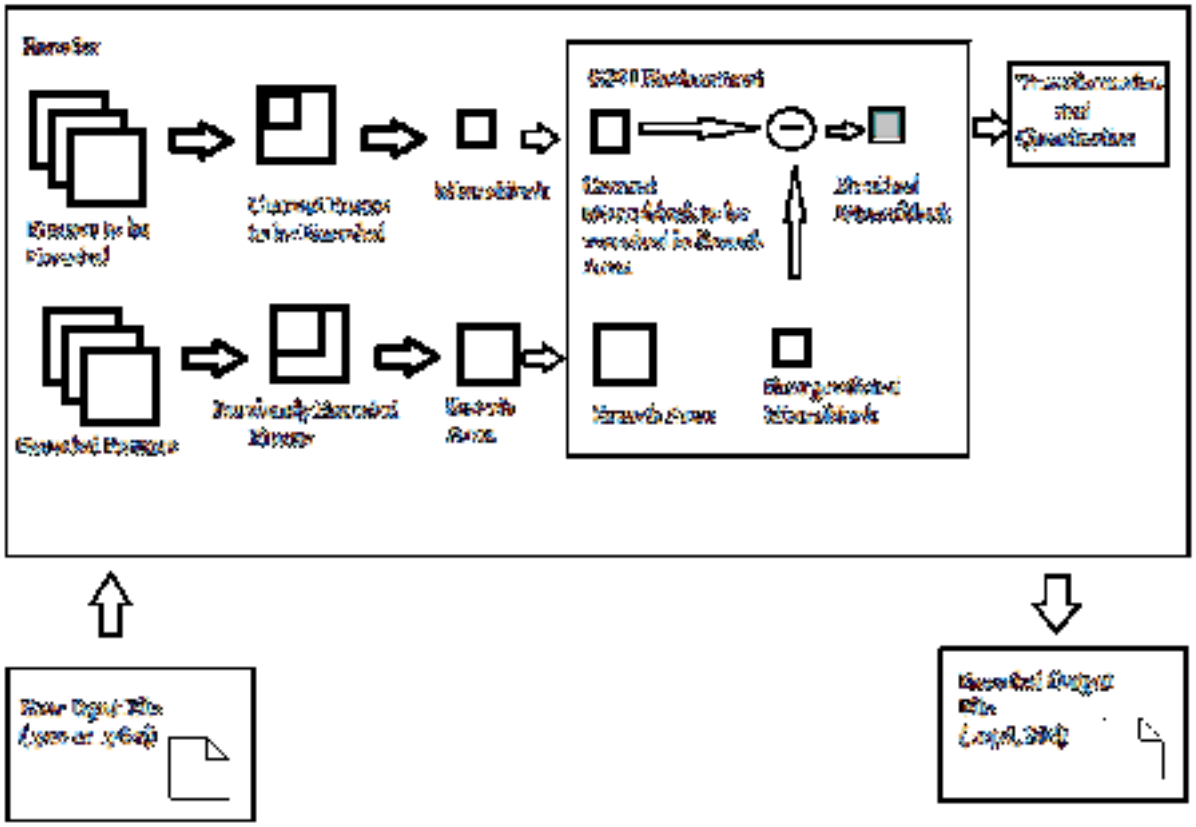

Fig.2. Proposed System Architecture

As shown in the fig. 2, the proposed architecture consists of the input and output file and the encoder. The proposed encoder is to be designed in such a way that it can be run on the hardware consisting of the GPU environment.

Each digital video consists of a series of frames which are sampled temporally. This video can be encoded frame by frame. Each frame is divided into some set of macroblocks. Each macroblock is individually encoded by means of considering a search region or area in the reference frame. So the search region comprises of many macroblocks which are individually checked with the current macroblock and the macroblock in the search area with the minimum Sum of Absolute Difference (SAD) is considered as the best match. The formula for SAD calculation can be given as:

$$\text{SAD} = \sum_{i=0}^{m-1} \sum_{k=0}^{n-1} (Cand(i,k) - Ref(i,k)) \quad (1)$$

Here, Cand(i,k) are the current macroblock values which are subtracted from Ref(i,k) i.e. referenced macroblock values for the size of mxn macroblock.

The macroblock found to have the least SAD value is chosen as the best predicted macroblock from the search range and is subtracted from the candidate or current macroblock to get a residual macroblock which is further transformed and quantized and encoded to form a bitstream which can be either stored or transmitted over the network. The algorithmic design is left to the implementer who can choose the best of the algorithms for parallelism from the ones available namely Full Search, Diamond Search, Hexagonal Search, etc. [3][4].

The main module of parallelism can be achieved by means of calling a kernel launch of GPU for finding the minimum SAD value for the candidate macroblock.

This experiment is estimated to achieve a very good performance in terms of the time required for motion estimation.

III. CONCLUSION

The purpose of the paper was to give a brief idea about the area in which the parallelism can be achieved for video compression to reduce the latency involved while doing it. The implementation details can be further studied with these references to define the scope and behaviour of the project. The only issue that becomes a bottleneck in H.264 video compression is the time required for the process of encoding. This can be eliminated by using the unconventional but new techniques of using parallelism via GPGPU and CUDA which are easy to understand and apply.

IV. ACKNOWLEDGEMENTS

We have carried out this research project with the support of the computer department of Rajarshi Shahu College of Engineering, Pune. We would like to place on record and express our gratitude to our project guide Ms. C.G.Thorat, our project coordinator Ms. V.M.Barkade and our Head of the Department Dr. A.B.Bagwan. Without their wholehearted support and guidance this project would not have been possible.

REFERENCES

[1] E.G. Richardson, Iain, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia* (John Wiley & Sons Ltd., 2003).

[2] NVIDIA CUDA™, *NVDIA CUDA C Programming Guide version 4.0*

[3] Ce Zhu, Xiao Lin, and Lap-Pui Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation", *IEEE Transactions on Circuits and Systems for Video Technology, Volume: 12, Issue: 5,* May .2002, 349-355.

[4] S. Zhu, K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation*", IEEE Transactions on Image Processing, Volume: 9, Issue: 2, Feb 2000, 287-290.*