Journal of Theoretical and Applied Information Technology, Islamabad PAKISTAN

31st May 2010. Vol.15. No.2.
© 2005-2010 JATIT. All rights reserved

www.jatit.org

COARSE GRAINED RECONFIGURABLE ARCHITECTURES FOR MOTION ESTIMATION IN H.264/AVC

[1]D.RUKMANI DEVI , [2]P.RANGARAJAN ^, [3]J.RAJA PAUL PERINBAM*
[1] Research Scholar, Department of Electronics and Communication Engineering, RMK EC, Chennai, India-601206
[2]Prof & Head, Department of Information Technology, RMD Engineering College, Chennai, India-601206
[3] Professor, Department of Electronics and Communication Engineering, RMK EC, Chennai, India-601206

ABSTRACT

This paper presents a brief survey on coarse grained Reconfigurable architectures for motion estimation. The motion estimation processor demands a very large amount of computing power in recent video coding standard H.264/AVC. To reduce computational complexity without affecting image quality, Motion estimation algorithms are implemented on various course grained Reconfigurable architectures such as Morphosys, Matrix, Rapid, Mora and Chess. The purpose of this study is to compare these architectures based on Granularity, resource utilization, memory bandwidth and computation model. The methodology and results presented here provide useful guidelines to system designers in selecting coarse grained Reconfigurable Architectures for motion estimation.

***Keywords: Coarse** grained, Reconfigurable architectures, motion estimation, H.264.*

1. INTRODUCTION

There is an increasing demand for multimedia processing solutions through flexible and highly parallel architectures.H.264 [1] video compression standard plays an important role in today's consumer market. Motion estimation (ME) (Figure.1) is a key technique in most algorithms for video compression. It is one of the most computational intensive subroutines of H.264. Compared to fixed block-size ME (FBSME), H.264 supports VBSME [2] which provides better estimation of small and irregular motion fields and allows better adaptation of motion boundaries resulting in a reduced number of bits required for coding prediction. In motion estimation, each frame of a video sequence is divided into fixed number of non-overlapping square blocks. For each block in current frame, best matching block is searched within the previous frame. In most block matching algorithms, the sum of absolute differences (SAD) is used as the main metric. VBSME requires support for 7 block patterns: 16x16, 16x8, 8x8, 8x4, 4x8, and 4x4. Supporting this feature is a challenging task in terms of resource utilization when implementing the application on hardware. Due to its highly parallel nature and algorithm's demand for a flexible solution, a reconfigurable architecture poses as an ideal candidate to respond to this compute intensive routine.

Based on the historical perspective, implementation of motion estimation has evolved through general purpose processors [3] [4] [5], ASIC [5] [6] [7] [8] [9], FPGA [10], and coarse grained reconfigurable architectures [11] [12] [13] [14] [15] [16]. Existing architectures either only support FBSME or implement VBSME with redundant hardware. While new processor advances including VLIW, SIMD, and out of order execution, have provided some advances in exploiting parallelism within applications, the processor's inherently sequential and generic architecture limits the ability to efficiently exploit potential parallelism within highly concurrent types of algorithms. Alternatively, application specific instruction set processors (ASIPs) allow designers to custom the microprocessor by adding custom instructions and execution units within the processor. However, such extensions are similar to VLIW or SIMD approaches and are still limited to the data access through the processor's register file.

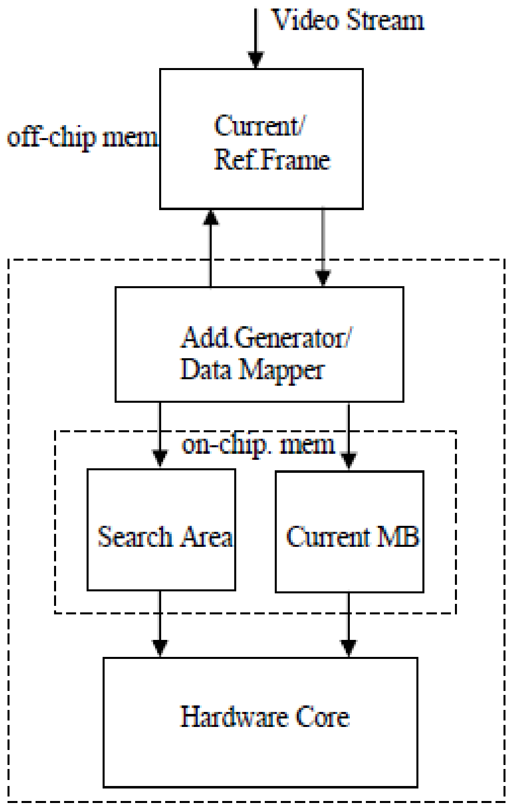

Figure.1. Block Diagram of Motion Estimation Processor

Several ASIC based approaches have been proposed for variable block size block matching algorithms to reduce computational complexities. However some of these architectures [5] [6] [7] are not capable of processing all the block sizes specified by the H.264. As an alternative, Yap [8] promises to support all block sizes. Architecture is formed of 16 processing elements (PEs) interconnected as a 1D systolic array where each PE computes a4x4 SAD. As an improvement, Ou [9] introduces a hierarchical 1D systolic architecture that employs partial SAD computation technique. Additionally in [9], a “*VBSME processor*”, based on an adder tree structure, supports all block sizes. This paper [9] achieves lower latencies with higher throughput compared to existing VBSME architectures. However partial SAD computation requires delay registers and extra accumulators; and “*VBSME processor*” consists of a fixed set of redundant adders. Therefore this architecture has high area overhead which can be improved by using a flexible routing architecture.

Coarse-grained reconfigurable architectures RAPID [14], MATRIX [12], CHESS [13], RAW [11], Morphosys [15], MORA [16] have been introduced to overcome some of the drawbacks of lookup table based fine grained reconfigurable architectures, such as FPGAs. In general, coarse-grained reconfigurable fabrics are composed of high level processing elements (PEs) with generic reconfigurable interconnect network. While fewer configuration bits are needed for the PEs, fully utilizing the functionality of each PE is difficult; leading to significant under utilization of coarse grained fabrics.

2. ASIC BASED APPROACHES

ASIC based architectures can be broadly classified into different categories depending upon various metrics such as topology of processing elements and methodology for accumulation of SADs etc. Based on topology, the architecture can be classified as 1D and 2D systolic arrays. 1D architectures [8, 17, and 18] consist of 1D systolic array of processing elements. They are simpler in structure but use a large number of registers for storing partial SADs and thus suffer in area and high latencies owing to the sequential computation and accumulation of SADs. 2D architectures [6, 7] consist of processing elements connected in a mesh-based architecture and SAD computations are direct mapped. J.F. Shen *et al* .,[6] does not support block sizes smaller than 8x8. This architecture uses a smaller 2D array. So, the partial SADs are calculated and added sequentially. This results in greater latencies. De Vos *et al.*, [7] do not support VBSME. It uses a large number of storage registers to store reference pixels. Also, loading of reference pixels in the propagation registers causes long latencies.

Based on methodology for accumulation of SADs, the architectures can be classified as partial and parallel sum SADs. In partial sum SAD architectures [6], reference pixels are broadcasted and SAD computation for each 4x4 sub block is pipelined. In this architecture, each processing element computes one pixel difference, accumulates it to the previous partial SAD and sends the computed partial SAD to the next processing element. This kind of architecture uses large number of storage registers due to the accumulation of partial SADs in each processing element. In parallel sum SAD architectures, all pixel differences for a 4x4 sub-block are computed concurrently and thus added in one clock cycle. In this architecture, reference pixels are reused between different processing elements which decrease memory bandwidth requirements. The direction of data transfer among different processing elements depends on the search pattern adopted. VBSME processor [9] consists of 16 separate SAD "*modules*" (Figure 2a) to process sixteen 4x4 motion vectors. It also consists of a chain of adders and comparators, (VBSME processor), to compute larger SADs. "*PE array*" (Figure 2b.) which forms the computation element of each SAD module is constructed by cascading four 1D arrays (Figure 2c). Each 1D array consists of a 1D systolic array of 4 PEs.

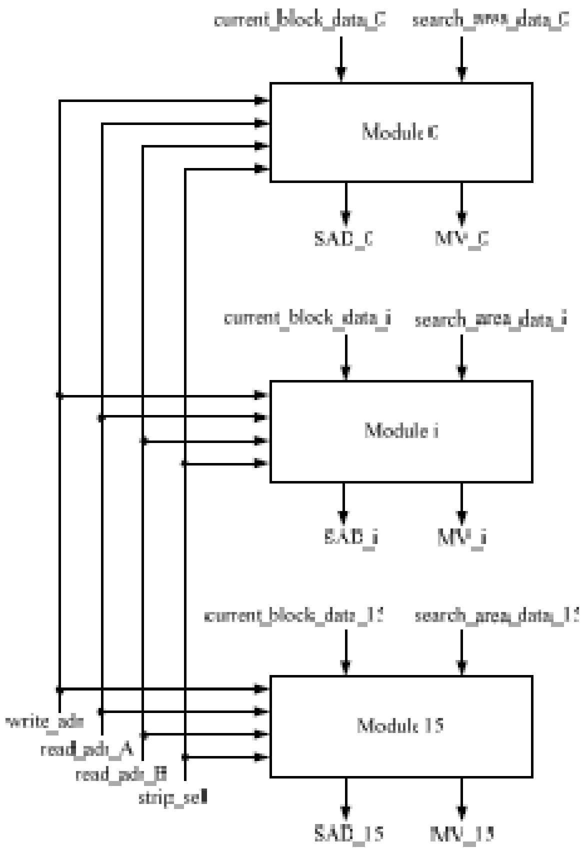

Figure.2a.SAD Modules

Each PE computes 1 pixel SAD. This circuit operates by scheduling the columns of the current 4x4 sub block "*current_block_data_i*" through a delay line, and broadcasting two sets of search block columns "*block_strip_A*" and "*block_strip_B*" on each clock cycle. Four block matching operations can be performed concurrently in one SAD module. The produced 4x4 SADs are then sent through a fixed series of adders and
comparators to produce 4x4 motion vectors. The 4x4 SADs are also sent in parallel to four sets of adders and comparators to produce 4x8, 8x4 SADs. The two 8x4 SADs are then again sent through a set of adders and comparators to form 8x8 SAD. 16x8, 8x16, and 16x16 SADs are computed similarly. This adder/comparator based chain of events form the adder tree structure.

An efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation [9] has separate SAD modules for 4x4 sub block computations with separate input ports for loading current block and search region data. Thus, it does not support reuse of search data between modules. This increases the amount of data transactions from the memory. Also, the VBSME is supported by a fixed set of adders based on a large adder tree leading to resource wastage. An intelligent routing scheme that uses same set of adders to compute different motion vectors each time is needed to overcome the wastage. Also, this architecture does not support other fast search algorithms like diamond search, hexagonal search etc.

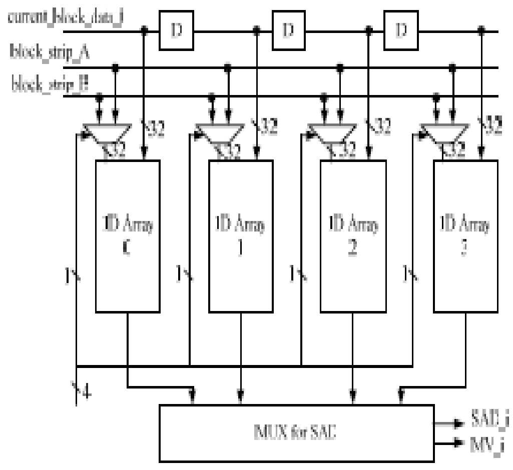

Figure.2b.PE Array

31st May 2010. Vol.15. No.2.
© 2005-2010 JATIT. All rights reserved
www.jatit.org

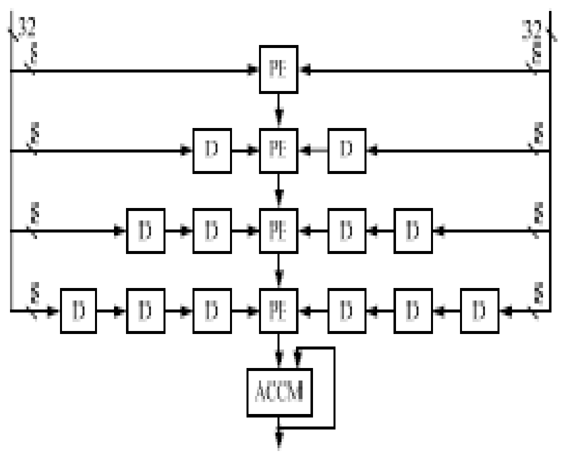

Figure.2c.1D Array

3. COARSE-GRAINED ARCHITECTURES

In general, flexibility is an important feature of reconfigurable devices, conventional FPGAs are simply too generic to provide high performance in many situations. General-purpose reconfigurable devices, while well suited to small or irregular functions, typically suffer a stiff penalty when implementing wide and complex arithmetic operations. These types of functions need to be built from too many small logical resources and end up being spread across too general a routing structure to be efficient. However, if the range of applications that a device is intended for is known beforehand, a designer can specialize the logic, memory and routing resources to enhance the performance of the device while still providing adequate flexibility to accommodate all anticipated uses. Common and complex operations can be implemented much more efficiently on specialized coarse-grain functional units while routing and memory resources can be tuned to better reflect the requirements.

Coarse-grained reconfigurable fabrics are composed of high-level functional blocks with generic reconfigurable interconnect network. In this section, a brief review and analysis of ChESS, RaPiD, MATRIX, Morphosys, and Mora architectures for implementing motion estimation algorithm. There is a need to choose carefully these architectures for an investigation based on granularity. They vary in granularity from 4-bit, 8-bit to 16- bit processing elements and buses and thus help in providing a broad view about mapping of motion estimation algorithm on coarse-grained architectures.

The ChESS [13] architecture is a reconfigurable arithmetic array targeted mainly for multimedia applications. The fundamental computation component is a 4-bit ALU with 16 instructions. The routing structure is based on 4-bit buses. Each ALU has a switchbox adjacent to it which serves as a cross point with 64 connections. Hence it needs about 64 bits to configure the switches and connections. Since each ALU has a corresponding switchbox associated with it the routing area consumes up to 50% of the total area. Mapping of the motion estimation algorithm on the ChESS architecture exploiting full parallelism, SAD computation for a 16x16 array would require a 512 ALU ChESS array.

MATRIX [12] is yet another coarse-grained reconfigurable computing architecture composed of 2D array of identical, 8-bit functional units overlaid with a configurable network. Each functional unit consists of an 8-bit ALU, memory and control logic. MATRIX also has a generic routing architecture. Mapping the motion estimation algorithm on the MATRIX architecture for a 16x16 block would require a 256 ALU MATRIX array. The MATRIX array with 8-bit functional units would require one functional unit for one pixel difference calculation. The performance result for motion estimation algorithm if mapped on a 256 ALU MATRIX array is [(M x 0.8M)/256 x 17 x 17] clock cycles considering a frame size of M x 0.8M. Also, support for VBSME in these architectures would involve huge routing complexity which is difficult to implement owing to the generic nature of routing architecture.

RaPiD [14] is a coarse-grained architecture mainly targeted for DSP applications. It consists of 1D array of functional units (ALUs, multipliers, Registers and RAMs). The complete RaPiD array contains 16 of these cells. The functional units (cells) in RaPiD are interconnected using a set of ten segmented buses that run the length of the data path. The buses in different tracks re segmented into different lengths. The implementation result for motion estimation as provided in the paper for a block size of 8x8 is 272+32M+14.45M2 clock cycles considering a frame size of M x 0.8M.

The implementation details show that the available parallelism in the system has not been exploited to the fullest extent. For a 16x16 SAD, only the row-wise differences are computed in parallel. The column wise differences are computed sequentially which decreases the performance to a great extent. This is because the linear array form of architecture fails to exploit this parallelism. Also there is huge underutilization of resources as one cell comprising of 3 ALUs is being used for computing just one difference per clock cycle. Assume that the second ALU is being used for adding the differences in a pipeline, the third ALU remains unutilized. Also, the ALUs are 16 bit, and the SAD computation involves 8 bit operation, which again adds to underutilization. Also, the reference frames considered are 8x8. So, the architecture does not support SAD calculation for smaller block sizes. The algorithm does not require ten buses for routing. So it leads to under utilization of routing resources too. Also, the real time video performance on a standard 720 x 576 image is about 12 frames per second using 100 MHz clock. So, the performance is quite poor with a huge dissipation of energy.

Morphosys [15] is a reconfigurable arithmetic array targeted mainly for multimedia applications. The main component of MorphoSys is the 8 x 8 RC (Reconfigurable Cell) Array. Each RC has an ALU-multiplier, a register file and is configured through a 32-bit context word. The ALU has 16-bit inputs, and the multiplier has 16 by 12 bit inputs. The ALU adder is 28 bits wide to prevent loss of precision during multiply-accumulate operation. Besides standard functions, the ALU has several additional functions e.g. absolute value of difference of two numbers and a single cycle multiply-accumulate operation. The controlling component of MorphoSys is a 32-bit processor, called Tiny RISC, based on [7]. Tiny RISC handles serial operations, initiates data transfers and controls operation of the RC array. Mapping the motion estimation algorithm on this array, 5304 cycles are required to finish the matching of the whole search area. If the image size is 352x288 pixels at 30 frames per second (MPEG-2 main profile, low level), processing of an entire image frame would take 2.1 x 10^6 cycles. At clock rate of 100 MHz for MorphoSys, the computation time is » 21.0 ms.

MORA (Multimedia oriented Reconfigurable array) [16] consists of a scalable 2D array of identical Reconfigurable Cells (RCs) organized in 4X4 quadrants and connected through a hierarchical reconfigurable network. The main building elements of the circuit are: a 256*8-bit *SRAM* acting as an internal data buffer, an 8-bit Processing Element (*PE*) and a *Control Unit* incorporating the *Configuration Memory*. The latter holds the control program of the RC, which is loaded during the "configuration phase" of the system. Each RC has two possible operative states: loading and executing. When the RC is in the loading state, data can be loaded through one or both the input ports and stored into the internal *SRAM*. The latter is dual-ported, thus enabling two independent read or write operations per clock cycle. As soon as all the required operands are available, the RC switches in the executing state.

4. CONCLUSION

This paper has attempted to give a survey on today's coarse - grained reconfigurable architectures (TABLE I) significantly relieving the burden of the computational complexity of motion estimation processor in H.264/AVC standard. Their abundant parallelism, high computational density, and flexibility of changing behavior during runtime make such architectures are superior to ASIC based approaches.

Table1.Performance of various Coarse-Grained Reconfigurable Architectures

CGRA	Granularity	Resource utilization	Memory Band Width	Architecture
ChESS	4 bit	low	Low	Hexagon mesh
MATRIX	8 bit	low	Low	2D Mesh
RaPiD	16 bit	high	Low	1D Array
Morphosys	16 bit	low	Low	2D Mesh
MORA	8 bit	high	high	2D Array

31st May 2010. Vol.15. No.2.
© 2005-2010 JATIT. All rights reserved
www.jatit.org

REFERENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra "Overview of the H.264/AVC video coding standard,"IEEE Trans. on Circuits and Systems for Video Technology, vol.13, no. 7, pp. 560–576, July 2003.

[2] Injong Rhee, ."Quadtree-Structured Variable-SizeBlock-Matching Motion Estimation with Minimal Error,"IEEE Trans. Circuits Syst. Video Technol., vol. 10(Feb.):pp.42-50, 2000.

[3] Lappalainen, V. Hailapuro, A. Hamalainen, T.D. Nokia Res.Center, Tampere, "Performance of H.26L video encoder on General -purpose processor," The Journal of VLSI Signal Processing.

[4] S. Reader and T. Meng, "Performance Evaluation of Motion Estimation Algorithms for Digital Signal Processors," Tech. report, Stanford University, 1999.

[5] P. M. Kuhn, "Fast MPEG-4 Motion Estimation: Processor Based and Flexible VLSI Implementations," Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology vol.23, pp 67-92, October 1999.

[6] J.F. Shen et al, " A Novel Low-Power Full-Search Block- Matching Motion-Estimation Design for H.263+," IEEE Transactions on Circuits and Systems for Video Technology, vol.11, no. 7, July2001, pp.890-897.

[7] L. de Vos and M. Schobinger, "VLSI architecture for a flexible block matching processor," IEEE Trans. Circuits and Systems for Video Technology, Vol. 5, pp. 417-428, 1995.

[8] S. Y. Yap and J. V. McCann, "A VLSI architecture for variable block size video motion estimation," IEEE Transactions on CAS II, vol. 51, no. 7, July 2004.

[9] Chien-Min Ou, Chian-Feng Le and Wen-Jyi Hwang, "An Efficient VLSI Architecture for H.264 Variable Block Size Motion Estimation," IEEE Transaction on Consumer Electronics, Volume 51, Issue 4, Nov. 2005 Page(s):1291 -1299.

[10] Alex Soohoo, "FPGA Co-Processing Architectures for Video Compression," Altera Corporation.

[11] E. Waingold et al., "Baring it all to Software: RAW Machines," IEEE Computer, September 1997, pp. 86-93.

[12] E. Mirsky, A. DeHon, "MATRIX: A ReconfigurableComputing Architecture with Configurable Instruction Distribution and Deployable Resources," Proc. IEEE FCCM'96, Napa, CA, USA, April 17-19, 1996.

[13] A. Marshall et al., "A Reconfigurable Arithmetic Array for Multimedia Applications," Proc. ACM/SIGDA FPGA'99, Monterey, Feb. 21-23, 1999

[14] Carl Ebeling, Darren C. Cronquist, Paul Franklin, Chris Fisher,"RaPiD - A Configurable Computing Architecture for Compute Intensive Applications," University of Washington Department of Computer Science & Engineering Tech Report TR-96-11-03.

[15] H.Singh,M.LeeG.Lu F. Kurdahi, N. Bagherzadeh, "MorphoSys:A Reconfigurable Architecture for Multimedia Applications,"sbcci, pp.134, XI Brazilian Symposium on Integrated Circuit Design, 1998

[16] Marco Lanuzza, Stefania Perri, Pasquale Corsonello, Martin Margala ," A New Reconfigurable Coarse-Grain Architecture For Multimedia Applications," Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), pp 119-126, 2007

[17] K.M. Yang, M. T. Sun, and L. Wu, "A family of VLSIdesigns for the motion compensation block-matching algorithm," IEEE Trans. Circuits Syst., vol. 36, no. 10, pp.1317–1325, Oct. 1989.

[18] Y.K. Lai and L. G. Chen, "A data-interlacing architecture with two dimensional data-reuse for full-search block matching algorithm ," IEEE Trans. Circuits Syst. Video Technovol. 8, no. 2, pp. 124–127, Apr. 1998.