Cogeneration of fast motion estimation processor and algorithms using loss less compression

VIJI.M[1],
M.E (VLSI Design), Srinivasan Engineering College[1].
Perambalur, Tamilnadu.
vijimuthaiyan225@gmail.com [1].

S. CHITRA[2] ,
Assistant Professor (ECE), Srinivasan Engineering College[2].
Perambalu[2],Tamilnadu.
chitra_12341@yahoo.com

Abstract -- Flexible and scalable motion estimation processor for the h.264 advanced video coding standard handling the processing requirements for high-definition video and suitable for FPGA implementation. The new motion estimation algorithm of kalman filter used to reduce the computational complexity. The eight custom instruction sets are used in the motion estimation. All processor instances are remaining binary compatible so recompilation process is not required. The motion estimation algorithm to the hardware architecture leading to a very efficient implementation.

Keyterms -- kalman filter, H.264, motion estimation, video coding.

I. Introduction

Motion estimation process is performed in the video. The videos in the form of continuous picture frame. The every picture frame is related to each other they have only the small difference between them. In the each frame is divided in to the sub macro blocks and the sum of absolute difference is to be calculated between the sub macro blocks using the motion vector.

The inter frame and intra frame prediction is to be done in the sub macro blocks. The motion estimation process is done at using the block matching algorithms in previous work. In proposed the new motion estimation algorithm of kalman filter in the architecture had nine different types of instruction set are used. The algorithm is run in the instruction set. The SAD operation is performed in parallel. The proposed work uses the six integer processing unit, two half pixel processing unit and one quarter pixel processing unit. Convert the sequence of each 8*8 sub partition of a macro block into parallel processing without sacrifices the
video quality. Motion vectors are medium predicted by the left, top and top right is replaced by the medium of the motion vectors of the top left, top right and top macro block for all kinds of sub block. 2D systolic array parallel adder tree to generate the SAD of larger block sizes. Share the overlapped search area of adjacent macro blocks and reduce the memory transfer from external RAM. The concept was briefly introduced in Yu- Wen Huang, Tu-ChihWang, Bing-Yu Hsieh, and Liang-Gee Chen [6]

Ching-Yeh Chen, Shao-Yi Chien, Yu-Wen Huang, Tung-Chien Chen, Tu-Chih Wang, and Liang-Gee Chen[3] Two hardware architecture to support traditional fixed block size motion estimation and variable block size motion estimation. Broadcasting reference pixel rows propagating partition SAD. The SAD tree is a 2D intra level architecture and consists of a 2D processing element array and one 2D adder tree with propagation register.

C-Y.kao and L. Youn-Long [8] the architecture had a total of 256 process element. The pipelined architecture delivered a throughput level. It completes a matching. Comparator will relate the main SADs and the best motion vectors and their output results. Data reuse the search ranges of 2 consecutive current macroblocks overlap with each other.

T. Dias, S. Momcilovic, N. Roma, and L. Sousa [5] Proposes an application-specific instruction set processor (ASIP) to implement data-adaptive motion estimation algorithms that is characterized by a specialized datapath and a minimum and optimized instruction set. ISA is based on register-register architecture and provides only a reduced number of different operations (eight) that focus on the most widely executed instructions in ME algorithms. The register file consists of 24 GPRs and eight special purpose registers capable of storing one 16 bits word each.

II. Liquid Motion Instruction Set Architecture (ISA)

The instruction set express the inherent parallelism available in the motion estimation algorithm in a simple way to minimize the overheads for instruction fetch and decode and to keep the execution units of the core as busy as possible. The number of execution units available in the proposed processor vary depending on the implementation, so it is important that binary compatibility between different hardware implementations is achieved, meaning a program only needs to be compiled once to be executable on any configuration.

The instruction set architecture consists of a total of nine different instructions. The first two arithmetic instructions for integer and fractional pattern searches, a

total of six control instructions that change the program flow and one mode instruction that sets the partition mode.

The arithmetic instructions express the parallelism with two fields that identify the number of points to be used by the pattern and the position in the point memory where the offsets for that pattern are defined. The control unit can then execute the instruction with a parallelism level that ranges from issuing each of these points to a different execution unit in a fully parallel hardware configuration, partition mode and reference frame of the core and configures the internal control logic to operate with different address boundaries and data sources.

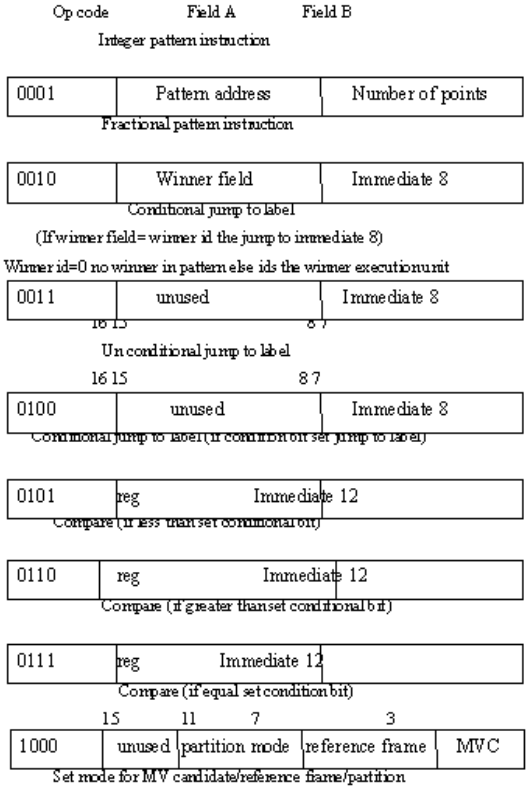

Fig 1. Liquid motion ISA

A. Programming Model

The processor offers a simple programming model so that a motion estimation algorithm programmer can access the functionality of the hardware without detailed knowledge of the microarchitecture. The algorithm designer use these constructs to create the arbitrary block matching algorithm ranging from classical full search to advanced algorithm.

B. Kalman filter

In the proposed work uses the kalman filter. It predict the motion vector of the current frame using the using the motion vector of the previous frame. The filter predicts the value and computing weighted average for the predicted value. The weights are calculated from the covariance, a measure of the estimated uncertainty of the prediction. This process is repeated. The method aim at producing values closer to the truth value of measurements and their associated values calculated by using the noisy observed values.

The algorithm had the two phases predict phase and the update phase. The algorithm needs the initial input matrix of the system to correctly predict the system next state. So we have to use the UMH algorithm to initialize it. The first frame is inter coded and the second and third frame is predicted using the UMH algorithm. The prediction value is stored in the P-frame. The algorithm take the initial value from the P-frame and it predict the values calculated by the kalman filter start from the fourth frame.

The basic equations of the kalman filter are,

$$x_k = A_k * x_{k-1} + B_k * u_k + w_k \quad (1)$$

$$z_k = H_k * x_k + v_k \quad (2)$$

The motion vectors for each macroblock in the frame are stored in a matrix. By using the motion vector we easily calculate the affine transform for each block. The affine transform is a 3*3 matrix. X_k represent the parameter to affine transform, as with each frame is going to predicted and updated.

Prediction for state vector and covariance:

$$x_{k|k-1} = A_k * x_{k-1|k-1} + B_k * u_k \quad (3)$$

$$P_{k|k-1} = A_k * P_{k-1|k-1} + A_k * Q_k \quad (4)$$

Compute kalman gain factor:

$$K_k = P_{k-1|k-1} * H_k^T * \text{inv}\,(H_k * P_{k-1|k-1} * H_k^T + R_k) \quad (5)$$

Correction based on observation or update phase:

$$x_{k|k} = x_{k-1|k-1} + K_k * (z_k - H_k * x_{k-1|k-1}) \quad (6)$$

$$P_{k|k} = P_{k|k-1} - K_k * H_k * P_{k|k-1} \quad (7)$$

z_k = calculated affine transform
u_k = input control is zero
A_k = state transition matrix
B_k = input matrix is zero

P_k = covariance of the state vector estimate
Q = process noise covariance is same factor of identify matrix.
R = measurement noise covariance is same factor of identify matrix.
H_k = observation matrix is identity matrix.

C. *Early Termination*

Early termination is a very important feature used to speedup execution in fast motion estimation algorithms. If the pattern fails to improve the SAD of the previous iteration, the algorithm terminates the current search loop. To implement this technique, each completing check pattern instruction sets a best_eu register indicating which search point has improved upon the current cost. This register is set to zero before each instruction starts, so the value of the best_eu register at the end of execution indicates if the instruction has improved the cost value (best_eu no longer zero) and if so which search point has achieved this improvement. The conditional jump instruction checks this register and changes the execution flow as required. The same hardware can be used to support a technique to avoid searching duplicate points by coding optimized subpatterns in memory.

III. System Architecture

A. *Fractional pixel execution unit*

Motion estimation process can be divided into two steps. The integer pixel motion estimation and fractional pixel motion estimation. The search range of fractional pel motion estimation in h.264 reference software is fixed to +3 for quarter pel accuracy case.

Full fractional search it require 49 points. Hierarchical fractional search reference software at ¼ pel accuracy needs to check 17 search points. The fractional pipeline is as fast as the integer pipeline, requiring the same number of cycles to compute each search position.

B. *Interpolation execution unit*

The engine supports both half and quarter-pel motion estimation. The interpolation hardware is cycled three times to first calculate the horizontal pixels, then the vertical pixels, and finally the diagonal pixels.
The IEU calculates these half-pels using a six-tap filter as defined in the H.264 standard. The IEU has a total of eight systolic1-D interpolation processors, each with six processing elements.
Which can broken into two steps,

In case of ½ pixel, 6 tap FIR filter is to be used along with the weight (1/32, -5/32, 5/8, 5/8, -5/32, 1/32). 1 to ½ pixels tap indicates the number of pixels. 6 tap means 6 pixel in vertical or horizontal direction.

¼ pixel interpolations is to be made using ½ pixel and original pixel that are calculated in step1 bilinear interpolation is used for ¼ pixel basically using ½ pixel.

C. *Integer execution unit*

Integer pel motion estimation takes most of the computational cost of the whole motion estimation. Each functional unit is pipelined at 64 bit word. All the access to reference and macro block memory is done through 64 bit wide data buses and the SAD engine also operates in 64 bit data in parallel. The memory is organized in 64 bit words and typically access is unaligned.

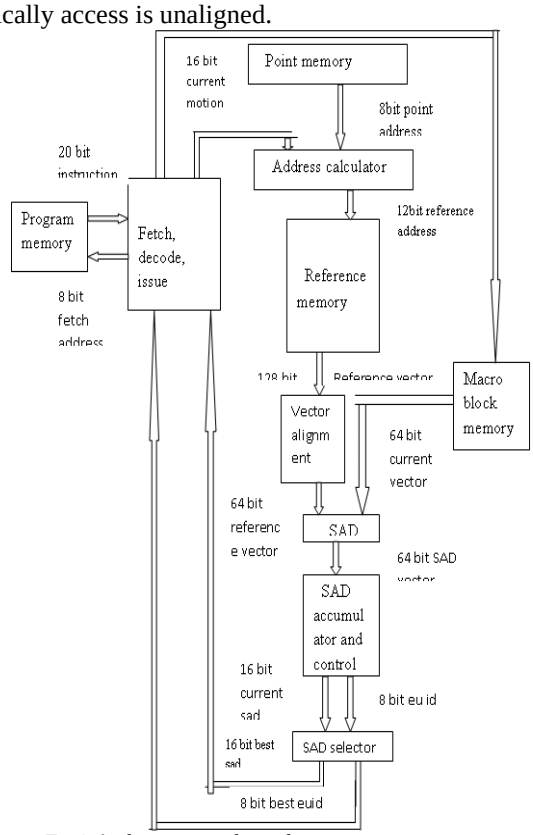

Fig 2.Architecture with single execution unit.

IV. Conclusion

The presented processors are the support of arbitrary fast motion estimation algorithms, integration of fractional and integer-pel support, the availability of a software toolset to ease the development of new motion estimation algorithms and processors. The configurable architecture with the more number of execution units. The architecture have 4 integer pel execution unit 2 fractional pel execution unit and 1 interpolation execution unit. The integer is always pipelined. The partitions are calculated sequentially but SAD cannot reuse. The large searches ranges linear increase in hardware resources and consequently increase in power consumption.

References

[1] D. Alfonso, F. Rovati, D. Pau, and L. Celetto, “An innovative, programmable architecture for ultralow power motion estimation in reduced memory MPEG-4 encoder,” in Dig. Tech. Papers Int. Conf. Consumer Electron., 2002, pp. 344–345.

[2] K. Babionitakis, G. Doumenis, G. Georgakarakos, G. Lentaris, K. Nakos, D. Reisis, I. Sifnaios, and N. Vlassopoulos, “A real-time motion estimation FPGA architecture,” J. Real-Time Image Process., vol. 3, no. 1–2, pp. 3–20, Mar. 2008.

[3] C.-Y. Chen, S.-Y.Chien, Y.-W.Huang, T.-C.Chen, T.-C. Wang, and L.-G. Chen, “Analysis and architecture design of variable block-size motion estimation for H.264/AVC,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 3, pp. 578–593, Mar. 2006.[Online]. Available: http://sharpeye.borelspace.com/.

[4] T. Dias, S. Momcilovic, N. Roma, and L. Sousa, “Adaptive motion estimation processor for autonomous video devices,” EURASIP J. Embedded Syst., vol. 2007, no. 1, p. 41-41, 2007.

[5] Y.-W. Huang, T.-C.Wang, B.-Y.Hsieh, and L.-G. Chen, “Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264,” in Proc. ISCAS, May 2003, vol. 2, pp. 796–799.

[6] Y.-W. Huang, C.-Y.Chen, C.-H.Tsai, C.-F.Shen, and L.-G. Chen, “Survey on block matching motion estimation algorithms and architectures with new results,” J. VLSI Signal Process. Syst., vol. 42, no. 3, pp. 297–320, 2006.

[7] C. Kalaycioglu, O. Ulusel, and I. Hamzaoglu, “Low power techniques for motion estimation hardware,” in Proc. Int. Conf. Field Programmable Logic Applic., Sep. 2009, pp. 180–185.

[8] C.-Y. Kao and L. Youn-Long, “AnAMBA-compliant motion estimator for H.264 advanced video coding,” in Proc. IEEE Int. SoC Design Conf., Seoul, Korea, Oct. 2004, pp. 200–206.