Conservative Approximation–Based Full-Search Block Matching Algorithm Architecture for QCIF Digital Video Employing Systolic Array Architecture

Ganapathi Hegde, Amritha Krishna R.S., and Pukhraj Vaya

This paper presents a power-efficient hardware realization for a motion estimation technique that is based on the full-search block matching algorithm (FSBMA). The considered input is the quarter common intermediate format of digital video. The mean of absolute difference (MAD) is the distortion criteria employed for the block matching process. The conventional architecture considered for the hardware realization of FSBMA is that of the shift register–based 2-D systolic array. For this architecture, a conservative approximation technique is adapted to eliminate unnecessary MAD computations involved in the block matching process. Upon introducing the technique to the conventional architecture, the power and complexity of its implantation is reduced, while the accuracy of the motion vector extracted from the block matching process is preserved. The proposed architecture is verified for its functional specifications. A performance evaluation of the proposed architecture is carried out using parameters such as power, area, operating frequency, and efficiency.

Keywords: Motion estimation, FSBMA, systolic array architecture, conservative approximation, low power.

Manuscript received Apr. 26, 2014; revised Mar. 27, 2015; accepted Apr. 13, 2015.

Ganapathi Hegde (corresponding author, ganapathihegde1@gmail.com), Amritha Krishna R.S. (amrithakrishnars@gmail.com), Pukhraj Vaya (prvaya@gmail.com) are with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bangalore, India.

I. Introduction

Video data needs to be compressed before storage and transmission. Several compression standards are currently in use. Compression of video sequences requires efficient very large scale integrated circuit (VLSI) architectures. Complex algorithms are required to estimate the redundancy possessed by video data [1]–[2]. Systolic array architectures that are of the cascaded type are employed for block matching algorithms (BMAs) to obtain higher throughput [3]–[5]. A programmable systolic array processor is used in a data-adaptable, modular architecture [6]–[8]. High-speed power-efficient algorithms and respective architectures for full-search block matching algorithms (FSBMAs) are developed to avoid unwanted mean of absolute difference (MAD) computations [9]–[11]. FSBMAs perform a block matching process to obtain a vector that contains the highest motion information, which is nothing but a motion vector (MV). In [12]–[16], algorithmic-level modifications to an FSBMA are proposed to reduce its computation complexity by exploiting global elimination, early termination, and partial distortion techniques.

Hybrid video compression techniques combining 3-D discrete wavelet transforms (3-D DWTs) and motion estimation (ME) to reduce the complexity of FSBMAs are introduced in the design part of the paper in [17].

In this paper, we have extended our initial work by employing a conservative approximation (CA) technique on a conventional FSBMA systolic architecture for the quarter common intermediate format (QCIF). This approach is faster and offers a reduced power consumption in comparison to a

conventional FSBMA architecture that does not employ CA.

The rest of this paper is organized as follows. In Section II, a brief overview of an FSBMA and the conventional method of its implementation into a systolic architecture for QCIF video input is presented. Optimization of the FSBMA architecture by employing a CA technique and the major building blocks used for its construction are presented in Section III. Our experimental results and performance evaluation are presented in Section IV. The paper is concluded in Section V.

II. FSBMA and Systolic Implementation

Video data consist of successive frames that offer correlation either in still or moving modes. An ME algorithm can be used to examine such correlations temporally and to extract any subsequent motion information. A motion compensation technique is then able to use this information to reconstruct a frame. In a real-time video scene, motion can be a complex combination of translation and rotation. Such motion is difficult to estimate and may require a large amount of processing.

In a BMA, an ME technique, it is assumed that all pixels within a given block have the same motion activity. A BMA is then able to estimate a motion on the basis of such rectangular blocks. It produces one MV for each block of pixels [6]–[8].

In a block matching process, each block in a current video frame is matched against candidate blocks in a search area on a reference frame. The best-matched candidate block is found and its displacement (MV) is recorded.

In a typical inter-frame coder, an input frame is subtracted from a predicted reference frame, and any resulting residue is then extracted [9]. Consequently, these residues along with an MV are transmitted instead of using the original frame. Hence, inter-frame redundancy is removed and data compression is achieved. At the receiver end, a decoder builds a residue signal from the received data and adds it to the reconstructed reference frames. The summation of the residue (difference between the actual input frame and predicted frame) and MV then gives an exact replica of the transmitted current frame.

The process of obtaining an optimal MV by matching a candidate block of a current frame with all the available blocks of a reference frame within a given search area is performed by an FSBMA. FSBMAs used with ME have a simple control flow and present different levels of potential parallelism; consequently, they are suitable for hardware implementation. An FSBMA is able to obtain the MV by identifying the least possible MAD values. However, the number of necessary computations is very high due to the large number of candidate blocks required to evaluate MAD. Systolic implementations exploit massive parallel computations at the frame level starting from the block level. Hence, they are suitable array architectures for FSBMA hardware implementation.

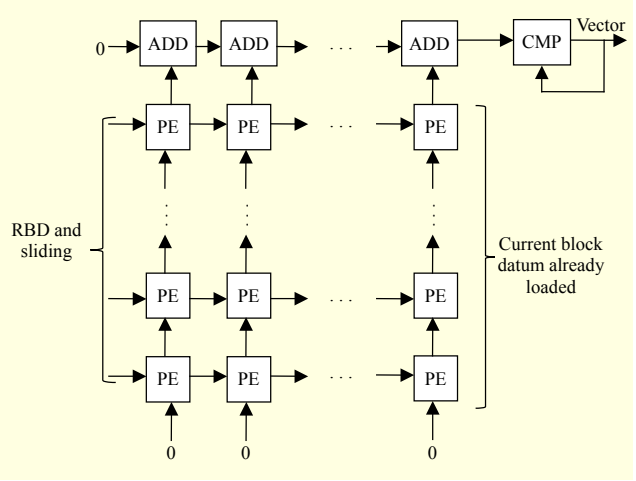

Fig. 1. Block matching process in FSBMA.

In a systolic FSBMA architecture, data transfers are prearranged and are displayed in the form of activity snapshots. Systolic array architectures represent important characteristics such as modularity, regularity, local interconnection, a high degree of pipelining, high synchrony, and global control. Extra delays can be inserted into a processing element (PE) array of a systolic architecture to ensure correct synchrony in a data flow. A systolic implementation of a block matching process is shown in Fig. 1. In an FSBMA implementation, block data is loaded into each PE node input. Reference block data (RBD) is then slid in from the left. Computation then begins from the bottom-left corner of the processor array. The total MAD for an MV is obtained by an adder (ADD) and propagated to a comparator (CMP) in the subsequent clock cycles. It is then compared with the stored MAD resulting from the previous comparison, and then the smaller of the two is kept in the CMP for subsequent comparison [6]. The aforementioned computation procedure is repeated until all possible candidate blocks are compared and a final MV is recorded.

A 2-D FSBMA architecture consists of three major units; namely a systolic array unit (SAU), an input/control unit (IOCU), and a block matching unit (BMU). A 2-D systolic architecture for a conventional FSBMA is shown in Fig. 2. The SAU shown in the figure contains both a PE array and a shift register array. The depicted architecture, for QCIF video input, includes sixty four (8 × 8) PEs and seven shift registers — each of which is 15 bits in length. A 2-D systolic array calculates the absolute difference (AD) in parallel and then sends the sum of the ADs to the parallel adder (PA), which in turn, calculates the MAD value for each block. The depicted BMU contains a PA, a candidate region monitor (CRM), and a best-match selection unit (BMSU) [16]. The IOCU comprises RBD and search area data (SAD). These data enter the first stage of the SAU serially

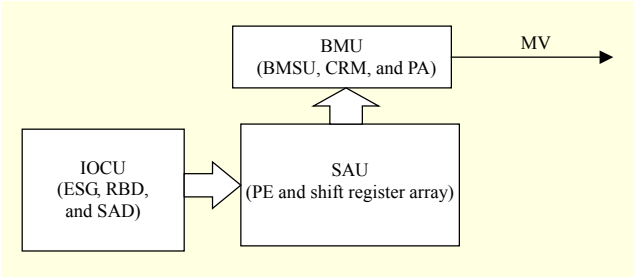

Fig. 2. 2-D systolic architecture for conventional FSBMA [17].

under the control of an enable signal generator (ESG) and are shifted in it. The PA takes inputs from the first row of PEs and sends the output to the BMSU (see Fig. 2).

MAD values are fed serially into the BMSU to complete the computation. The BMSU takes a single input from both the CRM and the PA. If the incoming sum values from the PA are smaller than the already existing value in the BMSU, then the BMSU replaces the current value in its register and the corresponding match position is recorded for the extraction of an MV. The output of a comparator is disabled by the CRM if the incoming data are invalid. A PE is used to store a candidate-block pixel value from RBD and shift an SAD value to calculate a partial sum. This partial sum is then passed to the up-neighbor PE. The search pixel data are shifted from one PE to the next (from left to right) [16]–[17].

III. Optimization of FSBMA Using CA Technique

Optimization of an FSBMA is achieved using the CA technique, which reduces computation of MAD. A flowchart of an FSBMA employing this technique is shown in Fig. 3. If the estimated value of MAD ($\overline{MAD}$) is greater than the minimum of all of the MAD values searched thus far, ($MAD_{min(sfs)}$), then it gets discarded; else, the existing MAD values will be updated (see Fig. 3).

This approach, when employed in conventional systolic FSBMA architectures, reduces *switching* activity. An ME unit computes the next distortion value (that is, the next MAD value) using data available for computing the current block matching process. This helps in asserting the disable switch of the SAU if the calculated distortion is greater than the current minimum MAD searched thus far. If the disable signal is asserted, then new data are not required for computing the next distortion. Hence, this lessens the computation requirement.

A block schematic of the CA-based 2-D systolic FSBMA architecture for QCIF and the corresponding SAU module that is used to build it, are shown in Figs. 4 and 5, respectively. This architecture includes an extra disable connection through a distortion approximation unit (DAU) to the SAU. The DAU becomes active and sends a signal to the SAU as it is about to

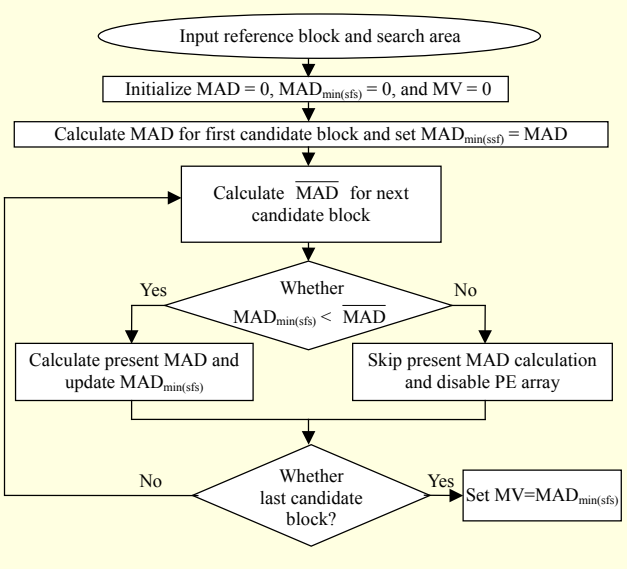

Fig. 3. Flowchart of CA-based FSBMA [10].

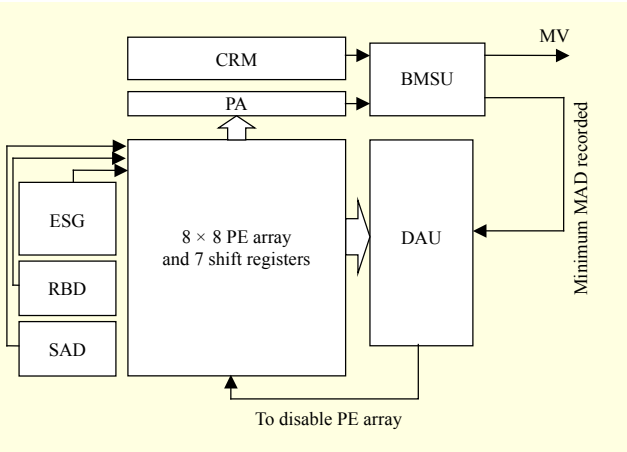

Fig. 4. CA-based systolic FSBMA architecture for QCIF [10].

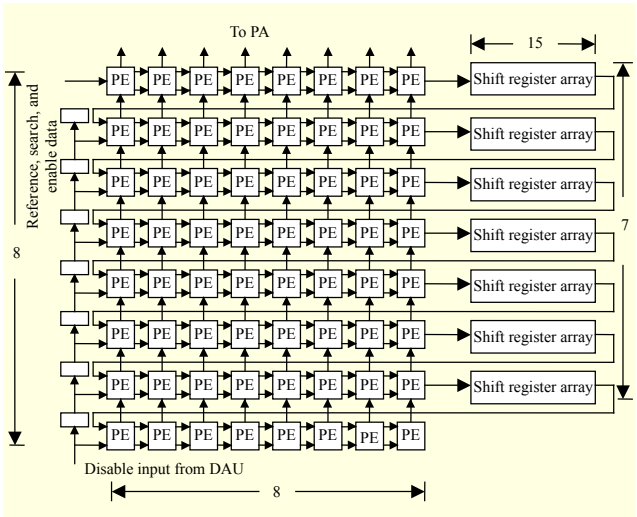

Fig. 5. SAU module used to build CA-based FSBMA.

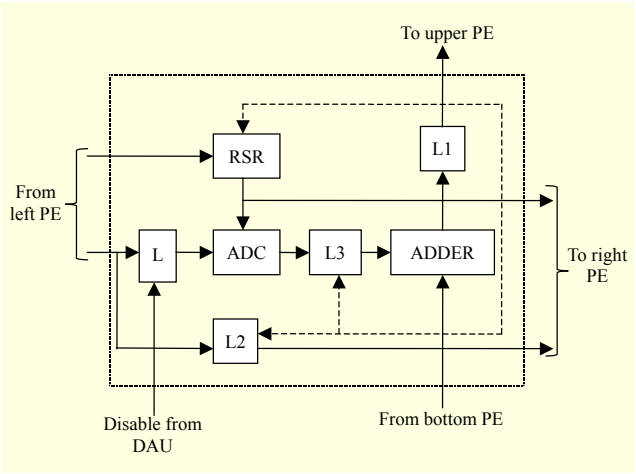

Fig. 6. PE module used to build SAU [10].

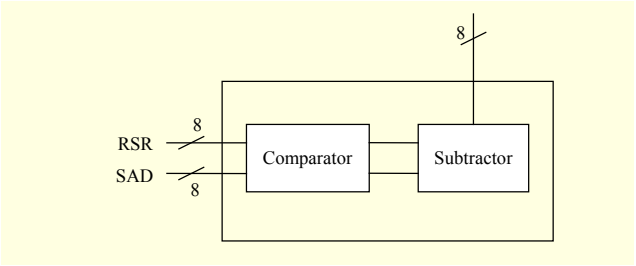

Fig. 7. ADC module schematic.

compute the redundant MAD values, which in turn, disables the SAU (see Fig. 4). When the SAU is disabled, then the pre-existing MAD values are retained and further entry of any new values is blocked. A BMSU generates an MAD value for each candidate block and then compares it with the current minimum MAD value. The BMSU outputs are then updated in accordance with the result of this comparison.

In Fig. 6, we can see that each PE consists of a reference shift register (RSR), an absolute difference calculator (ADC), an adder (ADDER), and three shift registers (L1, L2, and L3). Shift register L3 divides ADC and ADDER into two-stage pipelining. For the systolic part of the architecture, the RSR, L2, and L3 shift registers are used, all of which are 8-bit parallel in/parallel out registers. Shift register L1 is a 15-bit shift register. To block the computation of redundant distortion values, a latch (L) is included in each PE. It skips the MAD computation when the disable signal is received by the DAU module.

An ADC module consists of a comparator, which compares RBD taken from RSR and SAD. The lesser value among them is subtracted from the greater value using a subtractor. Hence, the AD value is obtained at the output of the subtractor. An ADC module block schematic is shown in Fig. 7.

A PA module takes input from the first row of PEs and sends the sum of these inputs to the BMSU module. It has eight

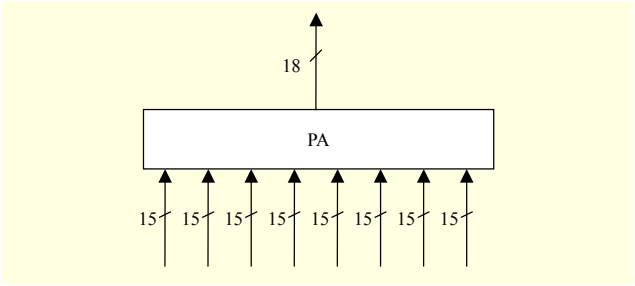

Fig. 8. PA module schematic.

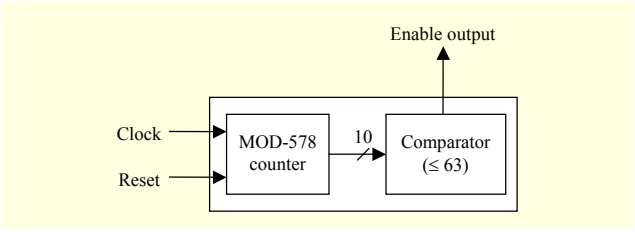

Fig. 9. ESG module schematic.

15-bit inputs coming from PEs and one 18-bit output that is sent to the BMSU. It takes one clock pulse to output a result. A block schematic of a PA module is shown in Fig. 8.

An ESG module contains a MOD-578 counter and a comparator unit. The output of the MOD-578 counter is fed to the comparator, which then compares this input against its own value of 63. If the input is in the range 0 to 63, then the ESG module is enabled. An RSR input is accepted for the first 64 pulses and then stored. Next, it is disabled for the remaining 514 pulses when an ADC and addition operations are performed. Computed MAD values are sent to the BMSU. A block schematic of an ESG module is shown in Fig. 9. Figure 10 shows a block schematic of an address generator for an RBD. It contains a MOD-8 counter, comparator (7), comparator (577), adder1 to add 169, adder2 to add 8, and adder3. It generates an address in such a way that it systematically takes the input values from memory and feeds the PE array. If the counter output is 7, then adder1 adds 169. Similarly, if the counter output is 577, then 8 is added to adder2. Adder3 adds the output of adder1 and adder2 to output the required address for the RBD.

An address generator for SAD contains a MOD-24 counter, comparator (23), comparator (577), adder1 to add 153, adder2 to add 8, and adder3. It generates an address in such a way that it systematically takes the input values from memory and feeds the PE array. If the counter output is 23, then adder1 adds 153. Similarly, if the counter output is 577, then 8 will be added to adder2. The adder3 adds the output of adder1 and adder2 to output the corresponding address of SAD. A block schematic of an address generator for SAD is shown in Fig. 11.

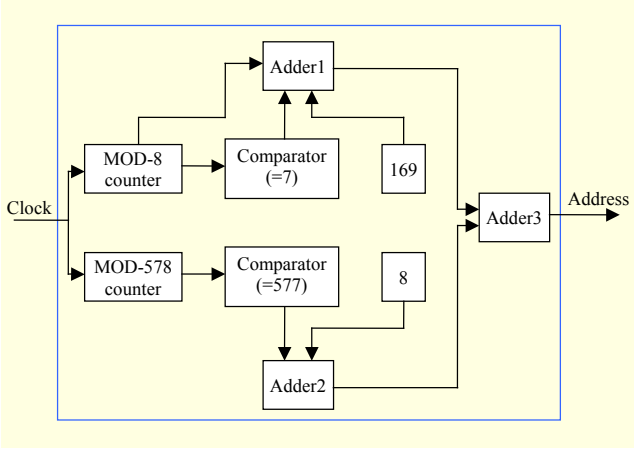

Fig. 10. Address generator for RBD.

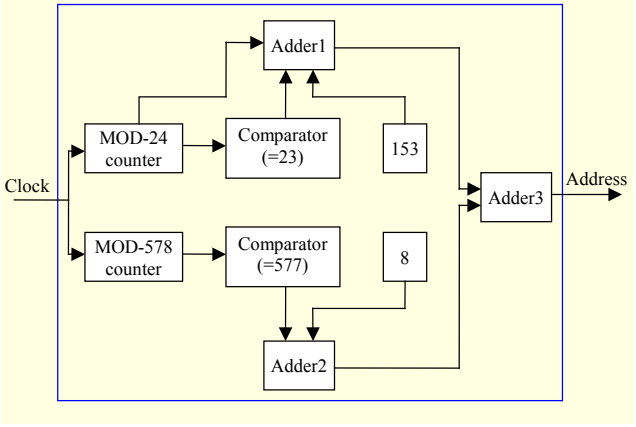

Fig. 11. Address generator module for SAD.

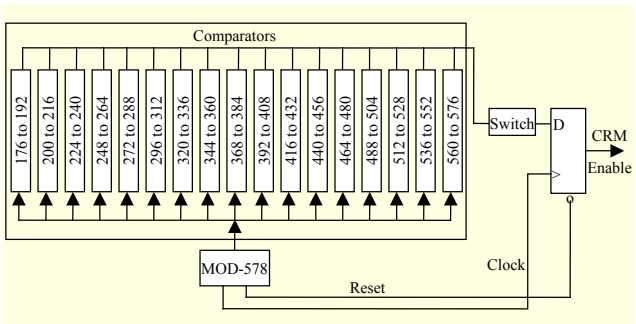

Fig. 12. CRM module schematic.

A block schematic of a CRM module is shown in Fig. 12. The CRM module produces an "enable" signal if a valid MAD value comes out of the PA module. If any one of the conditions among its seventeen available ranges is satisfied, then it is enabled; else, it will be disabled. The output signal from the CRM is given as an input to the BMSU. The D flip-flop in the design is to make one clock pulse delay. When the SAD is given as input to the PE and shift register array, the PA

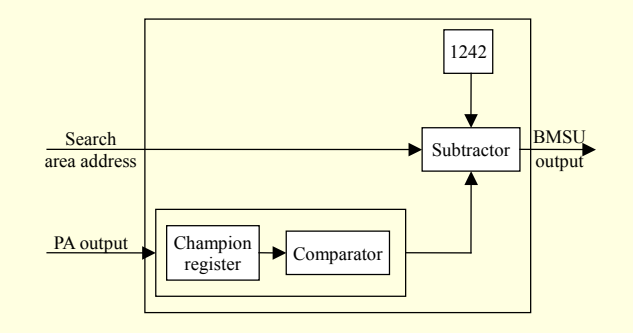

Fig. 13. BMSU module schematic.

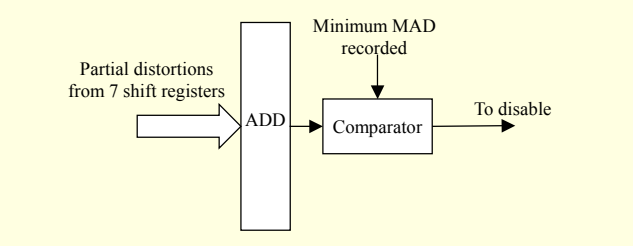

Fig. 14. DAU module schematic [10].

keeps producing MAD values within the valid search area region.

A BMSU takes inputs from modules CRM, SAD, and PA. It consists of different submodules, such as a champion register, a comparator, and a subtractor. The champion register is used to store the incoming MAD values that are output from the PA. The comparator compares an incoming MAD value against the current (latest) MAD value held in the register. If the incoming MAD value is less than this current MAD value, then an "enable" signal is sent to the subtractor unit. The incoming MAD value will then replace the current MAD value in the register and become the newly appointed current MAD value; else, the original current MAD value is retained in the register. The subtractor unit subtracts 1242 from the address generated by SAD; this then becomes the output of the BMSU. A block schematic of a BMSU module is shown in Fig. 13.

A DAU module is an important constituent unit of an FSBMA architecture, for it can reduce power consumption. A block schematic of a DAU module is shown in Fig. 14. Eight conservatively estimated MAD values are calculated from the output of an SAU using a combinational circuit unit. They are then added using a parallel adder (ADD) and compared with the minimum recorded MAD value. If the output of the adder is greater than the minimum recorded MAD value, then computation of MAD for future candidate blocks will not help to predict an MV (Fig. 14). Therefore, the DAU disables the SAU from further regular computation of the next block. Hence, redundant MAD computations are skipped, which helps with power reduction.

IV. Results

The complete architecture of the CA-based systolic FSBMA for QCIF (Fig. 4) is described using Verilog HDL and simulated using Modelsim SE 6.5 for functional verification. It is then synthesized using Xilinx ISE 10.1. It is mapped to a 90 nm technology standard cell library using a synopsis design compiler. Power, area, and frequency of operation of the architecture are also analyzed. Specifications considered for both the conventional FSBMA architecture and the CA-based systolic FSBMA for QCIF are presented in Table 1.

Various time parameters recorded from simulation results of different submodules of the FSBMA architectures are presented in Table 2. The total number of processing cycles involved in the computation of one frame of QCIF is observed to be 583; the values that make up this total are validated by a mathematical formula derived in [8].

After applying the CA technique to the conventional FSBMA architecture, many redundant MAD computations are removed and the corresponding clock cycles are skipped.

Table 3 presents the average number of clock cycles skipped for six randomly selected frames for both the CA-based FSBMA architecture and the conventional FSBMA architecture. From this information, we can deduce the differences in the number of cycles skipped between the conventional FSBMA and the CA-based FSBMA. The six frames were randomly selected from an input video sequence.

A comparison of the power, area, and frequency of operation between the CA-based FSBMA and the conventional FSBMA is presented in Table 4. It can be seen from the experimental results that there is a small increase in the total area of the CA-based architecture. This is due to the addition of an extra DAU module to the conventional FSBMA architecture. For the duration of the skipped clock cycles, the PE array will be disabled by the DAU; hence, power dissipation is reduced. Moreover, the frequency of operation increases.

The dynamic power in the CA-based FSBMA architecture depends inversely on the temporal correlation between the consecutive frames of the input video. The dynamic power is reduced when the correlation between them is high. The efficiency of the CA-based FSBMA architecture is increased by approximately 5% (Table 4). However, the incremental area remains approximately the same.

A summary of all the simulation results is represented as a timing diagram to get an overall view of the differences between the conventional and CA-based FSBMA architectures.

Table 1. Considered design specifications for FSBMA architectures.

Sl. No.	Parameter	Quantity
1	Video input (QCIF)	176 × 144 pixels
2	Block size	8 × 8 pixels
3	Displacement	8 pixels
4	Search area value	8 bits
5	Address generator (search area, reference block, and MV)	15 bits
6	Clock and reset	1 bit each
7	Total PEs	64
8	Number of shift register array	7

Table 2. Time parameters of different submodules of FSBMA architecture.

Sl. No.	Parameter	Quantity
1	Maximum delay time of adder (ns)	19.55
2	Maximum delay time of latch (ns)	6.03
3	Total permissible cycle time (ns)	25.59
4	Processing time for one frame (ms)	5.88
5	Processing cycles required for valid MAD	289
6	Total processing cycles required for one frame	583

Table 3. Clock cycles skipped for MAD computations of CA-based FSBMA architecture.

Frame number	Total number of clock cycles utilized for MAD calculation		Number of cycles skipped
	Conventional FSBMA [16]	CA-based FSBMA	
1	583	500	83
2	583	563	20
3	583	570	13
4	583	560	23
5	583	538	45
6	583	571	12
Average number of clock cycles skipped for random six frames			32.7

Table 4. Comparison of conventional and CA-based FSBMA architectures.

Sl. No.	Parameter	Conventional FSBMA [16]	CA-based FSBMA
1	Total dynamic power (μW)	655.9	489.7
2	Total area (square μm)	210.7	250.6
3	Frequency of operation (MHz)	50.6	58.7
4	Efficiency (%)	49.7	52.5

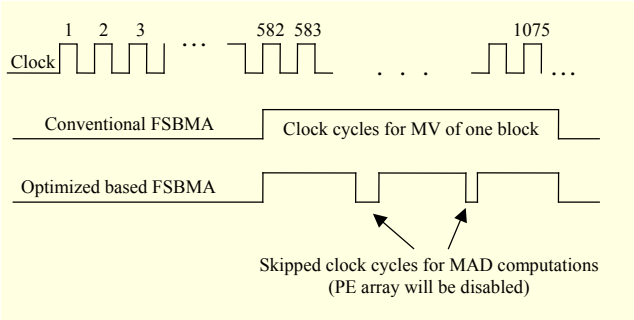

Fig. 15. Timing diagram of conventional and CA-based FSBMA architectures.

Fig. 16. Layout of CA-based FSBMA architecture.

It is shown in Fig. 15. The layout of the CA-based FSBMA architecture is shown in Fig. 16. It is extracted using synopsis back end IC and DC compiler.

V. Conclusion

In this paper, we have presented a power-efficient systolic-based FSBMA architecture for QCIF digital video input. It removes redundant MAD computations without compromising the accuracy of the MV. The experimental results show that it reduces the total dynamic power by 25% but at the expense of a 16% increase in total area when compared with the conventional FSBMA architecture featured in this paper. The efficiency of the proposed architecture is increased by 5% at an operating frequency of 58 MHz.

Acknowledgment

Authors wish to thank Dr. Cyril Prasanna Raj P. and Professor M.S. Ramaiah, School of Advanced Studies, Bangalore, India for helpful discussions and valuable comments.

References

[1] H.G. Musmann, P. Pirsch, and H.J. Grallert, "Advances in Picture Coding," *Proc. IEEE*, vol. 73, no. 4, Apr. 1985, pp. 523–548.

[2] S.Y. Kung, "VLSI Array Processors," *IEEE ASSP Mag.*, vol. 2, no. 3, July 1985, pp. 4–22.

[3] L.D. Vos and M. Stegherr, "Parameterizable VLSI Architectures for Full-Search Block-Matching Algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, Oct. 1989, pp. 1309–1316.

[4] K.M. Yang, M.T. Sun, and L. Wu, "A Family of VLSI Designs for the Motion Compensations Block-Matching Algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, Oct. 1989, pp. 1317–1325.

[5] T. Komarek and P. Pirsch, "Array Architectures for Block Matching Algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, no. 10, Oct. 1989, pp. 1301–1308.

[6] C.H. Hsieh and T.P. Lin, "VLSI Architecture for Block-Matching Motion Estimation Algorithm," *IEEE Trans. Circuits Syst.*, vol. 2, no. 2, June 1992, pp. 169–175.

[7] P. Pirsch, N. Demassiuex, and W. Gehrke, "VLSI Architecture for Video Compression – a Survey," *Proc. IEEE*, vol. 83, no. 2, Feb. 1995, pp. 220–246.

[8] S.C. Cheng and H.M. Hang, "A Comparison of Block-Matching Algorithms Mapped to Systolic-Array Implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, Oct. 1997, pp. 741–757.

[9] Y.C. Lin and S.C. Tai, "Fast Full-Search Block-Matching Algorithm for Motion-Compensated Video Compression," *IEEE Trans. Commun.*, vol. 45, no. 5, May 1997, pp. 527–531.

[10] V.L. Do and K.Y. Yun, "A Low-Power VLSI Architecture for Full-Search Block-Matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, Aug. 1998, pp. 393–398.

[11] Y.K. Chen and S.Y. Kung, "A Systolic Design Methodology with Application to Full-Search Block-Matching Algorithm," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 19, no. 1, May 1998, pp. 51–77.

[12] X.Q. Gua, C.J. Duamnu, and C.R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation," *IEEE Trans. Image Process.*, vol. 9, no. 3, Mar. 2000, pp. 501–504.

[13] M. Brunig and W. Niehsen, "Fast Full-Search Block Matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, Feb. 2001, pp. 241–247.

[14] Y.W. Huang et al., "Global Elimination Algorithm and Architecture Design for Fast Block Matching Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, June 2004, pp. 898–907.

[15] M. Jilang et al., "Low Power Systolic Array Processor Architecture for FSBM Video Motion Estimation," *Electron.*

Lett., vol. 42, no. 20, Sept. 2006, pp. 1146–1148.
[16] G. Hegde, C.P. Raj, and P.R. Vaya, "Implementation of Systolic Array Architecture for Full Search Block Matching Algorithm on FPGA," *European J. Sci. Res*., vol. 33, no. 4, July 2009, pp. 606–616.
[17] G. Hegde and P.R. Vaya, "Systolic Array Based Motion Estimation Architecture of 3D DWT Sub-band Component for Video Processing," *Int. J. Signal Imag. Syst. Eng*., vol. 5, no. 3, Nov. 2012, pp. 158–166.

Ganapathi Hegde received his BE degree in electronics and communication from the University of Mysore, India, in 1998 and his MTech degree in digital electronics from Visweswaya Technological University, Belgaum, India, in 2001. From 2001 to 2005, he worked as a lecturer at the Manipal Institute of Technology, India. Since 2005, he has been working as an assistant professor with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bangalore, India. His main research interests are VLSI architectures for video processing, asic design, and low-power VLSI design.

Amritha Krishna R.S. received her BTech degree in electronics and communication from Kerala University, India, in 2009 and her MTech degree in VLSI design from the School of Engineering, Amrita Vishwa Vidyapeetham, Bangalore, India, in 2012. Since 2012, she has been working as a senior systems engineer with Infosys Technologies, Bangalore, India. Her main research interests are digital electronics, VLSI design, and control systems.

Pukhraj Vaya received his MS degree in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1969 and his PhD degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1984. Since 1969, he has been involved in teaching and research. Now, he is working as a professor with the Department of Electronics and Communication Engineering, School of Engineering, Amrita Vishwa Vidyapeetham, Bangalore, India. His main research interests are semiconductor materials, photonic devices, VLSI architectures for video processing, and low-power electronics.