International Journal of Computing and Network Technology
ISSN (2210-1519)
Int. J. Com. Net. Tech. 4, No. 1 (Jan.-2016)

A Cost – Effective Programmable SoC for H.265/HEVC Full Search Motion Estimation using Xilinx ZYNQ-7 ZC706 FPGA

Yasser Ismail[1,2]

[1]*College of Information Technology, Department of Computer Engineering, University of Bahrain, Sakhair, Bahrain.*
[2]*Electronics and Communications Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt.*

Received: 02 Sept. 2015, Revised: 04 Dec. 2015, Accepted: 10 Dec. 2015, Published: 1 (January) 2016

Abstract: A complete Full Search Motion Estimation Video system that can be adopted and integrated into H.264/AVC and H.265/HEVC standards. The proposed system reduces the computational complexity as well as hardware complexity. The overall data needed by this system is greatly reduced by using smart and efficient local memory that uses data reuse principle. All components of the proposed system are optimized, and so, the speed of the proposed Motion Estimation system is greatly improved. Both of the current block and the corresponding search area are loaded efficiently inside the Processing Element (PE). The search area is loaded horizontally from a local memory while the current block is loaded once from an external memory. The local memory is implemented using registers and the addressing issues are done using a simple counter. This guarantees a fast processing, regularity of the data flow, simplicity of the hardware design, and 100% utilization factor of all components of the proposed system. Additionally, there are no complicated addressing modes to read or write data to/from the local memory. The proposed architecture is implemented using Xilinx ZYNQ-7 ZC706 FPGA tool. For a search range of 32×32 and block size of 16x16, the proposed Motion Estimation system can perform motion estimation of HDTV video at 123.53MHz operating frequency and achieving two levels of data reuse.

Keywords: Motion Estimation, FPGA, HDTV, H.264/AVC, H.265/HEVC.

1. INTRODUCTION

H.264/AVC (Advanced Video Coding) and H.265/HEVC (High Efficiency Video Coding) standards are recently used for many real-time applications [1-4]. Video conferencing, HDTV broadcasting, video-on-demand, and ultra frequency video transmission are examples for such real-time applications [5]. Such applications require very low bit-rate as well as high video quality. Previous two video standards achieve the last two requirements for real-time video applications by adding some complex tools to the video encoder. Multiple reference frames, half-pel and quarter-pel accurate Motion Estimation, parallel processing, and variable block sizes techniques are examples for such added tools.

Full Search Motion Estimation (FSME) is the well known algorithm used in both H.264/AVC and H.265/HEVC standards for removing the temporal redundancy of the transmitted video signal while keeping high video quality as well as low transmission bit-rate. However, it consumes most of the video encoding time [5]. As a result, many fast Motion Estimation algorithms were developed to tackle this problem. Three Step Search (TSS) [6, 7], New Three Step Search (NTSS) [8], Four Step Search (FSS) [9], Diamond Search (DS) [10], Cross Diamond Search (CDS) [11], Successive Elimination Algorithm (SEA) [12, 13], and Adaptive Search Window Size (ASWS) [14, 15] are examples of such fast Motion Estimation algorithms.

Although most of previous fast Motion Estimation algorithms provide great reductions in time encoding complexity, some of such algorithms are not implemented in VLSI and decrease the video visual quality [16, 17]. This is why Full Search Motion Estimation is the most used algorithm in most video coding standards due to its data flow regularity that allow an easy implementation in VLSI.

E-mail address: yasserali1977@gmail.com

Motion Estimation process starts by dividing the current frame into equal-sized blocks, each of size N×N pixels. The best match candidate block is calculated for each current block by searching a search area centered at the same position of the current block in the reference frame. Figure 1 shows the Motion Estimation process using a search area of size $2P_{max}\times 2P_{max}$.; where $2P_{max}$ is the range of a selected search area. The point located at the smallest cost is selected as the best match candidate block. The cost can be measured using the Sum of Absolute Difference (SAD) metric. The displacement between the center of the search area and the best match reference block is represented by the Actual Motion Vector (AMV) [5].

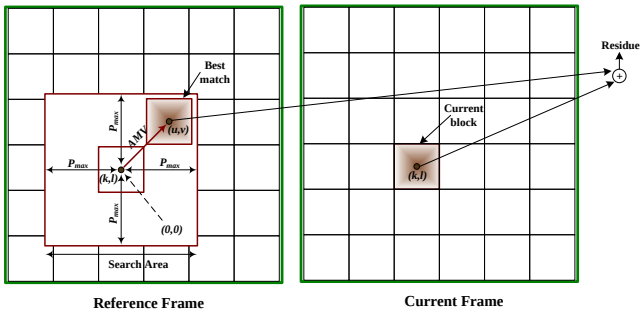

Figure 1: Motion Estimation process [5].

In this paper, a Full Search Motion Estimation architecture design is implemented on Xilinx ZYNQ-7 ZC706 FPGA. Regularity of data flow and reducing the I/O bandwidth required for video transmission are achieved in the proposed design. The implemented design is compared with the state of the art FSME technique in [18].

The paper is organized as follows. Section 2 presents the programmable system on chip. The proposed Motion Estimation system is discussed in details in section 3. Section 4 discusses the simulation results. Finally conclusion is drawn in section 5.

2. RELATED WORK

Recently, many researchers propose algorithms to speed up the Motion Estimation (ME) process. Some of such algorithms are not implementable due to their data flow irregularity. Other algorithms are implementable; however, they degrade the video quality. Therefore, although Full Search Block Matching (FSBM) algorithm is consider as the most exhausted Motion Estimation algorithm, most recent researchers implement FSBM for two main reasons. First, FSBM algorithm has a regular data flow that positively affects on area, power consumption, and the ME speed. Second, the high video quality results from FSBM algorithm can qualify such architectures to be used in high efficient state of the art video coding standard such H.264/AVC and H.265/HEVC [1, 2][4]. In [19], an adaptive search window size algorithm is used for each step of the Three Step Search (TSS) algorithm. The size of the search area is calculated using a parameter which is a function of the Sum of Absolute Difference (SAD) values of previous search steps. The calculated parameter is then allocated within certain threshold values to decide if the search window size will be incremented or decremented.. Although this algorithm can be implemented in hardware, the main disadvantage is the irregular data flow and the degradation in video quality [20]. Additionally, such algorithm increases the computations required to get the optimum Motion Vector (MV) since it increases the number of steps required for getting the final MV. In [21], the search window size of a 2D log-search algorithm is updated according to the SAD value calculated at the search center position. The SAD value is used to calculate two different thresholds that can categorize the motion of a current block into slow, medium or fast motion block. According to the selected category, a suitable search window size can be estimated. Another methodology of reducing the computations of ME process is to stop the search if further search points in search area are expected not to be selected as a best match. In [22], the Mean Absolute Difference (MAD) of previously estimated blocks is used as a threshold to stop the search for the current block. The used search pattern for such algorithm is the spiral search. If the current MAD exceeds a certain threshold, the threshold should stop. The implementation of such algorithm is hard due to the irregular search pattern (spiral search). Another approach for speeding up the ME process is to adapt the search window size using the prediction technique [23]. Getting the benefit of spatial and temporal homogeneity property of a video sequence, the motion activity of the current block can be estimated from the motion activities of previously surrounded blocks. The motion vectors of previously encoded blocks are used as indicators for the motion activities of the surrounded blocks.

Some of the above mentioned algorithms are not optimum for the hardware implementation since they have irregular data flow. Additionally, extensive book-keeping may be required to record the motion activities of previous encoded blocks. Consequently, more additional hardware may be required. More area and power consumption may result from such additional hardware. Finally, degradation of video quality is the main bottleneck of using such adaptive techniques that may result from accumulating error while adapting the search window size.

From previous discussions, we conclude that FSBM algorithm is the best implementable algorithm in hardware due to the regularity of its data flow and high resolution video quality. Full Search Block Matching The regularity of FS is maintained while reducing computations to about 60% as compared to full search. However, the drawback of the FSBM approach is the huge data required from the memory to perform exhaustive search for the best match motion vector. In this work, a smart data reuse is proposed to reduce the amount of data to be fetched from memory to perform the Motion Estimation process. As a result, the memory access time will be reduced and the speed of the whole ME process will increase.

3. PROGRAMMABLE SYSTEM ON CHIP

Application Specific Integrated Circuit (ASIC) [24, 25] and Field Programmable Gate Array (FPGA) [26, 27] are the most recent effective tools to implementing different algorithms in hardware. Although implementing an algorithm using ASIC flow is much faster than the case of using FPGA, FPGA implementation provides a very high flexibility to its implemented algorithms. Additionally, the cost for implementing an algorithm using FPGA tool is much lower than the case of using ASIC flow implementation.

Recently, the capability of FPGA boards to download huge design is greatly improved due to the advance in the technology used in FPGA fabrication. As a result, the capacity of FPGA boards is grown exponentially. Consequently, a full embedded system can be implemented on a single FPGA chip. It is well known that video applications require huge data to be processed. One of the main useful tools for hardware implementation of such video algorithm is the use of FPGA. This is because FPGA boards can provide high capacity that allows a complete video system to be implemented on one FPGA board. A complete Full Search algorithm is implemented in hardware using special video Xilinx ZYNQ-7 ZC706 FPGA board. The selected FPGA board is suitable for processing huge video data due to its huge hardware capacity [28].

4. SYSTEM ARCHITECTURE

The whole implemented FSME system is shown in Figure (2-a). The Current Block (CB) and the search area are fetched from the external memory through the De-multiplexer (Demux). The Demux distributes the data to either the Local Memory or the Processing Element (PE) Array. The Local Memory consists of three sub-memories. Local Memory send candidate blocks to the Processing Array which contains the data of both the current and the candidate blocks. After the absolute differences are calculated inside the PE array, they will be sent to the Adder Tree to get the Sum of Absolute Difference (SAD). The SAD value is then sent to the compare unit to find the minimum SAD between the CB and all candidates in the search area. After the comparison, the position of the final minimum SAD is stored in the motion vector memory. The motion vector memory sends all the stored actual motion vectors to the main processor. The Control Unit controls all activities of the processor components. The RTL top-level schematic of the implemented FSME system is shown in Figure (2-b). Four inputs and one output are used in our design. The first input (INDATA) is the data coming from the external processor. INDATA is 128 data bus to carry 16 pixels data. Each pixel is 8 bits. The second input is the main system clock. The last two inputs are the DOIT and the TRIGLEPULSE which are used as enable and reset signals for our implemented design. The output of our design is the actual motion vector represents the position of the best match candidate block in the search area. The position is represented in clock cycle number that can be easily translated to a motion vector before sending to the main processor. Since we need 32 clock cycles for initializing the ME process followed by 1024 clock cycles to cover the whole search area, a total of 11 bites are needed to represent a total of 1056 clock cycles at the output terminal. The second RTL level and the gate level of the implemented FSME system are shown in Figure 3 and Figure 4, respectively.

It is worth mentioning that this architecture is scalable one, so it can be easily used for both H.264/AVC and H.265/HEVC standards. Local memory will have same size but the PE array will be 16×16 or 32×32 in case of using H.264/AVC and H.265/HEVC standards, respectively.

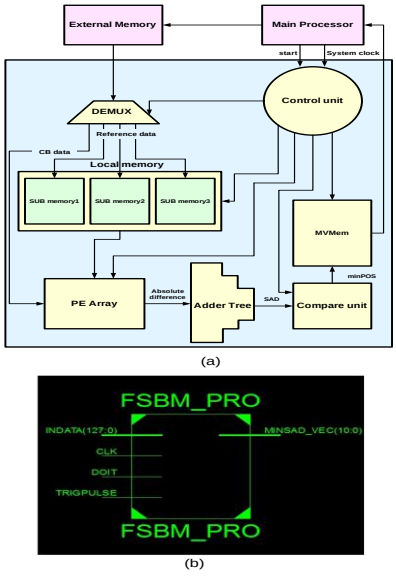

Figure 2: (a) The implemented FSME system block diagram. (b) RTL top-level schematic of the implemented FSME system.

A. Sum of Absolute Difference (SAD)

The output of the PE array is 256 Absolute difference values that are needed to be summed to form one Sum of Absolute Difference (SAD). To get the SAD value, all values output from the PE array will enter an adder tree to perform fast and parallel additions [5]. While we design our processor, we combine both the PE array and the adder tree units together to form the SAD unit as seen in Figure (5-a). The RTL top-level of the SAD unit is shown in Figure (5-b). It takes two inputs from the Local memory (the current block data (CBRin_data) and the reference block data (RBRin_data)). Both input have 128 bits each representing 16 pixels input at a time. The two inputs enter the first module (PE array) to produce 256 Absolute Differences (AD). Each AD is represented by 8 bits. These absolute differences enter the second module (adder tree) as one bus of size (8×256 bits) to get the final SAD represented by 16 bits. All other input signals are used to control the data flow inside the SAD unit.

Figure 6 represents the RTL second-level of the SAD Unit. It is worth mentioning that the number of bits in the output SAD is considered to avoid the overflow problem that may result if not enough number of bits in the output are considered.

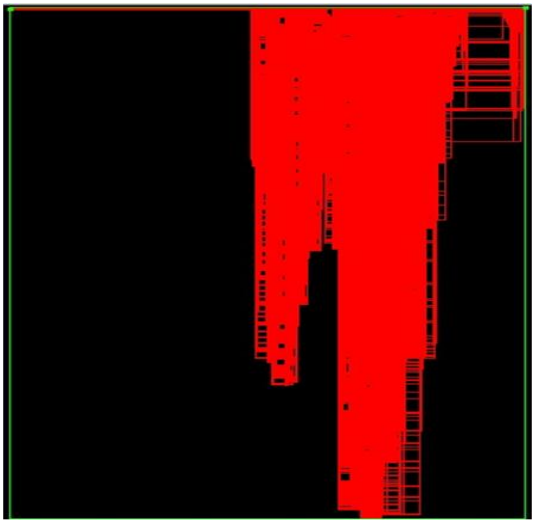

Figure 4: Gate level of the FSME system.

B. Demultiplixer and Local Memory Unit

The demultiplixer and local memory unit (demux_memory) is responsible for storing the data coming from the external memory. Demultiplixer takes the data from the external memory as a 16 pixels data bus and decide the data path either to the PE array (current block data) or to the local memory (the reference data) as seen in Figure 7. Figure 8 represents the RTL top-level of the demultiplixer and local memory unit. The value of select terminal (sel_demux) decides the data path. If the select terminal value is 0, the demultiplixer will open the path for the CB data to go to PE Array. If it is 1 or 2 or 3, the reference data will go to sub memory1, sub memory2, sub memory3 in local memory, respectively. The search area will enter to the PE array using a 16 pixel data bus (128 bits) as one column at a time from each internal local memory. Counter terminal in Figure 8 is used to select a particular column to fill the search area into the PE array. Figure 9 represents the architecture of each sub-memory. One of the main advantages of our local memory is its design simplicity. Each sub-memory consists of 16×16 register array. Each register is 8-bits in length to carry a value of one pixel. The data enter each sub-memory as 16 pixels each clock cycle from the bottom of the sub-memory. Each clock cycle, a new 16 pixels enter the sub-memory from the bottom shifting all old data upward. Once one sub-memory is filled the select terminal will switch to fill the next one. After filling the first left sub-memory, a counter will start counting column by column to fill the PE array with the reference block data.

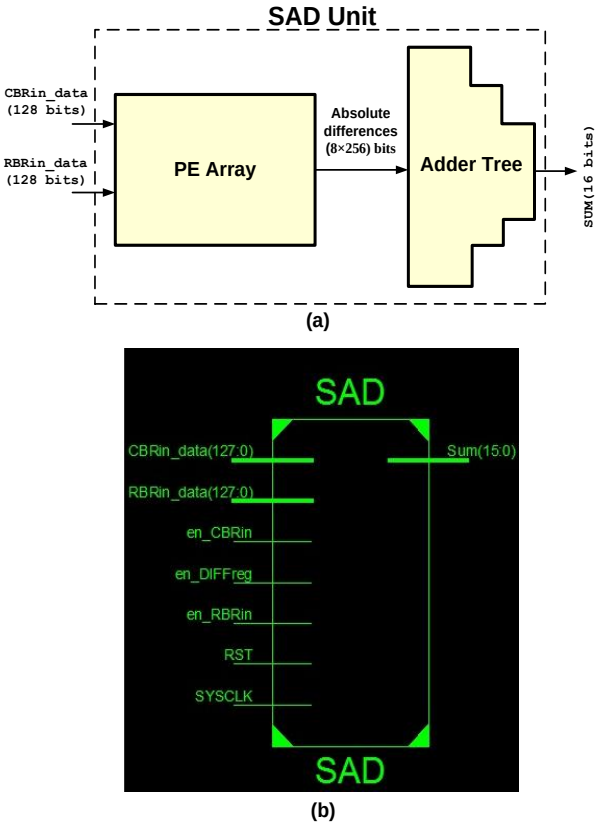

Figure 5: RTL top-level of SAD Unit.

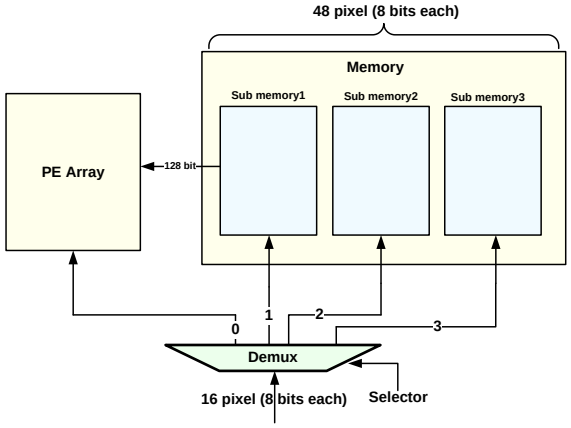

Figure 7: Top-level of demux_memory Unit.

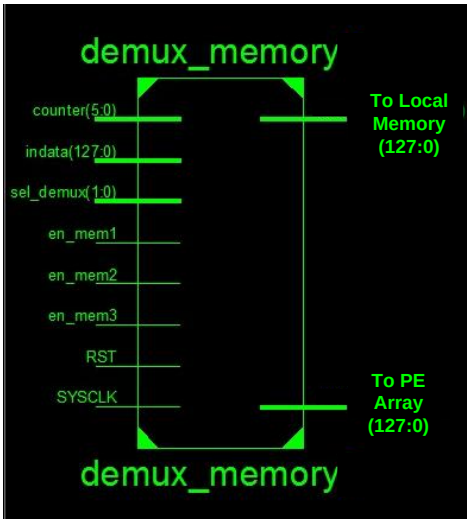

Figure 8: RTL top-level of Memory and Demux Unit.

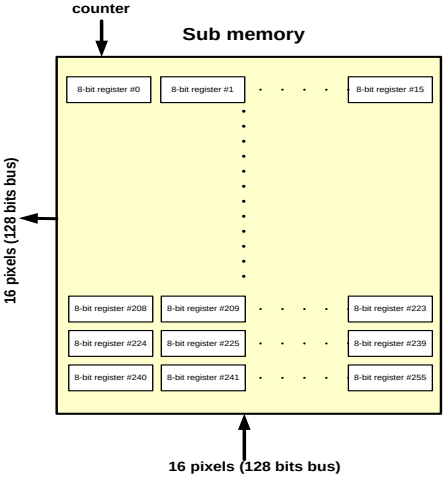

Figure 9: The internal architecture of each sub memory.

C. Compare Unit

The compare unit gets the value of the current SAD and its corresponding position to compare with the minimum SAD so far from previous reference blocks search. If the current SAD is less than the minimum SAD so far, the minimum SAD so far will be updated and the corresponding minimum position (represented by the clock cycle number), accordingly. Figure 10 and Figure 11 shows RTL top level and second level of the compare unit, respectively.

As seen in Figure 11, the 54-bits register is used to store two vectors. The first vector represents the minimum SAD so far and its corresponding position. The second vector represents the input SAD and its position. This register is initialized by maximum values at the beginning of the system operation. Those stored values of the SADs and positions are inputs to the COMPTRANS unit which does the comparison operation. The COMPTRANS unit uses the COMPARE16BIT to make the comparison and pass the result to two multiplexers (i.e., MUX 16 and MUX 10). The MUX 16 is responsible for passing the value of the minimum SAD so far and its corresponding minimum position will pass via MUX 10.

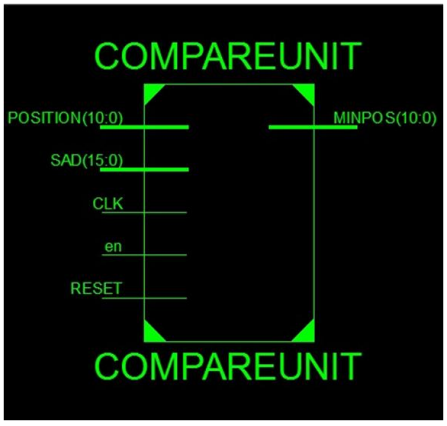

Figure 10: RTL top-level of Compare Unit.

D. Motion Vector Memory

Once the search for the optimum best match is completed, the actual motion vector corresponding to the best match candidate block in the search area will be stored in the motion vector memory (MVMEM). The implemented video processor can process HDTV video sequence of length (720 pixels/line)×(486 lines/frame). As a result, the MVMEM consists of 1395 registers, which equals to the number of the actual motion vectors per frame. For higher resolution video sequence, the MVMEM size should increase. MVMEM takes an input of 11-bits, representing the position of the best match candidate block, and stores it using FIFO principle. The

MVMEM is very simple, no addressing complexity is used. Figure 12 represents the RTL top level of the MVMEM.

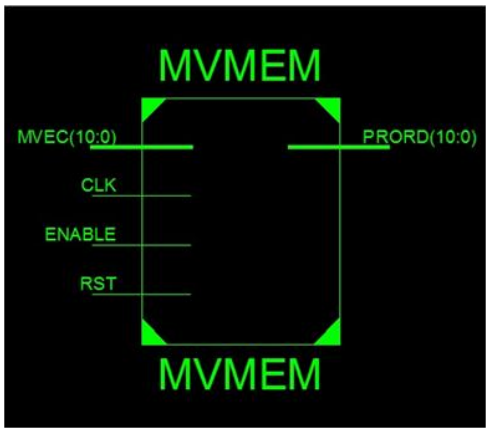

Figure 12: RTL top-level of MVMEM.

E. Control Unit

The control unit is the most important part in the processor. It controls all components inside the processor by providing the required control signals for each component. The control unit consists of two modules, the Up Counter (UPCOUNT) and the Control Signals controller (CScontroller). The control unit has three inputs, enable, reset and the system clock. The outputs are all the needed signals to control all components of the video processor. Figure 13 shows the RTL second-level of control Unit. The OR gate takes two reset input signals. One of them is coming from the main processor, and the other reset input is coming from the control unit (CScontroller). The CScontroller generates a reset signal after each full searching of a current block. Consequently, counter (upcount) has to restart counting the clocks of the new search process.

5. Implementation and Discussion

The implemented video video processor can process HDTV video sequence of length (720 pixels/line)×(486 lines/frame). The frames are divided into blocks of size 16×16. The maximum number of padding pixels (P_{max} in Figure 1) is 16. As a result, the size of the search area is 32×32. The proposed co-processor architecture is implemented using VHDL and full functional verification was performed using Modelsim tool with actual video sequence as the test input. For the hardware implementation, the architecture of the proposed co-processor is checked using FPGA tool. The architecture in [18] is implemented using FPGA tool and same environment used for the proposed co-processor.

Table 1: A Summary of implementations using Xilinx ZYNQ-7 ZC706 FPGA.

	[18]	**Proposed system**
	FPGA	FPGA
	Xilinx ZYNQ-7 ZC706 Board	Xilinx ZYNQ-7 ZC706 Board
	28 nm	28 nm
# of PEs	256	256
Block Size	16	16
# of Slice LUTs	13.01K	11.66K
# of Slice Registers	10.56K	14.59K
# of LUT Flip Flop	23.57K	26.25K
Max. Freq.	119.72MHz	123.53MHz
Data Reuse Level	Levels A and B	Levels A and B

For the FPGA implementation, Xilinx ZYNQ-7 ZC706 Evaluation Board with (7z045ffg900-2) FPGA chip is selected to implement the synthesized VHDL code of the proposed co-processor and the architecture in [18]. Table 1 summarizes and compares the proposed video system with the fast motion estimation architectures in [18]. The proposed video system is faster than the architecture in [18] in terms of both the throughput and the operating hardware frequency. This is due to the novel and smart procedure of loading both the current block pixels values and the search area inside the PE array. The maximum operating frequency of the proposed video processor is 123.53MHz. This guarantees real time ME for high-resolution video sequences applications such as HDTV and SDTV broadcast.

6. Conclusion

A Full Search Motion Estimation video processing system is designed and implemented using FPGA tool. The proposed motion estimation video system can perform ME for 30 fps of HDTV video at 123.53 MHz. the proposed ME video processor (system) can be generalized to be used for any higher resolution video sequence under a constrain of increasing the capacity of the MVMEM at no other addition hardware complexity. The proposed architecture improves the speed and hardware complexity compared to the state of the art fast ME techniques implemented on FPGA tools. This allows the proposed ME video system to be used for any high accuracy real time video applications such as HDTV and SDTV broadcastings. The future work will address the

possibility of adding more techniques and hardware to speed up the ME process, so that, the new architecture can be easily used for high resolution video sequences. The new architecture should maintain regular data flow in order to be easily implemented.

ACKNOWLEDGMENT

The author acknowledges the support of the Deanship of Scientific Research – University of Bahrain – Bahrain for supporting this work under the project number 20015/11. The author thanks his student, Najeeba Mohammed Jaffer, for her efforts in simulations.

REFERENCES

[1] J. Ohm, G. J. Sullivan, H. Schwarz, T. Thiow Keng, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 22, pp. 1669-1684, 2012.

[2] Z. Hao and M. Zhan, "Fast Intra Mode Decision for High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology,*, vol. 24, pp. 660-668, 2014.

[3] Y. Ismail, J. B. McNeely, M. Shaaban, H. Mahmoud, and M. A. Bayoumi, "Fast Motion Estimation System Using Dynamic Models for H.264/AVC Video Coding," IEEE Transactions on Circuits and Systems for Video Technology, , vol. 22, pp. 28-42, 2012.

[4] "High Efficiency Video Coding (HEVC) Text Specification Draft 6," ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 WP3, Feb. 2012.

[5] Y. Ismail, W. El-Medany, H. Al-Junaid, and A. Abdelgawad, "High Performance Architecture for Real-time HDTV Broadcasting," Journal of Real-Time Image Processing, Springer, ISSN: 1861-8200 (print version), and ISSN: 1861-8219 (electronic version), May 27, 2014.

[6] H. Amirpour, A. Mousavinia, and N. Shamsi, "Predictive Three Step Search (PTSS) algorithm for motion estimation," in *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, 2013, pp. 48-52.

[7] H. A. Choudhury and M. Saikia, "Reduced three steps logarithmic search for motion estimation," in *2014 International Conference on Information Communication and Embedded Systems (ICICES)*, 2014, pp. 1-5.

[8] L. Renxiang, Z. Bing, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 438-442, 1994.

[9] P. Lai-Man and M. Wing-Chung, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 313-317, 1996.

[10] Z. Shan and M. Kai-Kuang, "A new diamond search algorithm for fast block matching motion estimation," in *Proceedings of 1997 International Conference on Information, Communications and Signal Processing, 1997. ICICS.*, 1997, pp. 292-296 vol.1.

[11] C. Chun-Ho and P. Lai-Man, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1168-1177, 2002.

[12] C. Changryoul and J. Jechang, "Successive Elimination Algorithm for Constrained One-bit Transform Based Motion Estimation Using the Bonferroni Inequality," *IEEE Signal Processing Letters*, vol. 21, pp. 1260-1264, 2014.

[13] L. Hwal-Suk, J. Jik-Han, and P. Dong-Jo, "An effective successive elimination algorithm for fast optimal block-matching motion estimation," in *15th IEEE International Conference on Image Processing, ICIP 2008*, pp. 1984-1987, 2008.

[14] Y. Ismail, M. Shaaban, J. B. McNeely, and M. A. Bayoumi, "An Efficient Adaptive High Speed Manipulation Architecture for Fast Variable Padding Frequency Domain Motion Estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 1239-1248, 2011.

[15] S. Goel, Y. Ismail, and M. A. Bayoumi, "Adaptive search window size algorithm for fast motion estimation in H.264/AVC standard," in *48th Midwest Symposium on Circuits and Systems*, pp. 1557-1560 Vol. 2, 2005.

[16] J. Sung-Tae and L. Sang-Seol, "A 4-way pipelined processing architecture for three-step search block-matching motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 50, pp. 674-681, 2004.

[17] D. Xu, J. M. Noras, and W. Booth, "A simple and efficient VLSI architecture for a very fast high performance three step search algorithm," in *IEE Colloquium on High Performance Architectures for Real-Time Image Processing (Ref. No. 1998/197)*, pp. 6/1-6/6, 1998.

[18] Sumeer Goel, Yasser Ismail, and M. Bayoumi, "High-speed Motion Estimation Architecture for Real-time Video Transmission," The Computer Journal, vol. 55, pp. 35-46, 2011.

[19] L-W. Lee, J-F. Wang, J-Y. Lee, and J. D. Shie, "Dynamic search-window adjustment and interlaced search block matching algorithm," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 3, No. 1, pp:85-87, Feb. 1993.

[20] Srinivasarao, B.K.N.; Chakrabarti, I., "A parallel architectural implementation of the fast three step search algorithm for block motion estimation", International Multi-Conference on Systems, Signals and Devices, 2008. IEEE SSD 2008. 5th, pp. 1-6, 2008.

[21] S. Marlow, J. Ng, and C. McArdle, "Efficient motion estimation using multiple log searching and adaptive search windows," in Proc. of the IEE Intl. Conf. on Image Processing and its Applications, Vol. 1, pp. 214-218, 1997.

[22] M. Alkanhal, D. Turaga, and T. Chen, "Correlation based search algorithms for motion estimation," in Proc. of the Picture Coding Symp., pp. 99-102, April 1999.

[23] Sumeer Goel, Yasser Ismail, and Magdy A. Bayoumi, " High-speed Motion Estimation Architecture for Real-time Video Transmission," Oxford Journals - The Computer Journal (2012) 55(1): 35-46 first published online April 29, 2011.

[24] Batinaa, L., Orsa, S.B., Preneela, B. and Vandewalle, J., "Hardware architectures for public key cryptography", Integration, the VLSI Journal, Vol. 34, pp. 1-64, 2003.

[25] Bertoni, G., Breveglieri, L., Koren, I., Maistri, P. and Piuri, V., "Error analysis and detection procedures for a hardware implementation of the advanced encryption", *Standards IEEE Trans. Comput.*, Vol. 52 No. 4, pp. 492-505, 2004.

[26] Hodjat, A. and Verbauwhede, I. , "A 21.54 Gbits/s fully pipelined AES processor on FPGA", *12th Annual IEEE Symposium on Field – Programmable Custom Computing Machines, Napa, CA, USA*, pp. 308-309, 2004.

[27] Elias, G., Miri, A. and Yeap, T.H., "On efficient implementation of FPGA-based hyperelliptic curve cryptosystems", *Computers and Electrical Engineering*, Vol. 33, pp. 349-366, 2007.

[28] http://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html, 15 Dec., 2015.

Dr. Yasser Ismail received the B.Sc.degree in Electronics & Communications Engineering from Mansoura University, Mansoura, Egypt, in 1999, the M.Sc. degree in Electrical Communications from Mansoura University, Mansoura, Egypt, in 2002, the M.Sc. degree in Computer Engineering from University of Louisiana at Lafayette, Louisiana, USA, in 2007. Dr. Yasser Ismail got his Ph.D. from the University of Louisiana at Lafayette in May 2010. Dr. Yasser Ismail worked as an assistant professor in Umm Alqura University – KSA from 2010 to 2012. He is currently working as an assistant professor in University Of Bahrain (UOB) - Bahrain. Dr. Yasser permanently working at the Electronics and Communications Engineering Department – Faculty of Engineering – Mansoura University – Mansoura – Egypt. Dr. Yasser is served as a reviewer for several conferences and journals, including ISCAS 2010, ICIP 2010, ICIP 2011, ICECS2013, Transaction on Circuit and System for Video Technology (TCSVT), and IEEE Transactions on Image Processing, and Signal Processing. He has also gained many valuable projects from KSA, NSF, and Bahrain. Dr. Yasser served in the organizing committee of 2013 IEEE International Conference on Electronics, Circuits, and Systems (ICECS2013). His research of interest includes video processing, digital signal processing, Robotics, RFID, Localization, VLSI, FPGA, wireless communication systems, and low power embedded systems.

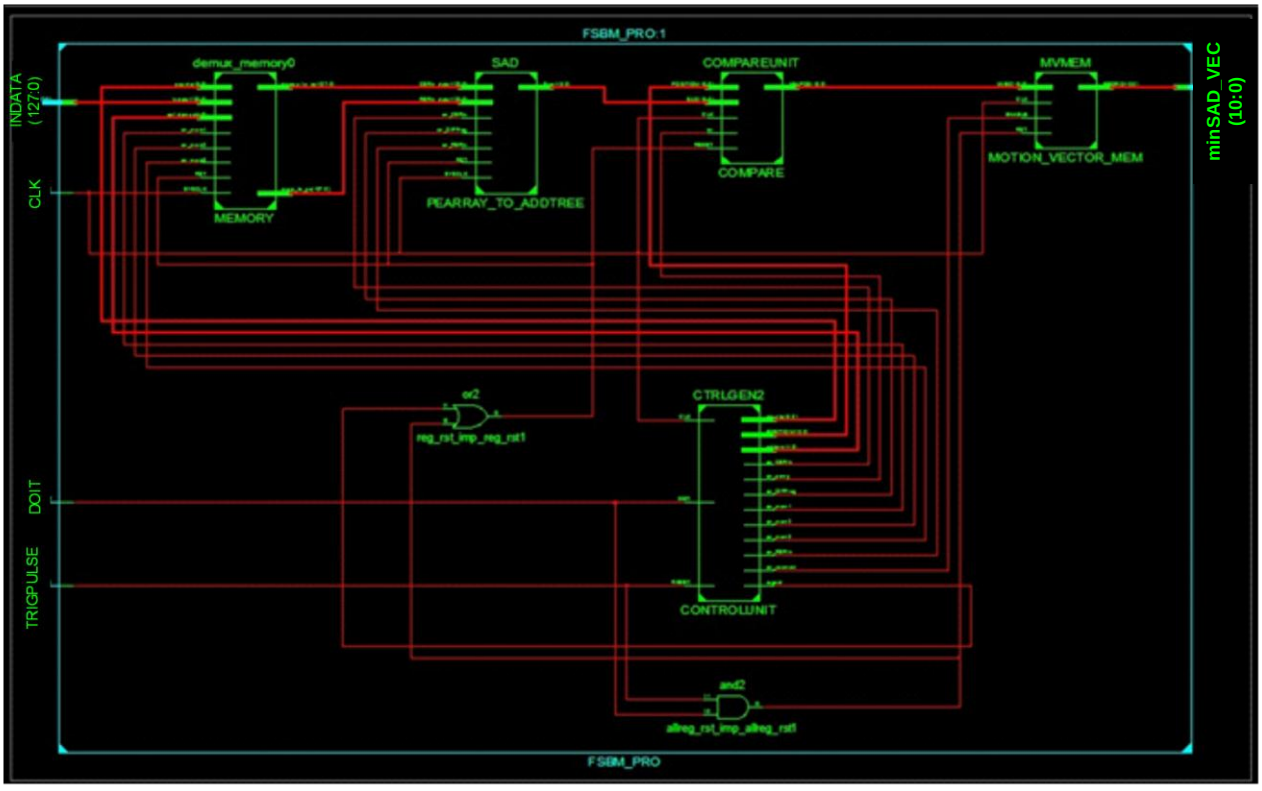

Figure 3: RTL second-level schematic of the FSME system.

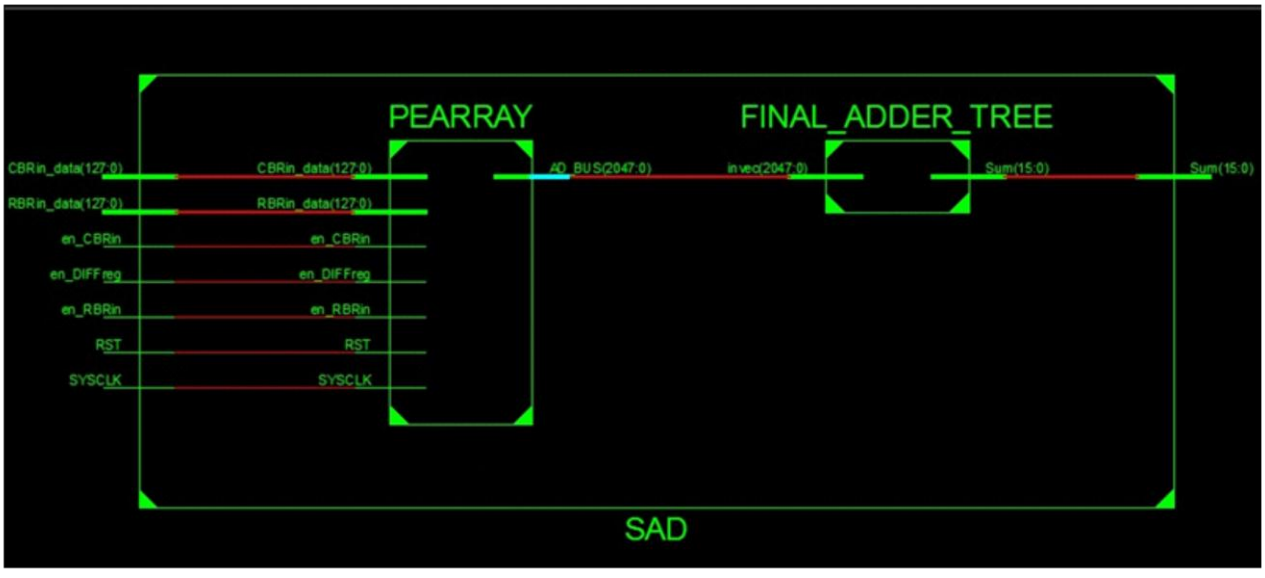

Figure 6: RTL second-level of SAD Unit.

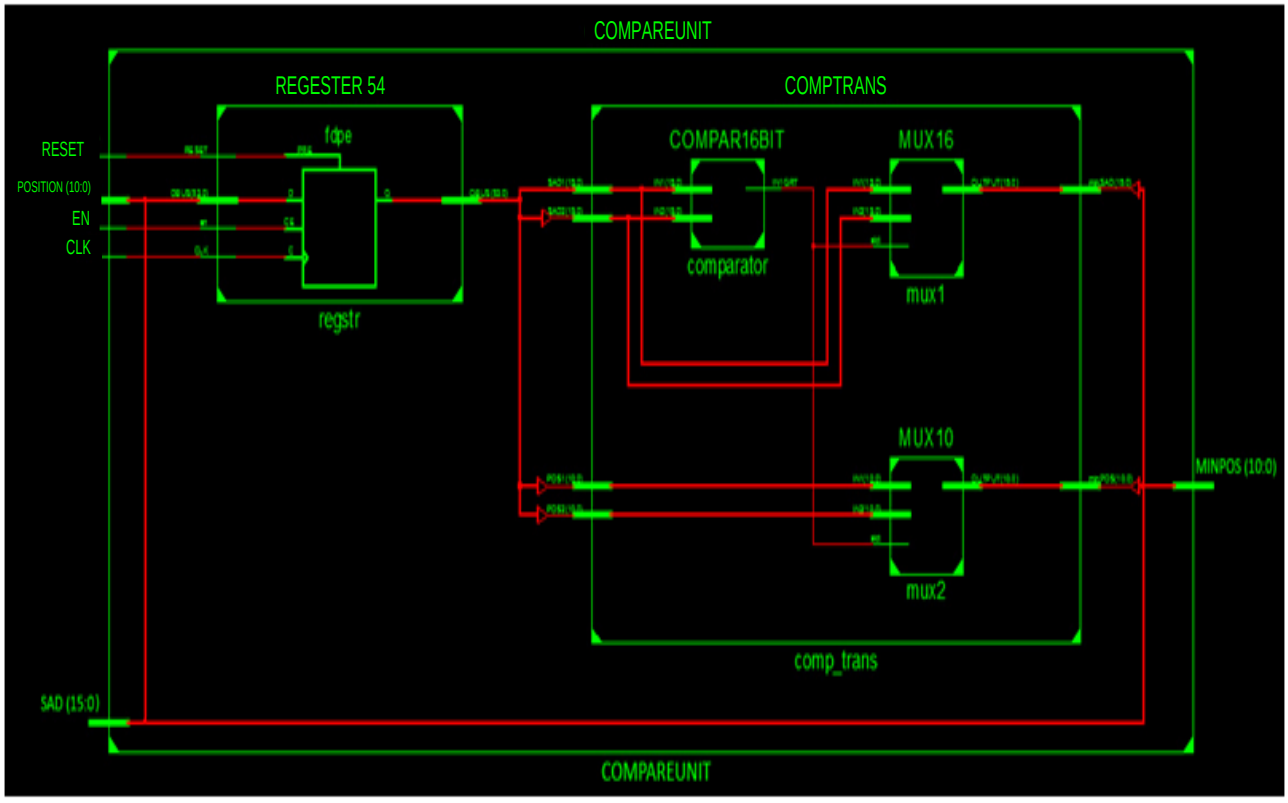

Figure 11: RTL second-level of Compare Unit.

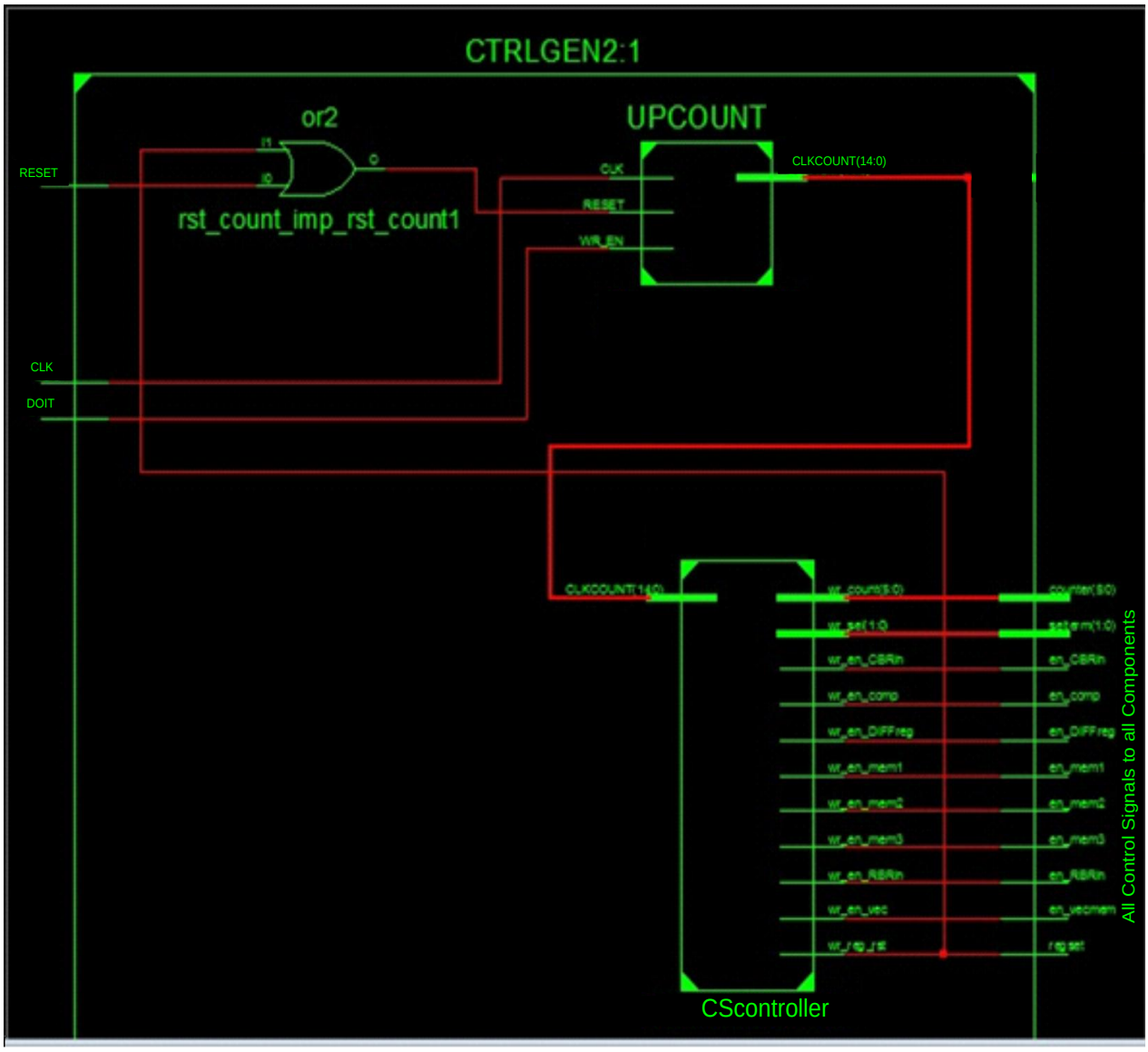

Figure 13: RTL second-level of control Unit.