

FPGA Implementation of Low Power Decompressed Real Time Video Segmentation

Saritha.K^{*1}, R.Lakshmi^{#2}

^{*1}P.G Student, ^{#2}Assistant Professor, Department of ECE, Asan memorial college of engineering, Chengalpettu, Tamil Nadu, India

Abstract: Background identification is common feature in video system .This project deals with the hardware implementation of the Gaussian mixture model algorithm of decompressed real time video. As this project is mainly focused on receiver end, decompression plays an important role. At the receiver side only compressed video will be captured. So the original video should be got back through decompression. Inverse Discrete Cosine Transform is used here, since it is effective and less noise interfered technique. Also scaling of image is also done so that the video which is taken by the camera will be compactable with the display devices. So this project in cooperates three processes called segmentation decompression and scaling with low power consumption. The reduction in power is proved using Quartus.

Keywords: Computer vision, Image motion analysis, Object detection, Subtraction techniques, DCT.

I. INTRODUCTION

THE REAL-TIME detection of moving objects in a video sequence has many important applications, such as video surveillance and traffic monitoring. Various algorithms for the detection of moving objects have been developed during the years. Here we are doing the detection of the dynamic parts of a scene by comparing two consecutive frames. Background subtraction algorithms detect moving objects by analyzing the difference between a frame and a reference model that includes the static parts of the scene. Dynamic part indicates the foreground and static part indicate the background. Target of these algorithms is the determination of the Bg model that also update the changes of the scene. To adapt the Bg model, using algorithm based on median filtering. These algorithms take into account changes in the lighting of the Bg and the eventuality of moving objects which become statics.

The Bg is represented with a statistical model based on a single Gaussian distribution per pixel. This allows a pixel by-pixel model of the Bg with larger variance for pixels that experience wide changes of lighting. The algorithm used here is known as Gaussian mixture model (GMM), and provides good performances in both presence of illumination changes and multimodal Bg. A multimodal Bg is characterized by objects showing repetitive motions, e.g., waves, moving leaves, or flickering light. When a pixel lies in a region where a repetitive motion occurs, its brightness oscillates between two or more values. This results in false Fg detection in most algorithms. On the contrary, when the GMM algorithm is used, the intensity distribution of the pixel is modeled using two or more Gaussian distributions and the problem of false Fg detection is solved. Due to the good performances, the GMM algorithm has been selected as the Bg detection algorithm in the OpenCV, Open source Computer Vision software library. In order to generate the updated Bg model, the GMM algorithm processes the video streams by computing a great number of parameters for each pixel of each frame, with a computational burden unreachable by computers in real-time applications. GMM algorithms allow real-time processing of HD videos and comply with the OpenCV algorithm. Here we are doing Bg identification circuits process gray scale videos. The proposed circuit has been experimentally validated through experimental measurements of the working frequency and implementing two running on-line video systems equipped with monitor and video sensor. Implementation

will be done through virtex6 FPGA kit. Main features of this project are an innovative, hardware-oriented, formulation of the GMM equations that allows hardware saving and speed improvement without affecting the output of the GMM algorithm. The implementation of the above cited GMM equations in an FPGA-oriented circuit that outperforms the existing system. The experimental demonstration of the proposed FPGA Circuit in running on-line video systems.

II. RELATED WORK

Several techniques for background subtraction and shadow detection have been proposed in the past years. Background detection techniques may use grayscale or color images, while most shadow detection methods make use of chromaticity information. The car tracking system of Koller et al. [9] used an adaptive background model based on monochromatic images filtered with Gaussian and Gaussian derivative (vertical and horizontal) kernels. McKenna et al. [11] proposed a background model that combines pixel RGB and chromaticity values with local image gradients. In their W4 system, Haritaoglu and collaborators [6] used grayscale images to build a background model, representing each pixel by three values; its minimum intensity value, its maximum intensity value and the maximum intensity difference between consecutive frames observed during the training period. Elgammal et al. [4] used a nonparametric background model based on kernel based estimators, that can be applied to both color and grayscale images. KaewTrakulPong and Bowden [7] used color images for background representation. In their method, each pixel in the scene is modeled by a mixture of Gaussian distributions (and different Gaussians are assumed to represent different colors). Cucchiara's group [3] used a temporal median filtering in the RGB color space to produce a background model. Shadow detection algorithms have also been widely explored by several authors, mostly based on invariant color features that are not significantly affected by illumination conditions. Julio improved an existing method for background subtraction and proposed a novel technique for shadow detection in grayscale video sequences. In our approach, the normalized cross-correlation is applied to foreground pixels, and candidate shadow pixels are obtained. A refinement process is then applied to further improve shadow segmentation. All these proposals required more area and more power. So this paper proposes an effective and efficient algorithm to reduce area and power.

III. PROPOSED ALGORITHM

There are various methods for efficient segmentation of moving objects. Adaptive Gaussian mixture based background learning method is relatively a more robust algorithm due to its high adaptation capability to cluttered rapidly changing background scenes. Most of the visual surveillance systems include monitoring of outside environments like garages, gardens and open vast areas, Gaussian mixture model would be an ideal fit to be used for object segmentation. The model is adaptive online background mixture model that can robustly deal with lighting changes, repetitive motions, clutter, introducing or removing objects from the scene and slowly moving objects. Gaussian distributions represent each pixel in the model. Due these promising features, we implemented and integrated this model in our visual surveillance system. In this model, the values of an individual pixel over time is considered as a "pixel process" and the recent history of each pixel, $\{X_1, \dots, X_t\}$, is modeled by a mixture of K Gaussian distributions. The probability of observing current pixel value then becomes

$$P(X_t) = \sum_{i=1}^K w_{i,t} \eta_i(X_t; \mu_{i,t}, \Sigma_{i,t})$$

where $w_{i,t}$ is an estimate of the weight (what portion of the data is accounted for this Gaussian) of the i th Gaussian ($G_{i,t}$) in the mixture at time t , $\mu_{i,t}$ is the mean value of $G_{i,t}$ and $\Sigma_{i,t}$ is the covariance matrix of $G_{i,t}$ and η is a Gaussian probability density function

$$\eta(X_t; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu)^T \Sigma^{-1} (X_t - \mu)}$$

Decision on K depends on the available memory and computational power. Also, the covariance matrix is assumed to be of the following form for computational efficiency

$$\Sigma_{k,t} = \sigma_k^2 \mathbf{I}$$

This assumes that red, green, blue color components are independent and have the same variance. The procedure for

detecting foreground pixels is as follows. At the beginning of the system, the K Gaussian distributions for a pixel are initialized with predefined mean, high variance and low prior weight. When a new pixel is observed in the image sequence, to determine its type, its RGB vector is checked against the K Gaussians, until a match is found. A match is defined as a pixel value within γ (≈ 2.5) standard deviation of a distribution. Next, the prior weights of the K distributions at time t, $w_{k,t}$, are

$$w_{k,t} = (1 - \alpha)w_{k,t-1} + \alpha(M_{k,t})$$

Where α is the learning rate and $M_{k,t}$ is 1 for the matching Gaussian distribution and 0 for the remaining distributions. After this step the prior weights of the distributions are normalized and the parameters of the matching Gaussian are updated with the new observation such as

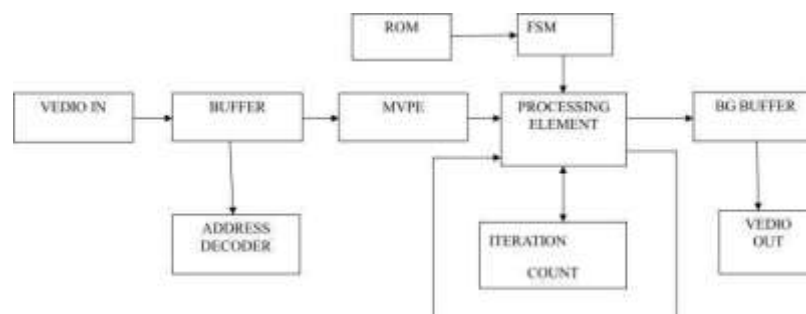
$$\begin{aligned} \mu_t &= (1 - \rho)\mu_{t-1} + \rho(X_t) \\ \sigma_t^2 &= (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \\ \rho &= \alpha\eta(X_t|\mu_k, \sigma_k) \end{aligned}$$

If no match is found for the new observed pixel, the Gaussian distribution with the least probability is replaced with a new distribution with the current pixel value as its mean value, an initially high variance and low prior weight. In order to detect the type (foreground or background) of the new pixel, the K Gaussian distributions are sorted by the value of w/σ . This ordered list of distributions reflect the most probable backgrounds from top to bottom since by equation of $w_{k,t}$ background pixel processes make the corresponding Gaussian distribution have larger prior weight and less variance. The first distributions are chosen as the background model, where

$$B = \operatorname{argmin}_b \left(\sum_{k=1}^b w_k > T \right)$$

T is the minimum portion of the pixel data that should be accounted for by the background. If a small value is chosen for T, the background is generally unimodal. Video consists of consecutive frames, 30fps. One frame is processed at a time. Corresponding mean, weight, variance will be calculated. Each frame is having k distributions. Each distribution is having the above parameters. Since calculating all the distribution is time consuming process, only three distributions is considered. One frame that is its mean, variance, weight are stored as the reference values. Other frames are compared with the reference, and then background is identified.

VI. IMPLEMENTATION OF THE ALGORITHM USING FPGA



Video consists of almost thirty frames per second. One frame processed at a time. So each frame is stored in a buffer. Motion Vector Processing Element calculates gaussian parameters of each frame and stored in buffer. Each of the parameters are calculated using same shifter and multiplier. Each frame is shifted to the background memory after processing. In buffer frames are stored in the appropriate address. So address decoder is provided to access the frames. To access the frame column and row address of the memory in which it is located should be known. The speed of the system

is measured in terms of the speed at which the frames are transferred to the buffer for the execution. PGA circuit is described in Verilog code and implements the GMM algorithm. Gaussian distributions for each pixel ($k = [1, 2, 3]$). The circuits process 13 parameters per pixel (the luminance value and 12 Gaussians parameters for the statistical model of the pixel). The software algorithm proposed in the OpenCV library represents the Gaussians parameters as double precision (64 bits) floating point numbers. Unfortunately, a hardware implementation that uses 64 bits floating point signals is not feasible. The required circuitry for the calculation of double precision signals is too large and slow for an effective hardware implementation. Further, the required bandwidth toward the memory is too high. As example, if the memory throughput is 128 bits, 13 clock cycles are needed to load and store the parameters for each pixel. The examination of the GMM algorithm reveals that the signals have limited dynamics. Mean, variance, and weight range in $[0, 255]$, $[0, 127]$, and $[0, 1]$, respectively. When dealing with limited dynamics signals, a fixed point representation instead of a floating point one provides improved performances while reducing hardware complexity and the bandwidth toward the memory. However, a reduced number of bits per signal could involve a poor quality of the processed images. So we use fixed point representation of the numbers is used and, in order to accurately determine the representation of the signals, the weights of the MSB have been fixed in accordance to the signals dynamics while the weights of the LSB have been chosen by using a word length reduction procedure directed to optimize the tradeoff between accuracy and hardware complexity. The optimized fixed point representation together with an innovative formulation of the GMM algorithm equations allows the design of a Bg identification circuit with high working frequency and low utilization of logic resources. For every frame of each video stream. $I_{ref}(m, n)$ is the reference foreground/ background image obtained with a double precision floating point version of the OpenCV GMM algorithm. $I(m, n)$ is the foreground/background image produced, calculated thru ModelSim simulations. For a binary image, the square of the difference between $I(m, n)$ and $I_{ref}(m, n)$ has only two possible values: "1" if the pixel has been differently classified in I and I_{ref} ; "0" if the classification of the pixel is the same. It is equivalent to the ratio of the number of pixels differently classified in I with respect to I_{ref} to the total number of pixels in a frame. MSE is a common measure of videos quality like peak signal-to-noise ratio. The GMM algorithm equations detailed require the implementation of nonlinear functions. It would be too complex for the realization of a fast and energy efficient circuit. Hardware implementations are therefore based on the manipulation and simplification of the above-cited equations.

Match: Verifies the match condition for the three Gaussian distributions and is composed, as shown in Figure 3.4, by three identical units (one for each Gaussian). The circuit employs a subtractor, a squarer, a multiplier, and a comparator.

Control Logic: Sorts the Gaussians in increasing $IF_{k,t}$ order and establishes, which Gaussian has to be updated. The $IF_{k,t}$ values are sorted by using three comparators and few logic gates. The output signals $G1$, $G2$, and $G3$ represent the first, second, and third Gaussian in increasing $IF_{k,t}$ order while NM (No-Match) and GU (Gaussian Update) establish which Gaussian must be updated.

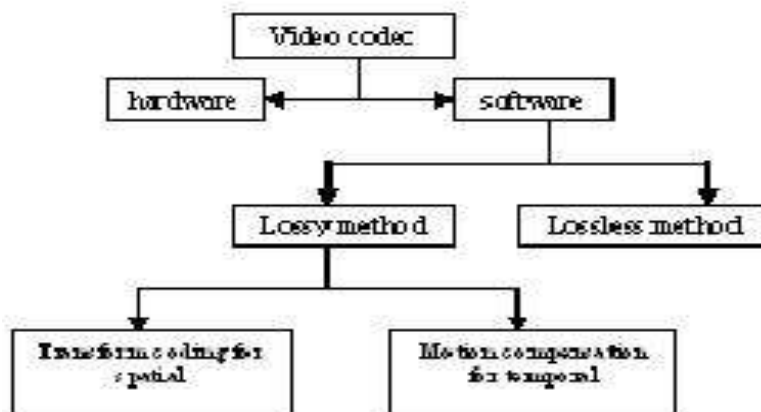
Weight, Mean, Variance, Matchsum, and No_match:

These circuitual blocks implement the parameters update equations. If the match condition is verified, Weight, "Mean", and "Variance" blocks update the parameters. The "Matchsum" block is the circuit that updates the matchsum signal. The matchsum signal is a counter, associated to every Gaussian, introduced in the OpenCV GMM algorithm. The matchsum signal counts the number of times that a given Gaussian is matched. Introduction of the matchsum entails the synthesis of three counters that are not on the critical path. The only drawback is the increase of the circuit area and of the power dissipation. The use of the matchsum counters improves the initial learning phase of the OpenCV GMM algorithm. If the pixel matches no Gaussians, the Gaussian with the lowest weight is replaced by a new distribution with the current pixel value as its mean, an high variance, and a low weight. During the initial phase of the video processing, this causes a very slow learning of the Bg. The proposed FPGA implementation of the OpenCV GMM algorithm is synthesized and implemented on Virtex6 Xilinx FPGA. ModelSim PE has been used to perform the circuit simulations. The performances analysis for both circuits have been conducted, including input and output registers that synchronize the circuits and provide timing performances that are not dependent on the I/O pads. The input and output registers are not counted has pipeline levels. When the number of pipeline levels is not indicated or is indicated as zero only the input and output registers are included into the circuit. When the number of pipeline levels is indicated as one, a further level of registers has been inserted into the circuit, breaking the maximum combinatorial delay, and improving the maximum working frequency. In this case, the optimal placement of the registers has been obtained with an iterative procedure that mixes hand placement of registers.

V. DECOMPRESSION

Digital video has become very important form of information technology and is now used in many different areas, such as board casting, teleconferencing, mobile telephone, surveillance, and entertainment. People now expect to be able to access video through a wide range of different devices and over various networks. To provide these kinds of services we must know what is video compression and decompression for storage and transmission (K.Holtz). Compression of video imagery has become a necessity and very important because of transmission and storage of uncompressed video would be extremely costly and impractical. For instance, a video sequence running for (90) minutes, at (25) frames per second, at standard resolution of (720 *576), and with (24) bit per pixel would require (134369280000) bit or approximately 156.43 gigabytes". This is not a problem if we only wish to access and deal with video through high - end systems with a lot of storage and network band width. However, since it is desired that video will be accessible from a wide range of devices having different capabilities and connected to different networks, therefore some form compression for digital video is required.

Compression refers to the process of reducing the number of bits required to represent the image and video comes in two forms lossless and lossy. The lossless compression is a process to reduce image or video data for storage and transmission while retaining the quality of original image (i.e. the decoded image quality is required to be identical to image quality prior encoding. In lossy compression, on the other hand, some information present in the original image or video is discarded so that the original raw representation of image or video can only be approximately reconstructed from the compressed representation with high compression ratio. For more compression can be achieved with approximate quality to source image lossy compression is almost usually used than lossless compression to compress digital video.

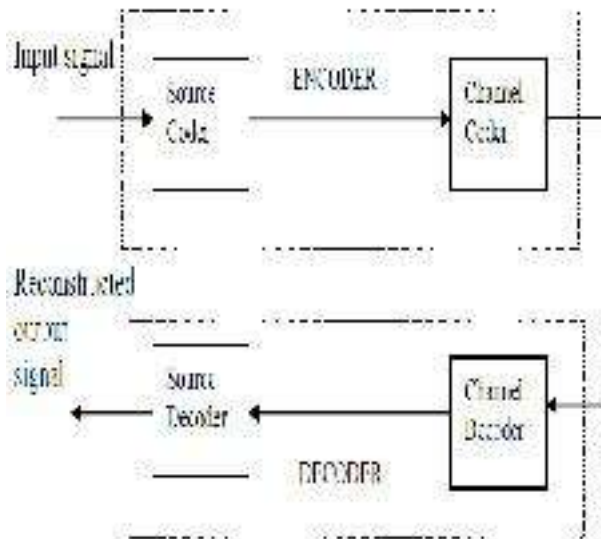


Video Codec is a codec i.e. computer program that compress/ decompress digital video data according to given file format and can be either software application or part of hardware. They process the video through complex algorithm that compress and decompress video files. Video codec implemented in hardware is most efficient way to code and decode video file, they are faster and demands much less processor power than a software code. After we are introduced to redundancy which is contained in image sequence of video file we are used DCT to reduce spatial redundancy, and differences between successive frames to reduce temporal redundancy by using skew measurement to measure asymmetry between them by calculating standard deviation and mean which tell us about contrast and brightness respectively to compress video file and stored it in another form . To reconstruct video file again we are needed to use IDCT

VI. DCT and IDCT

Discrete Cosine Transform (DCT) is a lossy compression scheme in which a $N \times N$ image block is transformed from the spatial domain to the DCT domain. DCT decomposes the signal into spatial frequencies components called DCT coefficients. The lower frequency DCT coefficients appear toward the upper left-hand corner of the DCT matrix and the higher frequency coefficients are in the lower right-hand corner of the DCT matrix. The Human Visual System (HVS) is less sensitive to errors in high frequency coefficients than it is for lower frequency coefficients. Because of this , the higher frequency components can be more finely quantized. This is done by the quantization matrix.

Each value in this matrix is pre-scaled by multiplying by a single value, known as the quantizer scale code. This value may range in value from 1-112, and is modifiable on a macroblock basis. Dividing each DCT coefficient with an integer scale factor and rounding the results does quantization. This sets the higher frequency coefficients (in the lower right corner, which are less significant to the compressed picture) to zero by quantizing in larger steps. The low frequency coefficients (in the upper left corner, which are more significant to the compressed picture) are quantized in smaller steps. The goal of quantization is to force as many of the DCT coefficients to zero, or near zero, as possible within the boundaries of the prescribed bit-rate and video quality parameters. Thus since quantization throws away some information it is a lossy compression scheme



The 12 Bit Signed Input Pixels Will Provide A 16 Bit Signed Output coefficient for the DCT. (12 bit signed has 11 data bits. For an 8x8 DCT, the 11 data bits can be multiplied 8 times (represented by 3 bits). 11 bits multiplied by 3bits can result in 11+3 or 14 bits. Added to these 14 bits are the sign bit and the fraction bit to give a total of 16 bits). The algorithm used for the calculation of the 2D DCT is based on the following equation .

$$C_{k,l} = \sum_{n=0}^{M-1} \sum_{m=0}^{N-1} f(m,n) \frac{\cos\left(\frac{\pi(n+1/2)k}{2M}\right) \cos\left(\frac{\pi(m+1/2)l}{2N}\right)}{4}$$

DCT of the rows are calculated and then the 1D DCT of the columns are calculated. The 1D DCT coefficients for the rows and columns can be calculated by separating equation1 into the row part and the column part.

$$C_{k,l} = \sum_{n=0}^{M-1} f(m,n) \frac{\cos\left(\frac{\pi(n+1/2)k}{2M}\right)}{2} \frac{\cos\left(\frac{\pi(m+1/2)l}{2N}\right)}{2}$$

where $k = \sqrt{M}$ for row = 0, $\sqrt{2N}$ for row = 0
 $l = \sqrt{N}$ for col = 0, $\sqrt{2M}$ for col = 0
 $M = \text{total \# of columns}$, $N = \text{total \# of rows}$

The 16 bit signed input will provide 12 bit signed out pixels for the IDCT. (In the 16 bits, disregarding the sign bit, there are 15 bits of data including the fraction bit. Doing an IDCT divides these 15 bits of data by 8 (represented by 3 bits) resulting in 15-3 or 12 bits. Since the IDCT is a perfect inverse of DCT, if you start with integer you must receive an integer back. So from the 12 bits, the one fraction bit is ignored and the sign bit is added back to give a final 12-bit result).The algorithm used for the calculation of the 2D IDCT coefficients are based on the following equation. IDCT of the rows are calculated and then the 1D DCT/IDCT of the columns are calculated. The 1D DCT coefficients for the rows and columns can be calculated by separating equation1 into the row part and the column part.

VII. EXPERIMENTAL RESULTS AND COMPARISON

Depending upon the device used usage of number of elements and total number of elements will differ. Device family using here is Cyclone II.In existing system 2% of logic elements and combinational functions are utilized. 1% register utilization occurred.50% pins are used. There by area utilization can be predicted.

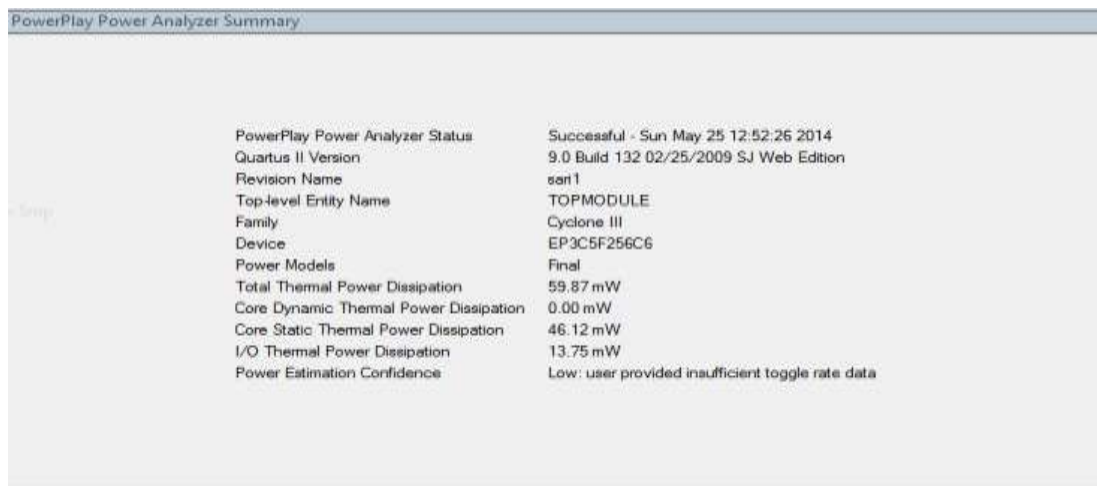
Flow Summary		
Flow Status	Successful - Sun May 25 12:57:26 2014	
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition	
Revision Name	sar111	
Top-level Entity Name	SEG_TOP	
Family	Cyclone III	
Met timing requirements	N/A	
Total logic elements	280 / 15,408 (2 %)	
Total combinational functions	278 / 15,408 (2 %)	
Dedicated logic registers	92 / 15,408 (< 1 %)	
Total registers	92	
Total pins	172 / 347 (50 %)	
Total virtual pins	0	
Total memory bits	0 / 516,096 (0 %)	
Embedded Multiplier 9-bit elements	0 / 112 (0 %)	
Total PLLs	0 / 4 (0 %)	
Device	EP3C16U484C6	
Timing Models	Final	

Amount of power consumed is the main parameter to analyze the performance of the circuit. Quartus is providing efficient tool, Power Play Power Analyzer which gives detailed information about power. Thermal power dissipated is 80.16mw.Static power dissipation and thermal Power dissipation is 47.39mw and 39.43mw respectively.

Flow Summary		
Flow Status	Successful - Sun May 25 12:48:53 2014	
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition	
Revision Name	sar1	
Top-level Entity Name	TOPMODULE	
Family	Cyclone III	
Met timing requirements	N/A	
Total logic elements	96 / 5,136 (2 %)	
Total combinational functions	96 / 5,136 (2 %)	
Dedicated logic registers	85 / 5,136 (2 %)	
Total registers	85	
Total pins	52 / 183 (28 %)	
Total virtual pins	0	
Total memory bits	0 / 423,936 (0 %)	
Embedded Multiplier 9-bit elements	0 / 46 (0 %)	
Total PLLs	0 / 2 (0 %)	
Device	EP3C5F256C6	
Timing Models	Final	

Data rate is calculated by using the Time Quest Analyzer and the report is obtained as shown .Clock frequency obtained is 204.88 MHz. In proposed system2% of logic elements and combinational functions are utilized. 1% register utilization occurred.50% pins are used. There by area utilization can be predicted. . Thermal power dissipated is 80.16mw.Static power dissipation and thermal Power dissipation is 47.39mw and 39.43mw respectively. Data rate is calculated by using

the Time Quest Analyzer and the report is obtained as shown .Clock frequency obtained is 204.88 MHz.



PowerPlay Power Analyzer Summary	
PowerPlay Power Analyzer Status	Successful - Sun May 25 12:52:26 2014
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	ser1
Top-level Entity Name	TOPMODULE
Family	Cyclone III
Device	EP3C5F256C6
Power Models	Final
Total Thermal Power Dissipation	59.87 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	46.12 mW
I/O Thermal Power Dissipation	13.75 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

VII. CONCLUSION

Both existing and proposed system are focusing on background identification using GMM. It differs on the resource utilization. GMM algorithm is a subtraction algorithm which compare each frame of video with reference frame. The comparison is mainly depending on the parameters like mean, variance and weight. For comparison and shifting, existing system is using separate comparator and shifter for each and every frame. So large area is occupied by existing system. But in Proposed system only one shifter and comparator is used for all frame. Since almost all the transmission system is sending compressed video, the original video can be captured only through decompression. So decompression technique is also used in proposed system. Video codec is the software which uses DCT and IDCT as an compression and decompression algorithm. In addition to the above processes, scaling technique is also implemented using the device called video scalar which do both up and down scaling. Though the proposed architecture in cooperate these three processes, the ultimate goal that is background identification is obtained with less power consumption.

REFERENCES

- [1] X. Benxian, L. Cheng, C. Hao, Y. Yanfeng, and C.Rongbao, "Moving object detection and recognition based on the frame difference algorithm and moment invariant features," in *Proc. 27th Control Conf.*, Jul. 2008, pp. 578–581.
- [2] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *Proc. IEEE Frame-Rate Appl. Workshop*, Sep. 1999, pp. 1–19.
- [3] R. Rodriguez-Gomez, E. J. Fernandez-Sanchez, J. Diaz, and E. Ros, "FPGA implementation for real-time background subtraction based on horprasert model," *Sensors*, vol. 12, no. 1, pp. 585–611, Jan. 2012.
- [4] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Real time moving object detection for video surveillance system in FPGA," in *Proc. Conf. Design Archit. Signal Image Process.*, Nov. 2011, pp. 1–8.
- [5] S. Y. Elhabian, K. M. El-Sayed, and H. A. Sumaya, "Moving object detection in spatial domain using background removal techniques state-of-art," in *Proc. Recent Patents Comput. Sci.*, vol. 1. Jan. 2008, pp. 32–34.
- [6] *OpenCV Library on Source Forge*. (2009) [Online]. Available: <http://sourceforge.net/projects/opencvlibrary/>