# An Optimal Design of Moving Objects Tracking Algorithm on FPGA

Lina Noaman Elkhatib, Fawnizu Azmadi Hussin, Likun Xia and Patrick Sebastian
*Centre for Intelligent Signal and Imaging Research*
*Electrical and Electronic Engineering Department, University Teknologi PETRONAS,*
Bandar Seri Iskandar, Perak, Malaysia
rds.r.mdbywalking@gmail.com,{fawnizu, likun_xia, patrick_sebastian}@petronas.com.my

*Abstract-* **This paper presents an optimal design and implementation of hardware system units for tracking moving objects on FPGA. Designs outline dataflow between different frequency domains using pipelining and FIFOs as buffers. The system implements a large amount of operations in order to apply a tracking algorithm called Adaptive Hybrid Difference. Two core units were built to implement the algorithm on FPGA, the adaptive threshold unit and the binary image builder unit. By implementing the proposed design with pipelining and parallelism, the number of clock cycles required to calculate the adaptive threshold over 30 frames has reduced from 23 × cycles to 4.3 × cycles. Additionally, optimal logic elements utilization has been achieved through the design.**

## I. INTRODUCTION

Implementation of image processing algorithms on field programmable gate array (FPGA) devices can achieve a system with better efficiency and dissipate less power and resources as compared to the use of software [a reference is required] implementation. Many hardware systems have been designed to handle video surveillance algorithms of parallel nature utilizing the parallelism capabilities of the FPGAs. On the other hand, object tracking and detection are the prime features of video surveillance systems from implementation perspective. Thus, many researchers in both embedded systems and surveillance applications areas have been interested in the design and implementation of object tracking and detection on embedded systems such as FPGA. However, an incorrect FPGA design may cost extra unnecessary gates as area and power which will slowdown the system. In addition, the building of large systems represents a big challenge requiring the adjustment of data flow between blocks of different frequencies. An additional important issue for tracking algorithms that must be met by design on FPGA is to achieve the real time conditions, which is estimated as 30 frames per second (fps) [1].

Segmentation is the common algorithm that has been used by many researches in the implementation of object tracking on FPGA. Meingast, et al. [2] have implemented their tracking system using image segmentation based on color threshold. The system works with 27 MHz frequency on image size of 768 $\times$288 pixels while the logic element (LE) used estimated by 798 LEs, which represent 10% of the device logic utility. It has achieved throughput of 25 fps.

Other works using segmentation algorithms and pattern matching through extracting spatial features of the detected objects were introduced by Yamaoka et al. [1, 3,] and Morimoto et al.[4]. The system works with 20 MHz frequency on image size of 80$\times$60 pixels using 31,987 LEs, which represent 56% of the device logic utilities. For one frame, it requires 13,745 cycles + 10 cycles $\times$ N to execute, where N is the number of detected objects [1,4].

Kazuhiro and Shinichi of Image Gravity Center and Matched Filter have implemented Planer Motion Tracking on FPGA [5]. The implemented algorithm has achieved a high performance of 1545.44 fps, while the logic elements used have estimated by 30,140 LE that represent 44% of the logic utilities. The system speed was set to 66 MHz, while the frame size used was 64$\times$64 pixel.

In a new research in 2010, Bravo et al. [6] implemented Principal Component Analysis algorithm for motion tracking. The main contribution of their work is to implement the complete PCA algorithm on FPGA in contrast to other works that use both FPGA and PCs in order to do that. The implemented system has achieved a throughput of 121 fps with 100 MHz frequency for 256$\times$256 pixels' frame size. The synthesized logic utility was estimated as 8,450 LEs that represents 86% of the device logic resources.

For FPGA devices manufactured using CMOS technology, the power consumption depends on the charging/discharging of the capacitance on gates and metal traces [7]. Thus the capacitance is affected directly by the number of toggled gates at any time and by the length of routs connecting the gates. The clock frequency of the system also is directly related to the frequency that is one of the power consumption factors [7].

The area is considered as one of the primary physical characteristics of a digital design [7]. Thus, area optimization of FPGA needs to be required to estimate the performance of the designed system as in [1].

Listed work in [1,3-6] have achieved high throughput. However, the logic utility in average was high, too, which increases the power consumption and the reserved area of the FPGA as well. Another tracking algorithm called Adaptive Hybrid Difference (AHD) by Shi et al. [8] for software implementation has not been implemented on FPGAs before. AHD is a background subtraction algorithm with improved adaptive background model. AHD works better than image difference algorithm and gives more accurate results than mixture of Gaussian models. In most of other tracking algorithms, operations of the frame is done repeatedly within the current frame itself, while in AHD, pixels' subtractions are done among number of frames and results are accumulated during processing. Therefore, the logic required by AHD is simple but the process of accumulating results through different time frames could potentially slow the system as a result of reading and writing data via storage devices. To overcome this bottleneck on speed while keeping the device utility minimum, we employed special techniques to deal with memory accesses and processing of frames. As a result, the proposed design optimized the logic utilization of FPGA.

In this paper, the design and implementation of Threshold Definer (TD) unit and Binary Image Builder (BIB) unit are explained in section V. The pipelining technique used to optimize the system is outlined through subsection B of section V. Techniques used to maximize the throughput and get the highest speed of the system are explained in subsections A and B of section V. Results obtained are discussed and conclusions are given in section VI.

## II. ESTIMATION OF THE ADAPTIVE THRESHOLD

The adaptive threshold (AT) as proposed in [8] is calculated by finding the means and standard deviations for locations of all pixels of the window within 30 frames as given by equations (1) and (2).

$$\mu'_{ij} = \frac{1}{N}\sum_{t=1}^{N} D^t(i,j) \tag{1}$$

$$\sigma'_{ij} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N}(D^t(i,j)-\mu'_{ij})^2} \tag{2}$$

where N is the number of frames used to find the threshold (N=30), t is the current frame number and $D^t(i,j)$ is the absolute value of the pixel difference between t-th and (t-k)-th frames as illustrated in equation (3).

$$D_k^t(i,j) = \left|G^t(i,j) - G^{t-k}(i,j)\right| \tag{3}$$

$G^t(i,j)$ is the pixel value in position (i,j) in the t^th frame. The value k is used to control the quality of tracking and it is determined experimentally according to test location conditions such as lighting, environment, speed and the type of moving objects [8].

Therefore, lower and upper limits of the adaptive threshold are (4).

$$T^{ij} = [\mu'_{ij} - \sigma'_{ij}, \mu'_{ij} + \sigma'_{ij}]. \tag{4}$$

## III. BUILDING THE BINARY IMAGE

Threshold is calculated at initial time to be used later to build the binary image. The binary image is a version of the current frame using only two values for pixel representation; zero and one. At tracking time a black color is given to the still pixels while a white color is given to the moving ones, obtaining the binary image. A binary image can be built by using (5).

$$B^t(i,j) = \begin{cases} 1 & if\,(\bigvee_{s=1}^{k} D_s^t(i,j) \notin T^{ij}) \\ 0 & if\,(\bigvee_{s=1}^{k} D_s^t(i,j) \in T^{ij}) \end{cases} \tag{5}$$

where $i=1,\dots,n,\quad j=1,\dots,m,\quad t=1,2,\dots$ .

## IV. SYSTEM DESIGN

The deployment model of the tracking system, which illustrates the data flow of the input and output through the system, is shown by Figure 1.

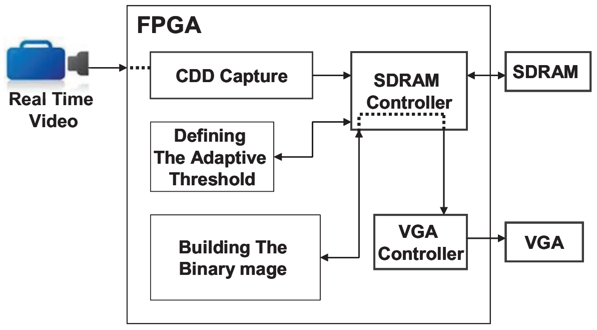

Figure 1. Deployment Model

Data are captured by the camera through the Charged Coupled Device (CCD) image sensor. The captured data are received by the FPGA via the “CCD capture” block where it is converted to the appropriate digital form for image processing. The formatted data is sent continuously to the external SDRAM memory through “SDRAM controller” block. Later on, after storing the required number of frames, the frames start to transfer to “Define Adaptive Threshold” block to calculate the threshold. When the threshold had been calculated, “Building Binary Image” block starts processing by sending the data to it, where the moving objects are detected and given a white color and the rest of the image is set to black. Read/write operations are interacting between blocks and the SDRAM while processing the threshold and building the binary image. The final output is a binary image of the moving objects over a frame. To observe the output of the binary image, it could be sent to VGA monitor through “VGA Controller” block that synchronizes the output frames.

The proposed system requires an 8 Mbytes SDRAM memory, 640 × 480 pixels of the window size, and 16 bits for the width of a pixel which is the size of the data row. To store 30 frames in the memory as the algorithm requires, 18.432 Mbytes is needed. Obviously, the required space is larger than the SDRAM space of 8 Mbytes. Thus only 6 frames at a time can be stored in the memory in addition to the two lookup tables used to accumulate thresholds over 30 frames. Values stored in LUTs are used later to build the binary image while new six frames are stored gradually after discarding the processed frames.

The core units of the system are the Threshold Definer and the Binary Image Builder units. The first calculates the threshold while the other builds the binary image of detected objects. Next sections show the design and architecture of those units in order to achieve the required throughput with optimal device utilization.

## V. PROPOSED DESIGN AND ARCHITECTURE FOR FPGA IMPLEMENTATION

The core of the system is represented by the processing units that generate the threshold and the binary image. Threshold Definer and Binary Image Builder units are not working concurrently. TD works first to determine the threshold and saves it in a specific location in memory. After TD units calculate the threshold, a control signal is generated and given to the BIB unit to start functioning. BIB starts constructing the binary image based on the current frame and the pre-calculated threshold. Control signals are generated by a finite state machine and are used to arrange the data path between the two units.

Hardware Architecture of units is illustrated in Figure 2. In addition, this figure displays the important connections between FIFOs, SDRAM and the designed units.

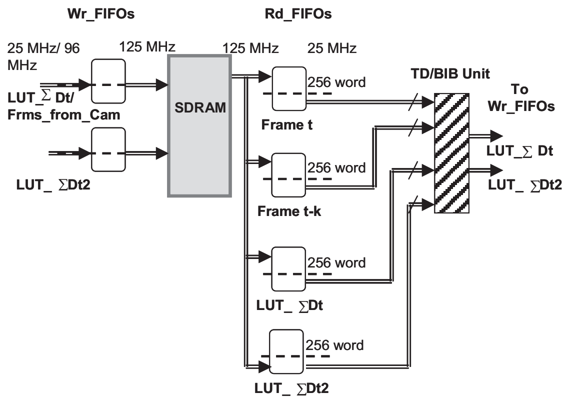

Figure 2. Architecture showing interactions between FIFOs SDRAM and processing unit.

TD and BIB are simple units built of add, subtract, comparison, AND and OR operations as shown in Figures 3 and 4.

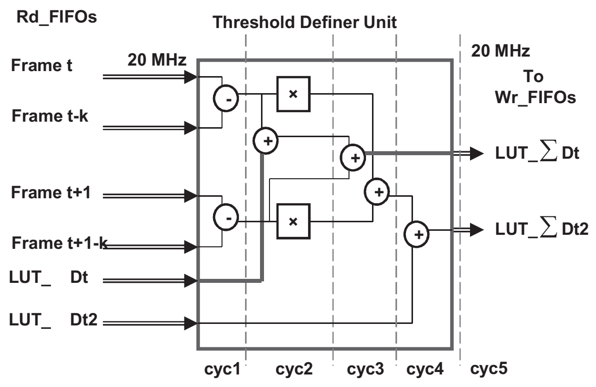

Figure 3. Threshold Definer unit's Architecture

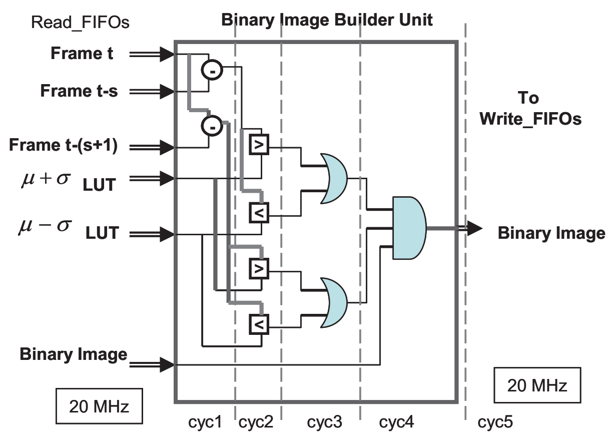

Figure 4. Binary Image Builder unit's Architecture

As in Figure 2, the SDRAM is working with a higher frequency than that of the designed unit. FIFOs are used to overcome the frequency differences between the SDRAM and the units' blocks of the design to avoid clock domain crossing (CDC) problems. Asynchronous dual-clock FIFOs are used where read/write operations can be done with different clock domains. Four FIFOs are used to read data from four different locations of the memory with clock speed of 125 MHz. Every location represents a different stored frame or a lookup table. From the other side, data are moved from FIFOs into the specified processing unit with a clock speed of 25 MHz. The speed of the system has been set to assure an automatic synchronization for data inputs. All four inputs are supposed to be ready at the same time when the positive edge of the unit's clock is reached. With the same frequency, the obtained results are synchronized to Write_FIFOs since they stay there until they are stored into the memory.

To save SDRAM's access time, a burst of 256 words has been set. Since the SDRAM has only single port with 16 bits width, it requires $4 \times 256 \times (1/125M) = 8{,}192$ ns to fill all Read_FIFOs. Since the unit has four 16-bit parallel data ports, it needs 256 cycles to buffer in that data. Thus, with 25 MHz clock, the time required is $256 \times (1/25M) = 10{,}240$ ns. In addition the size of FIFOs is 512 words, which allows the memory to load the next 256 words during the data processing of the first 256 words. The designed units will not work faster than the SDRAM, and the FIFOs will not be empty or full while processing a frame since:

- The required time to buffer any amount of data from the SDRAM into FIFOs is less than the time to take data from the FIFOs into the designed units as just calculated above.
- The size of FIFOs is chosen to adapt two memory burst (512 word), while the units are reading the first burst the SDRAM is buffering the next burst of data into the empty burst of FIFOs.
- The SDRAM has only a single port while the designed units have four parallel ports for input.
- The overlapped timing which is 10,240 – 8,192 = 2,048 ns, is enough to read the new 51 words

from the memory into the FIFOs. At that time the units are reading the last 51 words of the old read burst. Thus the overlapping of timing will not cause a problem since there will always be empty spaces for the new data to be written into FIFOs.

### *A. Throughput Improvement*

With the proposed design of Figure 2, the throughput of the threshold unit is 25 M / 640 × 480 × 30 = 2.7 fps and for the Binary Image Builder unit is 25 M / 640 × 480 × 6 = 13.56 fps. In order to achieve real time conditions, the speed has to be at least 30 fps [1]. Processing speed for the threshold definer unit does not affect the system since it is executed at the initial time before tracking starts. In addition, if the threshold is re-configured to adapt to the changes of the background; it could be re-executed at every specific periods of time, not for every new frame. Thus, throughput concerns must be paid only for the BIB unit since it is the one that works at tracking time. To improve the throughput, parallelization is used. By reducing the system frequency to 20 MHz and adding two Read_FIFOs, the throughput is increased to 21.7 fps. As shown in Figure 4, the additional two FIFOs are used to calculate pixel difference between two other frames in parallel with the first two FIFOs used for frame t and frame t-k. The additional FIFOs are used to find differences between the current frame t and one of the six preceding frames scheduled to be subtracted from the current frame to check that all results are belonging to the threshold domain.

### *B. Pipelining Datapath*

To read a new data row each clock cycle according to the design explained in section V without data overlapping, one of two things must be achieved through the designed units. Either to have inner buffer inside the unit to hold the data until it has been processed while receiving new data for the next clock cycle or to process incoming data within one clock cycle.

Increasing the storage units used among the FPGA by using inner buffers does not achieve the objectives of this research. In order to reduce the required clock cycles of processing data through the designed units, pipelining techniques have been used. TD unit must read from six FIFOs, find two hybrid difference values, find the power of them then accumulate the new results with the old results of lookup tables. All the operations mentioned should be executed within one cycle. By re-building the TD unit using pipelining in a way that assures that the critical path of any pipelined stage is taking only one cycle, the processing latency of the unit is reduced to one cycle. Figure 5 shows the pipelining data path that has been designed to achieve TD's operations within one clock cycle.

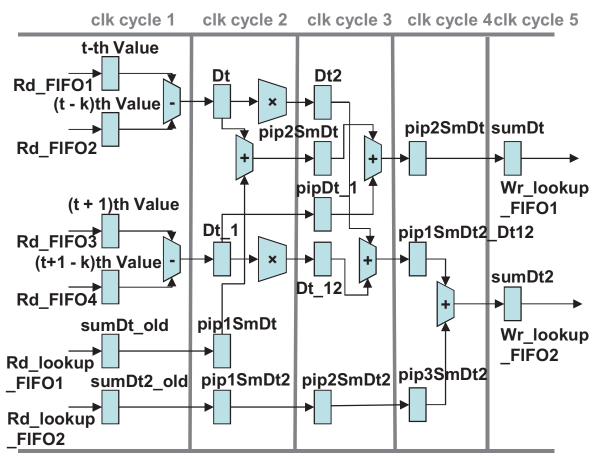

Figure 5. Pipelining data path diagram for TD unit.

The data path depth is 5, thus it costs 5 clock cycles of delay at the beginning of processing a new frame. Therefore, the overall latency to find threshold over 30 frames is 5 × 30 = 150 clock cycles. However, pipelining has optimized the required cycles from 23 M cycles to 4.6 M cycles, which is significant for synchronous behavioral modeling.

If TD and BIB units were designed using combinational logic, then the time can not be estimated through the number of cycles. The time of non-pipelined combinational logic is defined by the critical path among the whole logic. However, in a pipelined design, the time is determined by the critical path among the pipelining stages not over the whole unit. We distribute operations on pipelining stages in such a way that it takes a clock cycle for each stage to be implemented. Thus, the critical path of one stage of pipelining is shorter than the one generated by the non-pipelined combinational logic implementation.

## VI. RESULTS AND DISCUSSION

The system has been implemented on Altera Cyclone II device. From 33,216 logic elements (LE) of the Cyclone device, 151 LEs (<1%) have been used. Synthesis Utility of the designed system is illustrated in Table 1. By comparing the logic utilization used by [1, 3-6] since their designs used from 798 to 30,140 LEs, the proposed design costs optimal number of logic elements and thus optimal number of gates and device area. Choosing the AHD algorithm that does not require complex operations while reducing timing through pipelining, has achieved a design with optimal number of logic gates and meets the required speed. Using FIFOs has allowed us to adjust the different read/write speeds between the SDRAM and the designed units.

TABLE 1
FPGA'S SYNTHESIS UTILITY OF OUR SYSTEM

| Total Logic Elements | | LUT | Multipliers |
|---|---|---|---|
| 151/33,216 (<1%) – 145 for combinational function | | 56 | 0/70 |
| Registers | Internal Memory Bits | PLL | Memory Blocks |
| 95 | 8,192/483,840 (2%) | 2 | 1/4 (25%) |

Using parallel units with the pipelining has reduced the required number of clock cycles needed to find the threshold and building the binary image. The number of clock cycles was reduced from 23×10 ^ 6 cycles to 4.3 ×10 ^ 6 cycles for the TD unit. Throughput was calculated using the equation

$T = S \times F/C$

where S is the data row size, F is the frequency and C is the required number of cycles to implement the algorithm [4]. Thus, the throughput of the system at tracking time is 20M / (307,200 × 3) = 21.7 fps, with 20 MHz system speed.

## VII. CONCLUSION

This work has presented architecture of hardware design for object tracking units implemented on FPGA. A novel algorithm is implemented based on AHD that on FPGA. An optimal resources of 151 logic elements and 2 internal block RAMs have been used. The system is evaluated in terms of the speed and the cost on hardware resources. It achieved throughput of 21.7 fps with a low frequency of 20 MHz. Less than 1% of device's area and cost are needed by the proposed design. The real time frame rate for our system can be achieved by reducing the frame size used for the captured video. Thus, it may improve moving object tracking system using optimal resources, area and power consumption and meeting the real time conditions at the same time.

## ACKNOWLEDGMENT

The Authors would like to acknowledge the support of the Electrical and Electronic Engineering Department of Universiti Teknologi PETRONAS (UTP) during this study.

## REFERENCES

[1] K. Yamaoka, et al., "Image segmentation and pattern matching based FPGA/ASIC implementation architecture of real-time object tracking," in Design Automation, 2006. Asia and South Pacific Conference on, 2006, p. 6 pp.

[2] M. Meingast*, et al.*, "Automatic Camera Network Localization using Object Image Tracks," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1-8.

[3] K. Yamaoka*, et al.*, "Multi-object tracking VLSI architecture using image-scan based region growing and feature matching," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006, p. 4 pp.

[4] T. Morimoto*, et al.*, "An FPGA-Based Region-Growing Video Segmentation System with Boundary-Scan-Only LSI Architecture," in *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, 2006, pp. 944-947.

[5] S. Kazuhiro and H. Shinichi, "Implementing Planar Motion Tracking Algorithms on CMOS+FPGA Vision System," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 1366-1371.

[6] I. Bravo*, et al.*, "An Intelligent Architecture Based on Field Programmable Gate Arrays Designed to Detect Moving Objects by Using Principal Component Analysis," *Sensors 2010,* pp. 9232-9251, 2010.

[7] S. Kilts, *Advanced FPGA design: architecture, implementation, and optimization*: Wiley, 2007.

[8] S.-x. Shi*, et al.*, "A Fast Algorithm for Real-time Video Tracking," in *Intelligent Information Technology Application, Workshop on*, 2007, pp. 120-124.