An FPGA based Hardware Accelerator for Real Time Video Segmentation System

Indra Yasri
Electrical Department, Engineering Faculty, Riau University

E-mail: indyasri@yahoo.com

Abstract—Video Segmentation is the basic requirement for applications such as video surveillance, traffic management and medical imaging. The high computation power must be provided to support this operation. This paper discusses the design and implementation of hardware accelerators for video segmentation. The algorithm of Sobel edge detection operator is used to develop this hardware accelerator. To develop hardware accelerator datapath architecture the management of memory access is deployed and architecture based pipeline are made with the potential improvements in acceleration to the read data pixel from memory. In addition, a finite state machine is used to ensure the hardware accelerator controls the sequence of derivative computation, the write and read operations. System Integration uses NIOS II processor and Avalon bus interfacing. The hardware accelerator design was implemented on an Altera FPGA development board and has managed to achieve a video rate of 30 frames per second as required by NTSC standard definition video. The architecture core for memory access has succeeded to reduce memory bandwidth utility by 75%.

I. INTRODUCTION

Video Surveillance technology plays an important role nowadays. Many strategic places deploy security camera surveillance. The camera surveillance assists security guard to do monitoring for wide area. The large amount of installed camera and a large amount of captured image make extremely tedious to observe the video feeds for long periods of time.

The video must present ‘live show’ as what have been captured by surveillance cameras. The ‘live show’ means real time in computer engineering. With this real time performance make security guard able to give fast respond to the critical situation. In order to differentiate between critical situation and ordinary event which are showed by video surveillance the video image processing feature need to embedded. This feature reduces attention to ordinary events which are not security relevant on the video feeds. Many feature of video image processing are available.

The most common embedded video image processing feature in video surveillance is video segmentation and object recognition. One of the key algorithms used in video segmentation and object recognition is edge detection. Edge detection is operated by mathematic algorithm. Early edge detection methods used local operators to approximately compute the first derivative of grey level gradient of an image in the spatial domain. The position of local maximum of the first derivative is considered to be the edge point. Examples of gradient-based edge detectors are Prewit and Sobel operators [1, 2]. The Laplacian of Gaussian (LOG) operator for edge detection has been proposed by Marr and Hildreth [3] which uses a Gaussian function for image smoothing, then computes the second derivative and the edge points are displayed using zero crossing points.

With the advance of computation speed of application specific integrated circuit (ASIC) and general purpose microprocessor has lead to research study on the optimization of hardware architecture for faster speeds. This paper presents an edge detection hardware accelerator to expedite the edge detection computation using the technique of memory bandwidth reduction for embedded video segmentation. The increase of computation speed is significant compared to preliminary work [5].

II. HARDWARE ACCELERATOR ARCHITECTURE

Real time image processing for NTSC standard requires high computation power. The high computation power is required to support 30 frames per second processing with 720x480 pixels per frame.

In the case of the Sobel edge detection algorithm the processing requirement is to calculate the derivative pixels from the original image pixels. The procedures of calculation commence with the reading of the original image from the memory and loaded into a transit register. The next step is to perform the computation process of derivative image. Finally the
derivative result is written back to memory. To perform the algorithm with resolution 30 frames per second as required for real time processing, a dedicated hardware architecture with memory bandwidth reduction is implemented to deal with computation capacity and memory bandwidth constraint. The memory bandwidth reduction is achieved by parallelism in reading the original image pixel data from the memory and pipelining in processing of derivative computation as refer to [8]. This method manages to achieve 75% memory bandwidth reduction compared to the preliminary architecture [5]. The original and derivative images are stored in the memory with 32 bits wide and image data has an individual address. The pixel frames are stored as 1 byte per pixel. A row of pixels in a frame is stored from left to right at the respective address and a group of arrows stored from top to bottom, row by row. Read or write operation process to memory takes 18.5ns which is triggered by two cycles of a 110MHz clock operation. As discussed earlier 720x480x30 pixels arrive from the camera at approximately 14 million pixels per second or one pixel per 72ns. This means each operation writing or reading a single pixel to the memory will take approximately 20% of memory bandwidth. Aggregation of four pixels in writing to the memory with single access will save the memory bandwidth 5% instead of writing each pixel individually.

Another process which has the potential to reduce the memory bandwidth is derivative computation process. This process is accomplished by comparing the centre pixel with other pixels in 3x3 gradient Sobel operator kernels. As the centre pixel proceeds for derivative computation, simultaneously it accesses eight pixel values from the original image. If this computation processes normally it requires 20% multiplied by 8 pixels i.e. totally memory bandwidth required is 160%. This is unfeasible. In order to eliminate the memory bandwidth the computation deploys a pipeline together with parallelism on reading the original image pixel from memory. Since the reading of the original image pixel from memory is achieved by aggregation of 4 pixels then perform 3 parallel with additional 2 adjacent of 4 pixels. Using this method, the reading needs 3 pixels for every fourth pixel being computed and the left hand pixel will be stored within the accelerator to compute multiple derivative pixels. The memory bandwidth is therefore reduced to 15%. Once the derivative computation is complete, the result is written to the memory and another 5% bandwidth is utilized. The total memory bandwidth required for all process computation is 25%. The hardware architecture for Sobel accelerator is shown in Fig.1. The architecture design shows the interaction between block accelerator and memory. These 2 blocks are assisted by other blocks which can achieve memory bandwidth reduction. The architecture consists of 5 inputs and 2 outputs as shown in Table 1 below:

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Bits</th>
<th>Input/Output</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>1</td>
<td>Input</td>
<td>Clock=110MHz</td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>Input</td>
<td>Reset=1, Idle</td>
</tr>
<tr>
<td>Start</td>
<td>1</td>
<td>Input</td>
<td>Start=1, Read</td>
</tr>
<tr>
<td>Address</td>
<td>5</td>
<td>Input</td>
<td>Input address</td>
</tr>
<tr>
<td>Register</td>
<td>32</td>
<td>Input</td>
<td>In memory unit</td>
</tr>
<tr>
<td>Data In</td>
<td>32</td>
<td>Input</td>
<td>Parallel 32 bit</td>
</tr>
<tr>
<td>Data Out</td>
<td>32</td>
<td>Output</td>
<td>Parallel 32 bit</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>1</td>
<td>Output</td>
<td>Ack=1, No Error</td>
</tr>
<tr>
<td>Flag</td>
<td>1</td>
<td>Output</td>
<td>Ack=0, Error</td>
</tr>
</tbody>
</table>

There are 8 blocks functioning inside this hardware accelerator architecture, such as a clock divider controller which converts block from 110MHz clock input into 54 MHz with 18.5ns pulse width. This pulse is needed to support the operational of the accelerator control block and control address block. The second block is the accelerator control. This block performs Sobel edge detection computation process and also acts as a controller for the flow of image data from and to the memory unit. The output of the accelerator control block will utilize the input of the following blocks: address counter, memory access controller, and acknowledge generator. The next block is the address counter. The address location for the original image data and derivative image data is assigned by the address counter block. The assignment of the specific address is performed by increment on the base address for the respective image data pixel. The other block is the memory access controller to control the changing format of data when accessing from memory, from parallel to serial or vice versa. It is also in charge of memory control access register block to receive data from memory or to transmit the data to memory. One more block is memory access register also connected to the supporting function for memory access. The data is temporarily saved in this block before being sent to or received from memory. Once the packet of data is received or sent completely this block will trigger acknowledge generator to produce an acknowledge signal. Another block is the memory unit where data pixel, derivative address, 1st row rawpix address, 2nd rawpix address, and 3rd rawpix address are stored. The address decoder block determines the address to be accessed in the memory unit. The last block is acknowledge generator block.
where once the data has been completely read or written, the acknowledge generator block will produce an acknowledge signal.

III. SYSTEM DESIGN INTEGRATION

FPGA-based DSP system design mostly combination of system development tools and hardware description languages (HDL) tools. And sometime in order to accelerate the design process, it uses the third-party intellectual property (IP).

The implementation for the design in the previous section is performed on Stratix III Altera DSP Development board. In order to make complete function of video segmentation system, integration with other supporting module is needed. Model based design tools such as MATLAB Simulink with Altera’s DSP builder libraries is assist in development of video segmentation function. The HDL for Sobel edge detection hardware accelerator can be integrated with other hardware design within DSP builder libraries. The integrated Sobel edge detection hardware accelerator is operated by NIOS II processor. The operation of this design is to check whether the design able to accommodate the function of video segmentation. The process of design integration into NIOS II processor including bus peripheral interface using System-On-a Programmable-Chip (SOPC). DSP Builder and SOPC are affiliated under Quartus II development environment.

SOPC will arrange the connection for all of involved components. The connection is achieved with Avalon bus interfacing. Avalon bus interfacing is a standard bus interfacing for NIOS II processor.

After SOPC complete the integration process, the NIOS II processor ready to program. The programming of NIOS II is performed by ANSI C or assembly language. Once the NIOS II processor is programmed, it will control a whole of system.

IV. INTEGRATION MODEL AND IMPLEMENTATION

The model of integration will involve such as component, input buffer DDR2, output SRAM buffer, VGA controller, direct memory access (DMA), NIOS II processor and Sobel edge detection hardware accelerator as the main block of design. Moreover, the Sobel edge detection hardware accelerator is a co
processor and also serves to accelerate access to the memory.

In the design of hardware accelerator is necessary to accommodate ports for interfacing purposes because the traffic of data will flow through this ports. The hardware accelerator datapath controls the flow of data. There are 3 main activities involved inside hardware accelerator datapath e.g. read data in memory, derivative computation and write derivative result into memory. Read and write activities are performed by direct memory access (DMA). DMA read pixel data from the memory and put into 3 registers parallelly. It is 32-bit registers that accommodate 4 pixels with 8bits each of them. These registers also parallelly supply pixel by pixel into derivative computation arrays that executed by pipelining. The derivative computation result will move into a 32-bit result register that locate 4 pixels with 8bits each of pixels. Once this register loaded by 4 pixels the DMA will write the data pixel subsequently into memory.

All activities during processing of pixel data from memory into registers, registers into derivative computation then shift the result into result register and write it back into memory are required synchronization. The synchronization is needed to make sure the pixel data proceed in the proper timing according to the format of data. The synchronization will controlled by finite state machine as shown in Fig.2.

The finite state machine consists of 11 states. It is begun with idle state where initiated by reset. The start state is activated by set run value to 1. The following states are read 1st row rawpix, read 2nd row rawpix and read 3rd row rawpix. These states are activated by read=01, read=10 and read=11, respectively. These three states are representative of 1st row data image pixels, 2nd row data image pixels and 3rd row data image pixels that read parallel from memory. The acknowledge state will activated when each of these states is reached 720 data pixels. The next state is 1st group computation that activated by ack=10. This state will complete computation for 180 pixels in order to activate 2nd group computation. The write derivative state will activated by ack=11. This state will complete writing for 480 rows in order to comeback to acknowledge state. And finally acknowledge state move to idle state with the condition of the ack=01.

When direct memory access (DMA) perform read and write operation in memory, it requires address generator. The address generator will perform counting from the base address for reading of pixel data and as well for writing the derivative result. The inputs such as data pixel, offset counter enable signal, enable signal for both read operation and write operation are required to develop datapath for address generator. The datapath determine pixel addresses using two base address register: pixel read register and derivative write register. The main memory capacity is 8 Mbytes which organized as 1M x 32bits. The organization of memory is equal to 220 or 20bits. The pixel data input and the output of address selector is implementing in 20bits size. Pixel offsets is counting from the base addresses which starts from 0 and increment by 1 when group of 4 pixels data read from memory. Then add the offset to the base address for 1st read register address. The 2nd read register address is reached by add 720/4 to 1st read register address. Then lastly, the 3rd read register address is achieved by add 1440/4 to 1st read register address. The counting for derivative result address is start from 720/4 offset value and increased by 1 for each memory write. Address selector will select the appropriate calculated address to produce the memory address.

The address generator datapath has 4 totally of registers: pixel read register, derivative write register, offset pixel read register and offset derivative write register. The pixel read register and derivative write register represent for base address of pixel image data and derivative image data, accordingly. The offset pixel read register and the offset derivative write register represent for the counting of pixel groups read and derivative result write, accordingly. Together all these 4 registers are controlled by signal that generated from hardware accelerator control design block. The combinational blocks are deployed to add four address signals belong to 1st read register, 2nd read register, 3rd read register and write derivative. These 4 combinational outputs will reach address selector. The address selector represented by multiplexer to select one of 4 address signals. The detail of the datapath is shown at Fig.3.

Fig. 2. Finite state machine for synchronization
The remaining integration aspect of hardware accelerator system is Avalon bus interfacing signals. All the signals correspond to the input and output of hardware accelerator. The signals will be assigned when perform integration process with SOPC builder.

V. RESULT AND DISCUSSION

The FPGA based hardware accelerator implementation was verified by Modelsim software. The verification result is displayed in timing diagram as shown at Fig.4.

The ordering of states is correct according to finite state machine (FSM). The FSM is start by set run=1 and come to load=00 for reading first row of pixel data. Next is load=01 for reading second row of pixel. And 3rd row of pixel is read after set load=10.

In the first round trial, the embedded hardware accelerator edge detection is managed to implement using of SOPC builder. The Nios II processor controls a whole of the embedded system through the software. The software is written in assembly language.

Components such as on-chip memory for Avalon memory-map (MM) slave, interval timer, DMA controller, parallel input-output (PIO), JTAG UART, system ID peripheral, etc., were add during SOPC process integration. There some justification when add those components i.e. system I.D allows Nios II development tools to validate the built software application match with the right hardware system. The connection of those components is performed by means of the interconnection network which is known as Avalon switch fabric.

Base on the implementation, the performance of hardware accelerator in edge detection application shows able to reduce memory bandwidth access compare to the referenced architecture [5] as shown at Table II below:

![Fig. 3. Memory Address Generator](image-url)

![Fig. 4. Timing diagram of states verification](image-url)
<table>
<thead>
<tr>
<th>Memory access</th>
<th>The proposed Hardware accelerator design</th>
<th>The referenced architecture [5]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel read</td>
<td>5ns (25%)</td>
<td>20ns</td>
</tr>
<tr>
<td>Pixel write</td>
<td>5ns (25%)</td>
<td>20ns</td>
</tr>
<tr>
<td>Pixel read for</td>
<td>15ns (30%)</td>
<td>50ns</td>
</tr>
<tr>
<td>derivative</td>
<td>computation</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>25ns (~75%)</td>
<td>90ns</td>
</tr>
</tbody>
</table>

The memory bandwidth for such tasks: pixel read, pixel write and pixel read for derivative computation are reduced 25%, 25% and 30%, respectively. This performance is achieved with the modification on referenced [5] architecture. The modification is performed on data pixel read, to include parallelism on the reading data pixel from memory. In writing the derivative result into memory the same method is deployed. In order to accelerate the derivative computation the pipeline is implemented. There are 3 stages of pipeline and work along with parallel’s 3 of reading register to make 3x3 arrays derivative computations. Therefore, 75% in totally bandwidth reduction is reduced by this modification.

VI. CONCLUSION AND FUTURE WORK

A real time embedded hardware accelerator edge detection design with improved memory access was implemented on an FPGA Stratix III DSP FPGA board. The improved memory access has shown it significantly assisted real time performance in video segmentation. Its application is recommended for a video surveillance system and it represents a much needed enhancement of current technology.

In the near future the completed design integration will be presented and able to demonstrate the function of automated edge detection which is the basic of video segmentation application in video surveillance system.

REFERENCES