High Speed Video Processing Using Fine-Grained Processing on FPGA Platform

Zhi Ping Ang, Akash Kumar, Yajun Ha
Department of Electrical & Computer Engineering
National University of Singapore
4 Engineering Drive 3, Singapore 117583
Email: {angzhiping,akash,elehy}@nus.edu.sg

***Abstract*—This summary paper[1] proposes an FPGA-based array processor which performs Laplacian filtering on a 40 by 40 pixel grayscale video. The architecture comprises of bit-serial pixel processors interconnected to give a two-dimensional mesh array. This architecture features the novel use of partial reconfiguration which transfers data to and fro the array. Each processor occupies a configurable logic block and achieves a target frame rate of 10000 frames per second, at an operating frequency of 0.31 MHz on the Virtex-6 ML605 Evaluation Kit. The detailed correspondence between the contents of slice lookup tables and the Virtex-6 bitstream format is also documented.**

***Index Terms*—Fine-grained FPGA computing; High speed video processing; Partial reconfiguration; Bit-serial arithmetic**

I. Introduction

Several scientific and engineering fields use high speed video capture to investigate physical phenomena that are too rapid for human perception. Examples include examining the bio-mechanics of a hummingbird's in flight [1], and investigating a projectile's impact pattern in criminal forensics [2]. Computational processing often accompanies video capture to analyse the video.

High frame rate videos require processing power that matches the high data throughput. Otherwise, a mismatch in data rate can be accommodated using off-line processing. However, real-time processing is desirable as the system can analyse and make situational decisions based on an ongoing event. Another downside is that off-line processing requires storage to buffer the incoming data, which is not required by real-time processing. An alternate way of matching data throughput to limited processing capability is to drop $n-1$ frames for every frame processed. A reduced frame rate of processing loses accuracy as discarded data can be used to refine upon the collected data.

This research proposes a single-chip reconfigurable hardware architecture which eliminates the use of off-line processing and frame dropping by accelerating video processing using an array of pixel processors. Each processor operates on a pixel, providing a computational speed-up proportional to the input frame size. This array processor also features the novel use of partial reconfiguration to distribute data to all the processors.

[1]An expanded version of this paper is accessible at http://x.co/fccm13p64.

II. Hardware-based High Speed Video Processing

To address the disadvantages of off-line processing and frame dropping, dedicated hardware is used to attain high frame rate video processing. They are categorised into three classes.

A. Commercial Video Processors

Processing capability is available in most commercial image sensors. Unfortunately, they are largely primitive forms of processing. Most image sensors output in either RGB or YC_bC_r colour space format. Colour space conversion trivially transforms the image at the pixel level, therefore no useful higher level information such as edge features can be obtained. Another form of processing would be amplification, which uniformly scales the magnitude of every pixel. Likewise, this operation occurs at the pixel level and does not perform any higher-level analysis.

B. ASIC-based Video Processing in Research

Image sensors with built-in high speed processing capabilities are more advanced within the research community as compared to commercial sensors. This section explores some of the cutting edge technology realised on application-specific integrated circuits (ASIC):

1) A Programmable Vision Chip Based on Multiple Levels of Parallel Processors [3]: Zhang et al. developed a vision chip which performs edge detection on an input video of 128 by 128 pixel resolution at a rate of 500000 fps. The chip devotes a processing element for every pixel, therefore, the speed up achieved is substantial.

2) Switched Current Analogue Matrix Processor (SCAMP-3) [4]: The SCAMP-3 chip performs Sobel filtering on an input video of 128 by 128 pixels at a frame rate of 3600 fps. Similar to the chip mentioned in Section II-B1 speed up is achieved by devoting dedicated hardware to every image pixel.

3) A Real-Time Motion-Feature-Extraction Image Processor Employing Digital-Pixel-Sensor-Based Parallel Architecture [5]: The chip designed by Zhu and Shibata is fabricated on the 65 nm process. It features a 100 by 100 pixel sensor integrated with a row parallel processing unit. As this chip does row parallel as compared to pixel parallel processing in the previous 2 examples, the effective processing frame rate is on the order of a few hundred fps.

C. FPGA-based Video Processing in Research

Although ASIC-based chips achieve excellent frame rate processing, design and fabrication are tedious and expensive. The design turnaround time for ASIC-based designs can take several months. Moreover, fabricating ASICs is not cost-effective unless they are manufactured in high volumes (i.e. millions of units per fabrication run). Therefore, a more flexible and cost effective platform such as the field programmable gate arrays (FPGA) is preferred for low to middle volume usage. The following discusses cutting edge developments of high speed video processing on FPGA.

1) 2000 fps Real-time Vision System with High-frame-rate Video Recording [6]: The paper demonstrates a video capturing and centroid computation onto a dual-FPGA system. The first chip performs camera input processing, noise reduction and interfaces with a workstation; the second chip is responsible for video processing. The input video has a resolution of 512 by 512 pixels and processes at an effective frame rate of 2000 fps.

2) Development of High-speed and Real-time Vision Platform, H3 Vision [7]: In this research the dual-FPGA setup is similar to that of [6], except that the system performs optical flow computation on a 1024 by 1024 pixel input image at a frame rate of 1000 fps.

In both research, the downside is that two FPGA chips are required to achieve a high processing frame rate. It is preferable for a video capture system to be implemented on a single chip solution as a larger chip count translates to higher material costs. Moreover, a multi-chip solution would mean higher developmental effort and a larger power expenditure by the system compared to single-chip.

III. Proposed Architecture

Addressing the disadvantages of offline processing or frame dropping requires hardware processing. The inflexible and costly ASICs give FPGA-based solutions an upper hand in terms of implementation flexibility and cost effectiveness. However, the current state-of-the-art research in high speed imaging on FPGA is found to be unsatisfactory in terms of the use of multiple chips to implement a capture-and-process system. Therefore, this research paper proposes a single-chip FPGA solution which performs high speed video processing.

A. Specifications

The target frame rate is at least 10000 fps. The input video is grayscale with a bit-depth of 8 and has a resolution of 40 by 40 pixels. The Laplacian operator, widely used in applications such as artifact rejection [8] and scene classification [9], is realised and given by (1). The hardware architecture is a two dimensional mesh array consisting of interconnecting primitive pixel processors, whereby each processor processes a single pixel. A processor assigned to every pixel ensures pixel-level parallelism. The array is implemented on the Xilinx ML605 Evaluation Board.

$$\nabla^2 I_{x,y} = I_{x,y} - \frac{1}{4}\left(I_{x-1,y} + I_{x+1,y} + I_{x,y-1} + I_{x,y+1}\right) \quad (1)$$

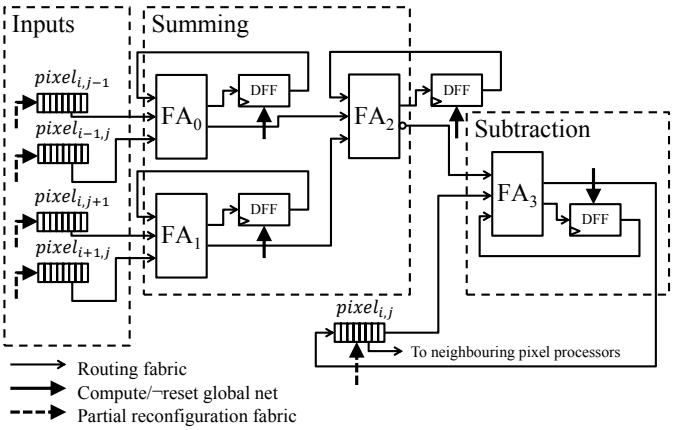

Fig. 1. Bit-serial architecture of a pixel processor

B. Bit-serial Arithmetic

On each pixel processor, the Laplacian operator is implemented using bit-serial arithmetic [10] with the architecture shown in Fig. 1. Pixel values reside in shift registers implemented using lookup tables (LUT). The registers shift out pixel values, least significant bit-first, into the bit-serial circuitry before returning to populate the result back. Bit-serial arithmetic can also be applied to various image filter kernels, for example, [11] implements the Sobel and Hough operator on a similar architecture.

C. Partial Reconfiguration for Pixel Data Distribution

Distributing pixel data using a bus interface is impractical when the mesh array occupies a large region. Therefore, the underlying reconfiguration circuitry is used to populate shift registers with inputs and readback outputs. Reading and writing configuration data is achieved by using the internal configuration access port (ICAP) [12]. The use of partial reconfiguration to distribute data throughout the FPGA fabric is novel as reconfiguration is originally intended to swap logic partitions for multiple use cases. This network is an often underutilized routing fabric which could potentially free up routing resources for a larger design. By making full use of the partial reconfiguration routing, a design requires a smaller area because less routing and logic is occupied.

IV. Xilinx Virtex-6 LUT-Bitstream correspondence

Knowledge of the bitstream format is required to populate the logic slices with pixel values. So far, the one-to-one correspondence between the contents of lookup tables and the requisite bitstream format has been poorly documented in both commercial and research literature. Therefore, this section details the work on deducing the LUT-bitstream correspondence on Virtex-6 architectures. The Xilinx FPGA Editor is used to alter the contents within LUTs of a slice and the bitstream of the modified configuration is generated. The original and modified bitstreams are then compared using RapidSmith [13].

Four consecutively addressed frames fully configure the LUT contents of a column of 40 region-aligned slices. The 256-bit

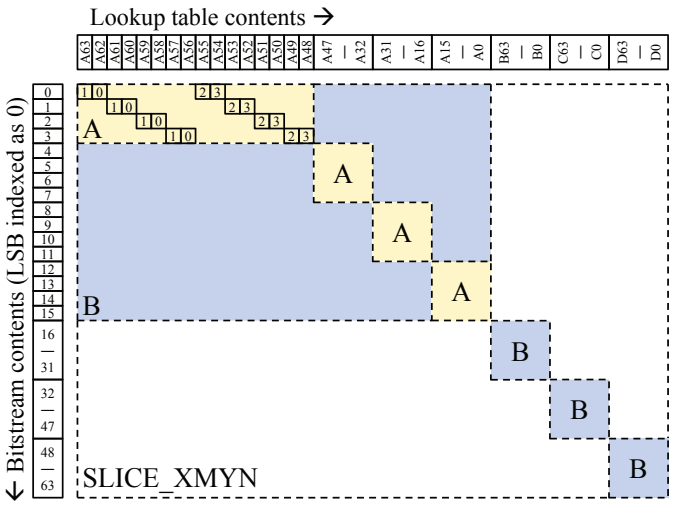

Fig. 2. Bitstream to LUT correspondence of a single slice

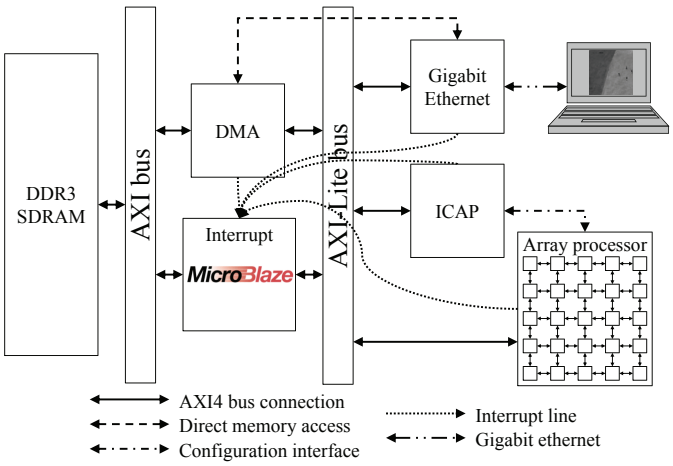

Fig. 3. System configuration

LUT contents of a slice consists of 8 words straddling across 4 frames [12]. Fig. 2 shows the detailed correspondence between LUT content and bitstream. The location within the bitstream which determines the value of the respective LUT entry is given by the intersection of both axes at a numbered box, the number representing the frame index where the bit resides.

The dotted boxes labelled by matching letters succinctly represents the recursive pattern of the bit correspondences. To give an example, given the 4 frame addresses which configure a slice as X, ..., X+3, the bit value in the entry A50 is determined by the 2nd bit of the bitstream which configures the frame addressed at X+3.

V. System Configuration

Fig. 3 shows the system configuration used to interface the array processor on the ML605 board. Peripherals and memories are connected together to a MicroBlaze processor using the AMBA AXI4 protocol. Peripheral control signals go through the slower AXI4-Lite bus, whereas high throughput traffic, such as DMA transfers, goes through the AXI4 bus. The following sections highlight pertinent details of the configuration.

A. User Constraints File (UCF)

Since data within the shift registers are exchanged using partial reconfiguration, LOC and BEL constraints [14] are used to place these registers at predetermined locations within the FPGA fabric. Assuming the ICAP is configured using a 32-bit interface at 100 MHz with 10% overhead, the array processor requires a minimum clock frequency of 0.31 MHz to reach the target 10000 frames per second. Therefore the clock net can be constrained to run at any frequency higher than 0.31 MHz. Lastly, the SAVE NET FLAG constraint prevents the removal of shift registers since they have no effect on external logic.

B. Operation

A computer passes video frames to the FPGA via a 1 GbE interface. The GbE hardware IP, programatically controlled using the Lightweight IP (LwIP) library [15], intercepts the data packets and populates a memory buffer using the DMA IP. The DMA interrupts MicroBlaze upon completion, which in turn starts transferring data from the memory to the ICAP. After the ICAP finishes populating the array, the MicroBlaze issues a start command to the array processor. The array processor interrupts the MicroBlaze when Laplacian filtering has completed, which then initiates a DMA transfer to pull configuration data out of the array through the ICAP to the memory. Processed data residing in the memory is then transferred out of the FPGA through the GbE interface to the computer.

VI. Methodology

Test video clips are obtained from the UCF-Lockheed-Martin UAV Data Set, courtesy of the Center for Research in Computer Vision at University of Central Florida[2]. The array processor is first simulated on ModelSim for functional correctness. On the ML605 hardware, test inputs are passed into the array processor through a workstation connected to the FPGA via a 1 GbE interface. The processed results are relayed to the computer for analysis. MATLAB is used to measure the amount of truncation error incurred by taking in the original grayscale images and computing the ideal Laplacian image using floating point precision. The MATLAB, simulated and actual outputs are compared on a pixel-to-pixel basis.

VII. Results

Fig. 4 shows the post-routed layout of the entire system on the ML605. The array processor (highlighted in green) is neatly sited in a rectangular region at the top left corner as a result of the UCF placement constraints.

A. Resource Utilization

The resource utilization of the array processor is shown in Table I. This agrees well with the model of the pixel processor shown in Fig. 1, where 4 slice flip-flops and 4 LUTs are

[2]The data set is accessible at http://crcv.ucf.edu/data/UCF_Aerial_Action.php.

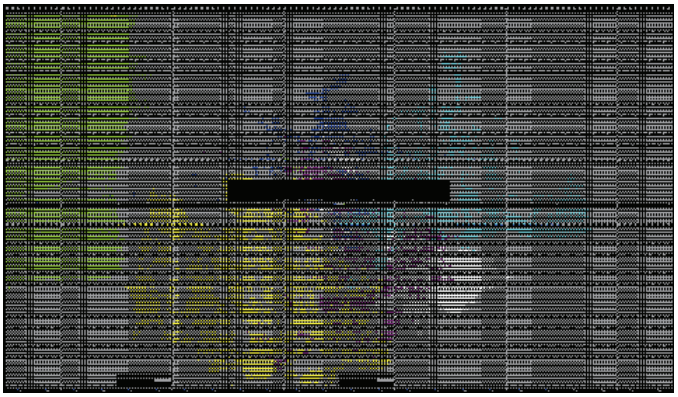

Fig. 4. Post routed layout on the XC6VLX240T. Coloured regions correspond to the following modules: light green – array processor, yellow – ethernet, blue – ICAP, cyan – DDR3 SDRAM bus interface, white – MicroBlaze, purple – AXI4 bus interface

required to implement a single processor. On average, a pixel processor consumes one configurable logic block. The figures under the occupied slice column gradually decreases as the size of the array processor increases, due to the fixed overhead involved in implementing the AXI4-Lite bus logic.

Table I
RESOURCE UTILIZATION OF ARRAY PROCESSOR PERIPHERAL

Size of array processor	Average resource per pixel			
	Occupied slices	Flip-flop	LUT	LUTRAM
2 × 2	4.000	3.500	4.000	4.000
4 × 4	2.813	3.875	4.000	4.000
8 × 8	2.328	3.969	4.000	4.000
16 × 16	2.145	3.992	4.000	4.000
32 × 32	2.061	3.998	4.000	4.000
40 × 40	2.063	3.999	4.000	4.000
60 × 60	2.080	3.999	4.000	4.000

B. Comparison Between MATLAB, Simulation and Implementation Outputs

The output images of the simulation and ML605 implementation are identical, whereas the outputs between the MATLAB model and the other two slightly differ due to truncation error in computing the quarter pixel value. The pixel-to-pixel error approximately follows that of the multinomial distribution function given by the coefficients of $P(x)$ in (2), where the term ax^b means that the probability of the pixel-to-pixel error being b is a.

$$P(x) = \frac{1}{256}\left(1 + x^{\frac{1}{4}} + x^{\frac{1}{2}} + x^{\frac{3}{4}}\right)^4 \tag{2}$$

The use of truncation to compute the quarter pixel value leads to an overestimation of the computed Laplacian value that is at most 3.0. On average, 0.58 bits of precision is lost in the computed Laplacian. Fig. 5 shows the output results of frame #1 of the test video clip. Observe that the outputs from the Verilog and FPGA are grainer than the MATLAB output due to the noise introduced by truncation error.

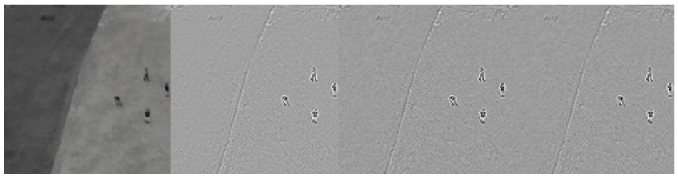

Fig. 5. (From left) Original frame #1, MATLAB output, Verilog simulation output and FPGA output

VIII. CONCLUSION

In this paper, the LUT-to-bitstream correspondence specific to Virtex-6 has been fully reversed engineered and documented. A working implementation of a 40 by 40 pixel has been realized on the ML605, which has been verified to be functionally correct with respect to its Verilog model. On average, a pixel processor requires 1 CLB. The array processor is able to achieve the target frame rate at a mere 0.31 MHz. To explain the discrepancy between the MATLAB and Verilog simulation outputs, a multinomial error distribution adequately models the truncation incurred.

REFERENCES

[1] D. Warrick, B. Tobalske, and D. Powers, "Aerodynamics of the hovering hummingbird," *Nature*, vol. 435, no. 7045, pp. 1094–1097, 2005.

[2] M. Thali, B. Kneubuehl, P. Vock, G. Allmen, and R. Dirnhofer, "High-speed documented experimental gunshot to a skull-brain model and radiologic virtual autopsy," *The American journal of forensic medicine and pathology*, vol. 23, no. 3, pp. 223–228, 2002.

[3] W. Zhang, Q. Fu, and N. Wu, "A programmable vision chip based on multiple levels of parallel processors," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 9, pp. 2132–2147, 2011.

[4] P. Dudek and S. Carey, "General-purpose 128× 128 simd processor array with integrated image sensor," *Electronics Letters*, vol. 42, no. 12, pp. 678–679, 2006.

[5] H. Zhu and T. Shibata, "A real-time motion-feature-extraction image processor employing digital-pixel-sensor-based parallel architecture," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 1612–1615.

[6] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki, and K. Tajima, "2000 fps real-time vision system with high-frame-rate video recording," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1536–1541.

[7] I. Ishii, T. Taniguchi, R. Sukenobe, and K. Yamamoto, "Development of high-speed and real-time vision platform, h3 vision," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3671–3678.

[8] P. Van Dokkum, "Cosmic-ray rejection by laplacian edge detection," *Publications of the Astronomical Society of the Pacific*, vol. 113, no. 789, pp. 1420–1427, 2001.

[9] B. Yousefi, S. Mirhassani, and H. Marvi, "Classification of remote sensing images from urban areas using laplacian image and bayesian theory," in *Proceedings of SPIE*, vol. 6718, 2007, pp. 1–9.

[10] K. Johansson, "Low power and low complexity constant multiplication using serial arithmetic," Ph.D. dissertation, Linköping, 2006.

[11] C. Nagendra, M. Borah, M. Vishwanath, R. Owens, and M. Irwin, "Edge detection using fine-grained parallelism in vlsi," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 1. IEEE, 1993, pp. 401–404.

[12] *Virtex-6 FPGA Configuration*, Xilinx Inc., September 2012, uG360 (v3.5).

[13] C. Lavin, M. Padilla, P. Lundrigan, B. Nelson, and B. Hutchings, "Rapid prototyping tools for fpga designs: Rapidsmith," in *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 2010, pp. 353–356.

[14] *Constraints Guide*, Xilinx Inc., January 2012, uG625 (v13.4).

[15] A. Dunkels, "lwip–a lightweight tcp/ip stack," *Available from World Wide Web: http://www. sics. se/ adam/lwip/index. html*, 2005.