System architectures for infrared pedestrian tracking

Robert Walczyk[†], Alexander Balazs[*], Alistair Armitage[**], T. David Binnie[***]

School of Engineering and the Built Environment
Edinburgh Napier University
[†]r.walczyk@napier.ac.uk
[*]09013567@napier.ac.uk
[***]td.binnie@napier.ac.uk

School of Computing
Edinburgh Napier University
[**]a.armitage@napier.ac.uk

Abstract — **This paper describes an FPGA-based implementation of a real-time pedestrian detection and tracking system for infrared (IR) video streams. The system includes hardware accelerators for adaptive background subtraction, morphological filtering and connected component labelling (pedestrian detection). Extracted features are provided to an embedded processor core for tracking analysis. The hardware/software co-design is discussed. The implementation is based on the XUP V2P (XC2VP30 FPGA) development board and uses the proprietary embedded design kit. The implementation features a reduced bandwidth scheme, hence the total memory requirement can be handled by the embedded memory blocks of a single FPGA chip.**

Keywords — **FPGA, Embedded Processors, System Architectures, Pedestrian Detection, Infrared Video.**

I Introduction

Thermal infrared imaging technology has advantages over conventional visible imaging technologies in the detection of people under conditions where there is little or no visible illumination. The recent reduction in the cost of infrared sensor arrays make thermal imaging commercially viable in industrial safety and security systems [1].

Automated real-time monitoring of thermal infrared video of an external environment is of interest where people detection and tracking is an application requirement. The processing of thermal image streams requires a different approach to that of visible video data. Conventional vision processing systems struggle to cope with variation in lighting, shadows, reflections and other ambient image conditions. Infrared images, although lower resolution, represent thermally emitted radiation which is more tolerant to changes in ambient conditions. Infrared image processing does however have to cope with other factors such as variation in background temperature, clothing, and a much lower signal-to-noise ratio.

For real-time people detection and tracking the processing systems must operate at frame rate (24-30 Hz). FPGAs allow the parallelism, pipelining and concurrency necessary to reduce clock latency and speed up the frame processing time. The facility of FPGAs to host on-chip processors which allow customisation of the processing architecture is particularly advantageous for application specific video processing.

A key aspect of developing an efficient implementation of a function on an FPGA with an on-chip hardware processor is the partitioning of the design into processor and logic functions and the interfacing of the processor with the surrounding reconfigurable logic. The way the processing tasks are divided between the logic and the processor affects the amount of FPGA resources that are used. The performance of FPGA applications that use an embedded processor is determined by the mechanisms chosen to enable communication between the processor and its surrounding resources. Partitioning can also significantly reduce development time and allow for the implementation of more complex algorithms. Once the interface between the processor and the FPGA fabric is defined the implementation of algorithms can become a software issue and the designer is no longer concerned about the utilisation of hardware resources

although if too heavy a load is carried by the processor the speed of operation will be compromised. Partitioning the design and developing the processor-logic interface is non-trivial. It requires the use of specialist software tools and technical knowledge of computer architecture and embedded system development.

In this paper Hardware (HW) Software (SW) co-design for pedestrian detection and tracking from IR video streams is described. To aid development, a simplified sample application was created to emulate the representation of pedestrians in an infrared video stream. The hardware system was partitioned for optimum hardware acceleration of computationally intensive tasks such as background subtraction, morphological filtering and connected component labelling. The embedded processor unit was used for the implementation of tracking algorithms.

The remainder of this paper is organized as follows: Section II gives an introduction into the field of real-time image processing using FPGAs, for visual pedestrian detection, and for pedestrian detection using infrared cameras. Section III describes the development platform for the HW/SW co-design. Sections IV, V and VI introduce hardware accelerators for adaptive background subtraction, morphological filtering and connected component labelling respectively. The system architecture for HW/SW implementation is discussed in Section VII. The following section describes implementation details of the sample application developed for the purpose of this project. The last section gives results, conclusions and plans for further work.

II Literature review

Real-time pedestrian detection is an active research area. Though, many software attempts for such implementations fail in practice due to heavy computational load. Video processing applications require high computing power and cannot be handled by ordinary processing platforms. The early stages of video processing are computationally intensive and can benefit from the hardware acceleration provided by FPGAs. Examples of these, depicted in Figure 1, include separating moving objects from the background, removing video acquisition noise and extracting features of interest. However, tracking algorithms are complex, and are best be handled by software in a CPU.

Different examples of such systems can be found in the literature [2][3]. They feature soft and hard processor cores respectively, either for housekeeping tasks such as to control the general behaviour of the system or for implementation of tracking algorithms. Both systems refer to pedestrian detection in good lighting conditions, however recently

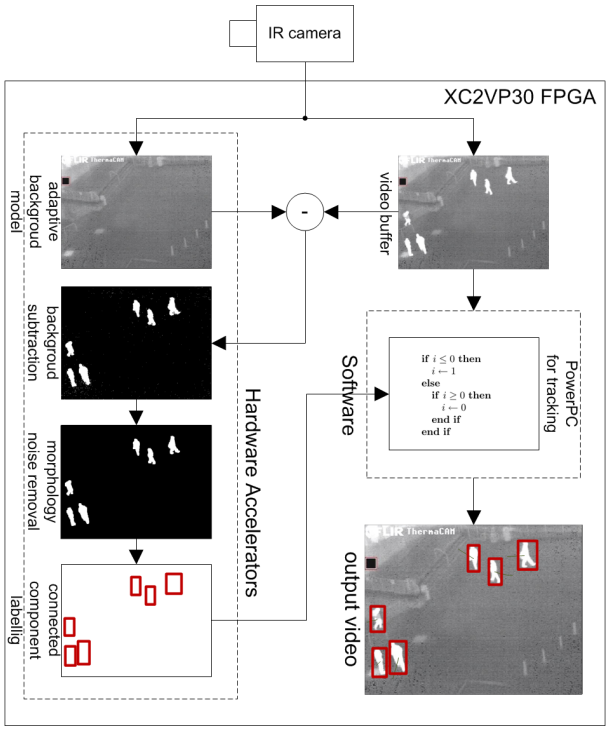

Fig. 1: Pedestrian detection and tracking system data flow

such applications are equipped with infrared imagers to become independent from lighting conditions [4].

Implementation of a real-time video processing system is non-trivial task. The system level integration includes highly demanding processing steps, depicted in Figure 1. The first step is to separate regions of interest (ROIs) from the background. Over the years different background separating algorithms have been proposed and compared [5][6]. The subsequent processing step is equally prevalent in the field of image processing as the predecessor. The low-level noise removal is based on the Mathematical Morphology (MM) and by applying multiple MM features at the same time, the architecture and memory requirement of this unit can be kept to the minimum [7]. The task that requires the heaviest resource is the Connected Component Labelling (CCL). This is a process where multiple features for each pedestrian are extracted, then forwarded for further processing (in this case tracking). In order to maintain real-time performance, a fully pipelined architecture based on the single pass CCL algorithm was developed [8]. The extracted features of the labelled objects are stored in the memory and shared with PowerPC 405. The processor, used to determine whether objects are connected (merged) and to track their move, was integrated in order to efficiently communicate with the FPGA [9].

III Development platform

This section gives detail of the development platform chosen for the purpose of this project.

IR Video Source
A FLIR Systems Thermacam PM595 was used as an infrared image source, equipped with 320×240 uncooled microbolometer focal plane array. The temperature range of the IR imager -40^{0}C up to 500^{0}C.

Video Decoder
The video source was digitized by the VDEC1 Video Decoder Board from Digilent Inc. This PCB is equipped with the ADV7183B Video Decoder chip from Analog Devices that can be freely configured over the I^{2}C protocol. The signal is converted into the digital data stream at 54MHz frequency by three 10-bit ADCs. The ITU-R BT.656 format defines output signal and provides information about the colour space, the number of samples and sampling format. The luminance (Y) colour channel is decoded and gives an 8-bit depth digital equivalent of the IR input signal.

FPGA Development Board
The XUP Virtex-II Pro FPGA Development System was chosen with the Xilinx XC2VP30 FPGA. The chip features 30,816 Logic Cells, 18 x 18-bit multiplier blocks, two PowerPC 405 hardware processor cores and 2,448 K bits of embedded RAM.

PowerPC Embedded Processor Core
The PowerPC 405 architecture is a 64-bit architecture with a 32-bit subset. Its memory management is optimised for embedded software environments. It has cache-management instructions for optimising performance and memory control in complex applications that are numerically intensive such as tracking. Furthermore the processor contains thirty-two 32-bit General-Purpose Registers (GPR) which can be accessed by all software as the GPRs are the means for getting data from the FPGA to the processor.

Debug and Verification
For the purpose of debug and verification the XSGA output port was interfaced. The real-time video stream (25MHz pixel clock) is displayed on the monitor connected through the standard DB15 connector. All the hardware accelerators operate synchronously at a pixel clock rate.

IV Adaptive Background Subtraction

Less than ten years ago pedestrian detection systems were implemented as background separation systems, whereas nowadays background subtraction techniques are used as one of the subsystems within the processing platform. There is a number of techniques suitable for such an implementation: running Gaussian average, temporal median filter, mixture of Gaussians (MoG), Kernel Density Estimation (KDE) and others [5][6]. The choice of the right algorithm is crucial for the overall system performance, it is the most memory demanding processing step of all. According to the literature, algorithms such as the running Gaussian average and the median filter offer acceptable accuracy while using less memory than MoG and KDE.

Since IR radiation gives limited information in terms of colour spaces (luminance Y represents the greyscale intensity), there is no particular application for the technique based on the MoG. For the purpose of pedestrian detection from infrared video streams, the running average algorithm based on the single Gaussian was chosen to model the background frame independently at each (i, j) pixel location.

An input pixel B at a time t can be classified as a foreground object if it holds the following inequality:

$$|B_t - \mu_t| > k\sigma_t, \tag{1}$$

where both parameters, probability density function (μ_t) and standard deviation (σ_t), are calculated in a run time over the period of n frames. Their values can be calculated in a similar fashion according to the following equation:

$$\mu_t = \alpha B_t + (1 - \alpha)\mu_{t-1}, \tag{2}$$

where the μ_{t-1} is the average computed in the previous image scan and the α is a an empirical weight often chosen as a trade-off between stability and quick response.

V Morphological Filtering

The morphological opening is an operation commonly used for filtering purposes in such applications [7]. It is based on erosion (ε) followed by dilation (δ), hence all the MM set theory features apply. Thanks to ε random image pixels will be removed, whereas δ merges objects that split during the preceding operation.

A popular approach for implementation of such application is a raster scan based system with a spatial filter mask where consecutive lines are stored within line buffers. Depending on the operation, logic AND and OR are applied on the neighbouring pixels within the filter mask for ε and δ respectively.

Thanks to MM set theory features, assuming that the Structuring Element SE (spatial mask) is reflection invariant ($SE = \widehat{SE}$) and decomposable ($SE = SE_1 \oplus SE_2$), it can be implemented in more efficient way with only a single FIFO to store intermediate processing data [7]. For a binary input image B, erosion can be split into two subsequent

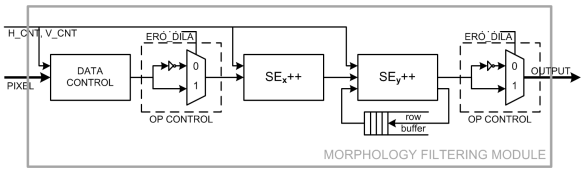

Fig. 2: Block diagram of the decomposed dual system architecture for erosion and dilation

tasks:

$$\varepsilon_{SE}(B) = B \ominus (SE_1 \oplus SE_2) = (B \ominus SE_1) \ominus SE_2 \quad (3)$$

whereas, according to MM set theory, ε and δ are dual of each other with respect to set complementation:

$$B \oplus SE = (B^c \ominus \widehat{SE})^c \quad (4)$$

Therefore, dilation can be written as follows:

$$B \oplus SE = (B \oplus SE_1) \oplus SE_2 = (B^c \ominus SE_1)^c \oplus SE_2 = \\ = ((B^c \ominus SE_1)^{cc} \ominus SE_2)^c = ((B^c \ominus SE_1) \ominus SE_2)^c \quad (5)$$

From the implementation point of view, according to Equation (5), the architecture based on morphological operations can be significantly optimized. By splitting SE, the results of one processing stage can be processed by the subsequent unit in a pipelined fashion. Moreover, the same architecture can perform both ε and δ by inverting input and output pixels. Such an architecture, depicted in Figure 2 is small and capable of processing streaming data with minimal latency and low number of comparisons.

VI Connected Component Labelling

Object detection algorithms, often referred to as Connected Component Labelling (CCL) algorithms, can be categorised according to whether they scan the entire image to locate components, or trace their contour or process a number of pixels at a time in parallel. They differ in execution time as well as in memory requirement. A successful implementation of a single-pass CCL algorithm described in [10] requires only a single scan through the image frame in order to label all the objects and extract their features of interest such as location, coordinates of the bounding box, size or Centre of Gravity (CoG).

The implementation based on a single-pass CCL algorithm requires relatively low amount of hardware resources comparing with other real-time processing techniques such as the classical 'two-scan' approach and the countour tracing based. Moreover, it features over 142× and 18× memory saving respectively for images at a resolution of 320 × 240 pixels with up to 255 objects per image frame.

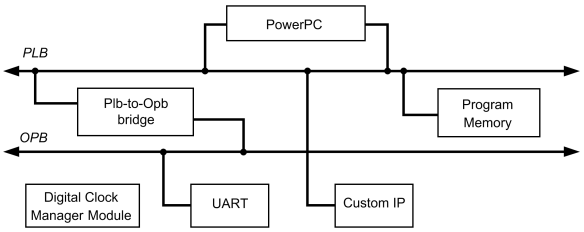

Fig. 3: Block diagram of the processor's bus connection

VII System Integration

In this section the integration of custom logic into the embedded system is described. This section details how the wrapper for Hardware/Software (HW/SW) implementation of the pedestrian detection application has been created. For this implementation, an early design decision was to implement the FPGA logic functions as a subsystem of the embedded processor. A further design decision was the choice of bus system for the processor-logic interface. This involves the optimisation of the number, width and usage mode of the software addressable registers. The register values can be updated by the processor in software and read by the logic or updated by the logic and read by the processor. Proprietary tools are used to generate a system where the processor is connected to the program memory blocks and a digital clock manager. These modules are represented as peripheral cores. The important implementation parameters are the processor internal clock speed, the bus speed and program memory size. Any peripheral core must be connected to either the Processor Local Bus (PLB) or the On-chip Peripheral Bus (OPB) to communicate with the processor core. When connected to the OPB, the peripheral core becomes part of the memory map accessible to the processor. The peripheral core will have a base address and a high address signifying where in the memory map it resides, and how much memory it occupies. The processor thus interacts with the peripheral cores as if they were part of the memory. Bus connected peripheral cores are assigned with a unique address. In order to use the peripherals, appropriate driver files and libraries have to be included.

The HDL-based modules which make up the application are joined together to form a custom core. The main steps for integrating the custom core are as follows:

- Interfacing the IP via PLB bus in slave mode. According to [9], the OPB bus is dedicated for slower peripherals such as an UART. Moreover, in order to communicate with the processor, the data needs to be transferred over additional OPB to PLB bridge. This can be seen in Figure 3.

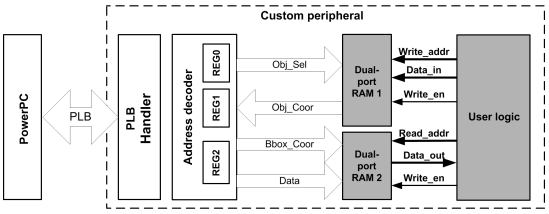

Fig. 4: Register level interface

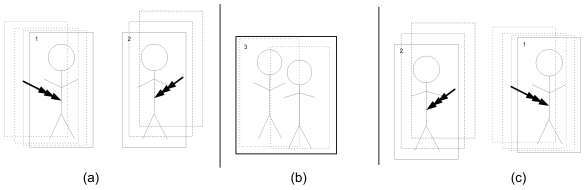

Fig. 5: Pedestrians merging. (a) Before merge. (b) During merge. (c) After merge.

- Selecting software accessible registers.
  This determines the contents of the wrapper for HDL files responsible for communicating with the bus.
- Selecting the number of registers and their width.
  For the purpose of this implementation, three 32-bit registers have been chosen. Figure 4 shows how this design is interfaced with the processor using system registers. This is further discussed in the subsequent section.

For integration of the peripherals within the system, the peripheral has to be given an address by the address generator after it has been manually connected to the PLB. The I/O signals had to be changed to external to be able to interact with external peripherals (VGA display, buttons, switches). The steps above are crucial for the customisation that enables communication between FPGA logic and processor. A key part of this operation is the generation of macros which define the read and write routines of the software addressable registers. Every time a read or write is issued in software on any of the registers the custom cores base address is used as a parameter for those routines.

VIII Implementation

The purpose of the pedestrian detection application was to emulate random pedestrian movement without the need to integrate the video processing platform. Different movement speed and direction can be controlled separately by making use of the manual input peripherals (switches and buttons) on the FPGA development board. Depending on the current movement direction, addresses for the pedestrian objects are generated by the FPGA. With reference to Figure 4, these addresses, one horizontal (x) coordinate and one vertical (y) coordinate, are written into the Block RAM (Dual Port RAM1). After reading and storing these values the processor creates object sized rectangles around the objects and checks if they overlap. If there is an overlap, then a rectangle with size that contains both objects is created by the processor to illustrate that there has been a merge. A second function is to provide binary data at coordinate addresses making up all the rectangles. With this implementation the constant flow trail of the objects is handled by the FIFO buffer. In Figure 4, the address decoder and PLB handler are part of the application wrapper, whereas gray boxes refer to the user logic. The object- coordinates are updated at 60Hz rate leaving plenty of time for the processor to generate data. A memory element, also depicted, has been added to the system to store the different objects' coordinates in different addresses. REG0 holds the ID of the object, REG1 holds the issued objects position. Since REG0 and REG2 are controlled by the processor they are set to write mode in the bus setup. REG1 is set to read mode.

The video processing works in the way that once an object is labelled it is represented by two sets of coordinates, namely the top-left and bottom-right corner coordinates. This data is then written into a block RAM so that every labelled object's data goes into a different address. The processor gets the data from that memory block via registers. From this data the processor creates the bounding boxes and provides data for the FIFO in order to leave flow trails of the objects' movements. A problem occurs when objects that are heading in opposite directions merge as depicted in Figure 5. At that moment the two objects are labelled as one, therefore only one set of coordinates is available for the processor. For this problem an interesting approach has been suggested in [11] for multiple object tracking. In order to keep the bounding boxes around the object during the merge event, the processor has to constantly preserve the height and width of each object even when only two sets of coordinates are received, because of merge.

IX Results and Conclusions

In this paper the HW/SW co-design for real-time pedestrian detection and tracking from IR video streams is described. The system was partitioned into logic hardware acceleration, to perform video processing tasks, and the software unit, hosted on the embedded hard processor core for the implementation of tracking algorithm.

With the proposed architecture significant memory savings were achieved ranging from 30% for

morphological filtering unit (comparing with classical buffer-line based approach) up to 18× lower memory utilisation comparing with currently the most efficient CCL implementation based on the 2-bit variation of the contour tracing algorithm. Moreover, these figures increase exponentially with increasing image size.

The system (hardware accelerators) operates at a frame rate at the 25MHz pixel clock frequency with the bus and PowerPC operating at 100MHz. For the 320 × 240 pixels video stream and up to 127 objects detected per image frame the overall system occupies 129 out of the 136 on-chip memory blocks. Moreover, such an implementation requires only 9% of flip-flops and 12% of 4-input LUTs of the XC2VP30 FPGA.

The system was integrated, verified and tested with both emulated pedestrian movement as well as with real-time video data. Currently, the algorithm implemented in software handles simple merges and splits (object partially occluded) using coordinates of two corner points of the bounding boxes. However, in order to cover more complex scenarios, we are working on multiple object tracking based on the Kalman filtering where other extracted features such as size or CoG would be employed.

References

[1] J.E. Scalera, III Jones, C.F., M. Soni, M.B. Bucciero, P.M. Athanas, A.L. Abbott, and A. Mishra. Reconfigurable object detection in flir image sequences. In *Field-Programmable Custom Computing Machines, 2002. Proceedings. 10th Annual IEEE Symposium on*, pages 284 – 285, 2002.

[2] Vinod Nair, Pierre-Olivier Laprise, and James J. Clark. An fpga-based people detection system. *EURASIP J. Appl. Signal Process.*, 2005:1047–1061, January 2005.

[3] Fredrik Kristensen, Hugo Hedberg, Hongtu Jiang, Peter Nilsson, and Viktor Öwall. An embedded real-time surveillance system: Implementation and evaluation. *J. Signal Process. Syst.*, 52(1):75–94, 2008.

[4] Maoxiang Huang, Chenhao Wang, and Yuncai Liu. High-speed video transfer and real-time infrared spots detection based on fpga. *Computer Science and Engineering, International Workshop on*, 2:154–159, 2009.

[5] M. Piccardi. Background subtraction techniques: a review. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3099 – 3104 vol.4, 2004.

[6] Y. Benezeth, B. Emile, and C. Rosenberger. Comparative study on foreground detection algorithms for human detection. In *Proceedings of the Fourth International Conference on Image and Graphics*, ICIG '07, pages 661–666, Washington, DC, USA, 2007. IEEE Computer Society.

[7] H. Hedberg, F. Kristensen, and V. Owall. Low-complexity binary morphology architectures with flat rectangular structuring elements. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 55(8):2216–2225, 2008.

[8] R. Walczyk, A. Armitage, and T. D. Binnie. Comparative study on connected component labeling algorithms for embedded video processing systems. In *IPCV'10. CSREA Press*, volume 2, 2010.

[9] J. Noseworthy and M. Leeser. Efficient use of communications between an fpga's embedded processor and its reconfigurable logic. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, FPGA '06, pages 233–233, New York, NY, USA, 2006. ACM.

[10] R. Walczyk, A. Armitage, and T. D. Binnie. Fpga implementation of hot spot detection in infrared video. In *Signals and Systems Conference (ISSC 2010), IET Irish*, pages 233 –238, 2010.

[11] Xin Li, Kejun Wang, Wei Wang, and Yang Li. A multiple object tracking method using kalman filter. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 1862 – 1866, 2010.