Xilinx FPGA Implementation of a Pixel Processor for Object Detection Applications

Peter Mc Curry, Fearghal Morgan, Liam Kilmartin

Communications and Signal Processing Research Unit,
Department of Electronic Engineering, National University of Ireland, Galway.
Tel: +353-91-524411, Fax: 750511
E-mail: peter.mccurry@nuigalway.ie

Abstract

This paper describes an FPGA and distributed RAM architecture for an image pixel processor implementing primary elements of an object detection system. A comparison of the system performance with existing DSP processor-based alternatives is detailed. An implementation using the RC1000-PP FPGA-based development platform and Handel-C hardware programming language is outlined. A system architecture is proposed for an image pixel processor for elements of a machine vision based object detection system.

Keywords
FPGA, Virtex, RC1000-PP, Handel-C, Object Detection, Image Processing.

1. Introduction

Considerable research is currently ongoing, in many different fields of signal processing, to evaluate and compare programmable software and hardware platforms for the implementations of various signal processing algorithms. A design decision regarding which type of platform to use for a particular algorithm is particularly crucial where the sampling frequency, or the dimensionality of the signal being sampled, is high.
Previous research [1] describes a real time object detection, image-processing system based upon a network of software programmable TMS320C44 floating-point digital signal processors. This paper examines an alternative implementation of some of the object identification algorithmic stages ref. [1] using an FPGA-based platform and multiple independently accessible RAM data banks. The particular image processing algorithms examined are generally applicable to many different machine vision and image coding applications. The paper also compares the processing capabilities of a Xilinx FPGA-based platform with those of the system based on a dedicated signal processor.

2. Object Detection Application

The application outlined in [1] provides automated score detection in the game of hurling by detecting the presence of the object (ball) in an image frame and monitoring the ball's position and motion in three dimension relative to the known position of the goal posts. Figure 1 shows a block diagram outlining the algorithmic structure of the object identification process of the system already developed.

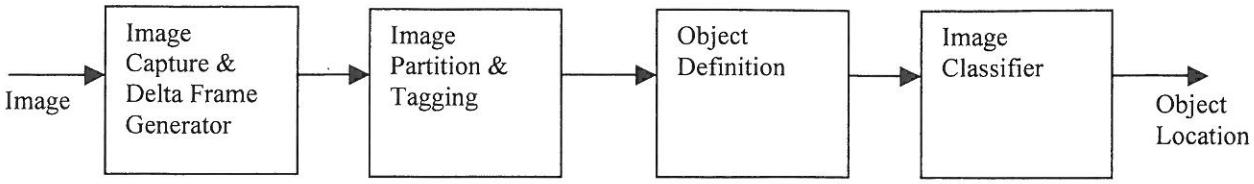

Figure 1 – Object Identification Process Algorithmic Block Diagram

The object identification process can be broken down into the following stages:

1. Image Capture and Delta Frame Generation

   The image capture and delta frame generation stage of processing identifies any regions of motion in each incoming monochrome video frame in which an object may exist. A difference, or delta frame, is generated by subtracting each incoming video frame from a reference background frame (in which the object does not exist), on a pixel-by-pixel basis. Non-zero data in the delta frame indicates motion. This technique is often used in image recognition systems ref. [2]. The background frame is updated at regular frame intervals in order to keep the background image up to date with the scene being observed by the system.

2. Image Partition and Tagging

   The delta frame is then further analysed by sub-dividing the complete delta image frame into 8x8 pixel square regions, termed blocks. An energy value is calculated for each of these blocks by simply summing the square of the pixel values forming the block. If the energy value for a given block is above a certain threshold value, then the block is tagged to indicate that it contains an element of motion compared to the background and hence requires further analysis to determine if it contains the object object.

3. Object Definition and Object Classification

   The classification stage makes a decision as to whether any 16x16 pixel square region of interest of the delta image contains the object. A single isolated block can form the centre of the 16x16 pixel square region passed on to the classification stage. However, before block image data is passed to the image classification stage, the object definition stage first determines if any tagged blocks have neighbouring tagged blocks. If so, a 16x16 pixel square region formed as an overlap of the neighbouring blocks, is passed to the classification stage. Whereas some of the morphological algorithms in the object classification stage are specific to the particular application for which the system outlined in [1] was developed, there are a number of filtering and pixel processing based routines which are of a more general nature, and hence could be applied to other image processing applications.

A two-dimensional video frame position of the object may be transferred to a further processing stage, which estimates the precise three-dimensional position of the object.

3. DSP Processor based Object Detection System

This section outlines the operation and limitations of the DSP processor solution described in [1]. The complete video processing system outlined in [1] is implemented on a network of interconnected Texas Instruments TMS320C44 processors. Figure 2 shows the structure of the processor network and the tasks allocated to each processor.

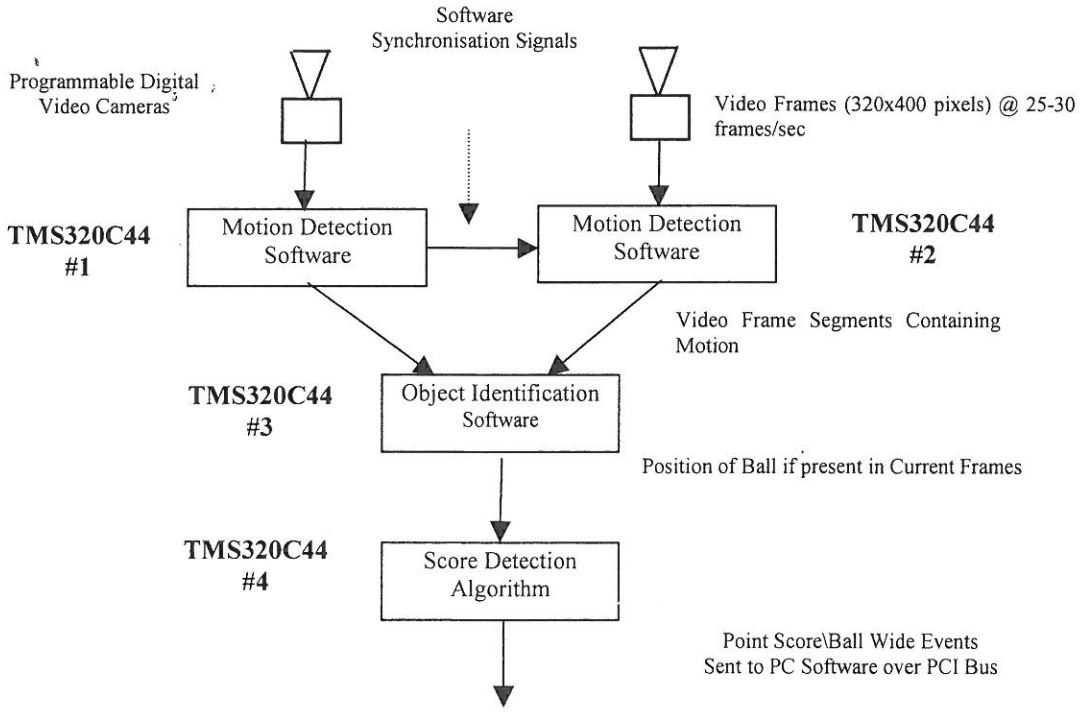

Figure 2. TMS320C44 Processors Configuration for Object Detection

The TMS320C44-based system outlined above is capable of a maximum throughput of approximately 30 frames per second for a frame size of 400x300 pixels and using a 30 MHz clock. This provides reasonable performance for object speeds less than 100 kph. However, this system operates at nearly full processing capacity and, for higher object speeds, common for the sliotar in the game of hurling, the accuracy of the overall system deteriorates rapidly.

Improved performance and accuracy could be maintained if the system:

(i) Used higher clock speeds
(ii) Used more co-processors and distributed memory banks
(iii) Used higher frame rates (up to 100 frames per second) and a suitable video camera
(iv) Processed larger frame sizes, e.g., 512x512 pixel frames

This paper considers an alternative, FPGA-based, pixel and image-processing platform using multiple distributed independent memory banks and compares performance to that of the general purpose DSP processor approach.

4. FPGA-based Implementation

This section describes the application of FPGA technology and distributed memory banks to the object detection application.

4.1 FPGA Technology and RAM bandwidth

FPGAs allow large-scale parallel processing and pipelining of data flow. Latest FPGAs provide enormous processing resources, significant on-chip RAM and support very high clock speeds. FPGAs are therefore suitable for implementing object detection systems discussed above. Implementation of the first two stages of the object detection system using an FPGA is described in this paper. The system supports a 512x512 pixel frame size. Since each pixel contains one byte of data, the frame size is 262,144 bytes. However, even the significant on-chip RAM provided in Xilinx Virtex FPGAs [3] (262,000 bits in the case of XCV400) is not sufficient to support a useful level of internal RAM frame buffering in the object detection application described. Therefore, additional external SRAM banks are required to provide storage during processing of image data arrays. The high I/O capability of FPGAs supports access to multiple RAM banks simultaneously, enabling effective and efficient pipelining. Partitioning and pipelining of the algorithms using multiple, distributed memory banks offers significant improvements in performance compared to the processor-based implementation described. The use of multiple memory banks removes a significant performance bottleneck.

4.2 RC1000-PP Virtex Based Implementation

Figure 3 illustrates the system level architecture of the pixel processor, comprising an FPGA, internal RAM and four separate SRAM memory banks.

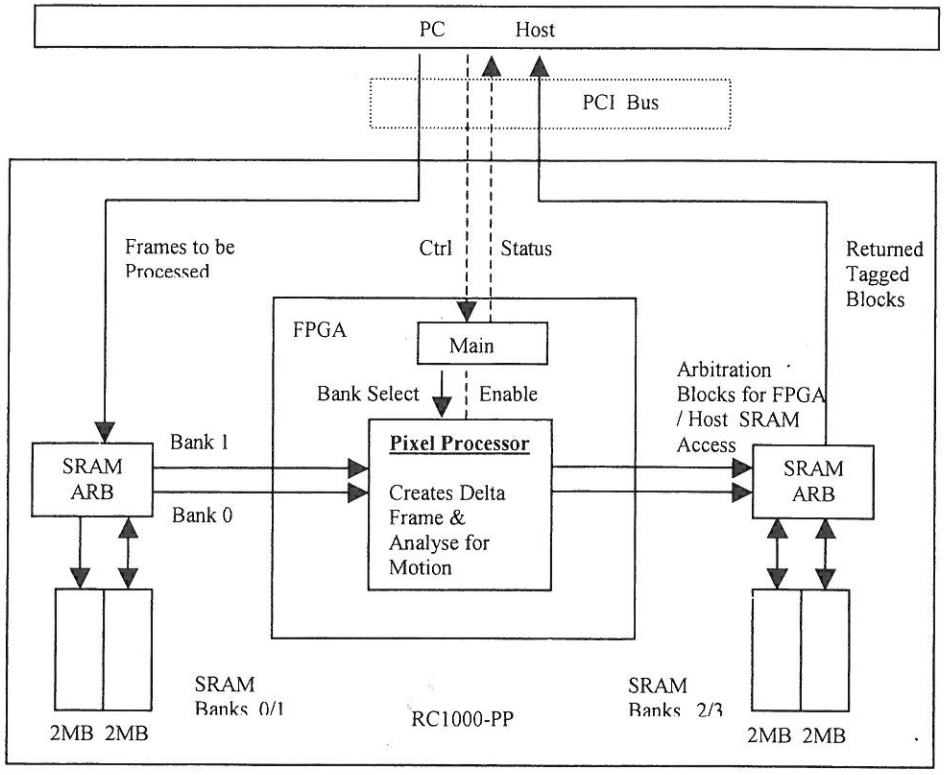

Figure 3. System level architecture of pixel processor.

The RC1000-PP development system [4] provides such an architecture and comprises a Xilinx Virtex (or XC4K) device, 8 Mbytes of external SRAM divided into four separate independently accessible RAM banks, and a PCI interface to host PC. The RC1000-PP development system comprises a XCV400 Virtex FPGA, offering up to 468,000 usable gates and 10kbytes of internal RAM. The development system uses the Handel-C programming language to define FPGA functionality. Handel-C is a C like language, which supports parallelism and flexible data size. The Handel-C code is compiled and translated into a Xilinx netlist format before place and route, pin allocation and creation of FPGA configuration bitstream file. Handel-C simulation enables early high-level simulation of the system and promotes fast proto-type development. Pre-defined routines such as RAM read/write, FPGA/host interfacing, I/O control etc enable development of a very effective user interfaces. It should be noted that Handel-C requires use of an entirely synchronous design methodology.

4.3 Object Detection System Level Description

External SRAM banks 0 and 1 are used to store 16 image frames, including the current and background frames. The host writes current image to SRAM banks 0/1 before relinquishing ownership to the FPGA for processing. All accesses by host and FPGA are 32-bit allowing concurrent processing of 4 pixels. The Handel-C function Main handshakes with the host, controls the FPGA pixel processing unit. SRAM banks 2/3 alternately store a reduced image data set, containing only tagged block pixel data on a frame-by-frame basis, as well as frame number, time and the location of the tagged block within the frame. Further processing (object definition and object classification) can be performed either on the host or, for greater performance, on the FPGA. Considerable spare FPGA bandwidth, internal RAM and the availability of the tagged image data alternately in bank 2 or 3 enables substantial object definition and object classification processing to be performed by the FPGA in parallel with subsequent frame tagging. This is currently under investigation.

4.4 FPGA Pixel Processor Core Architecture

Figure 4 illustrates the pixel processor core architecture.

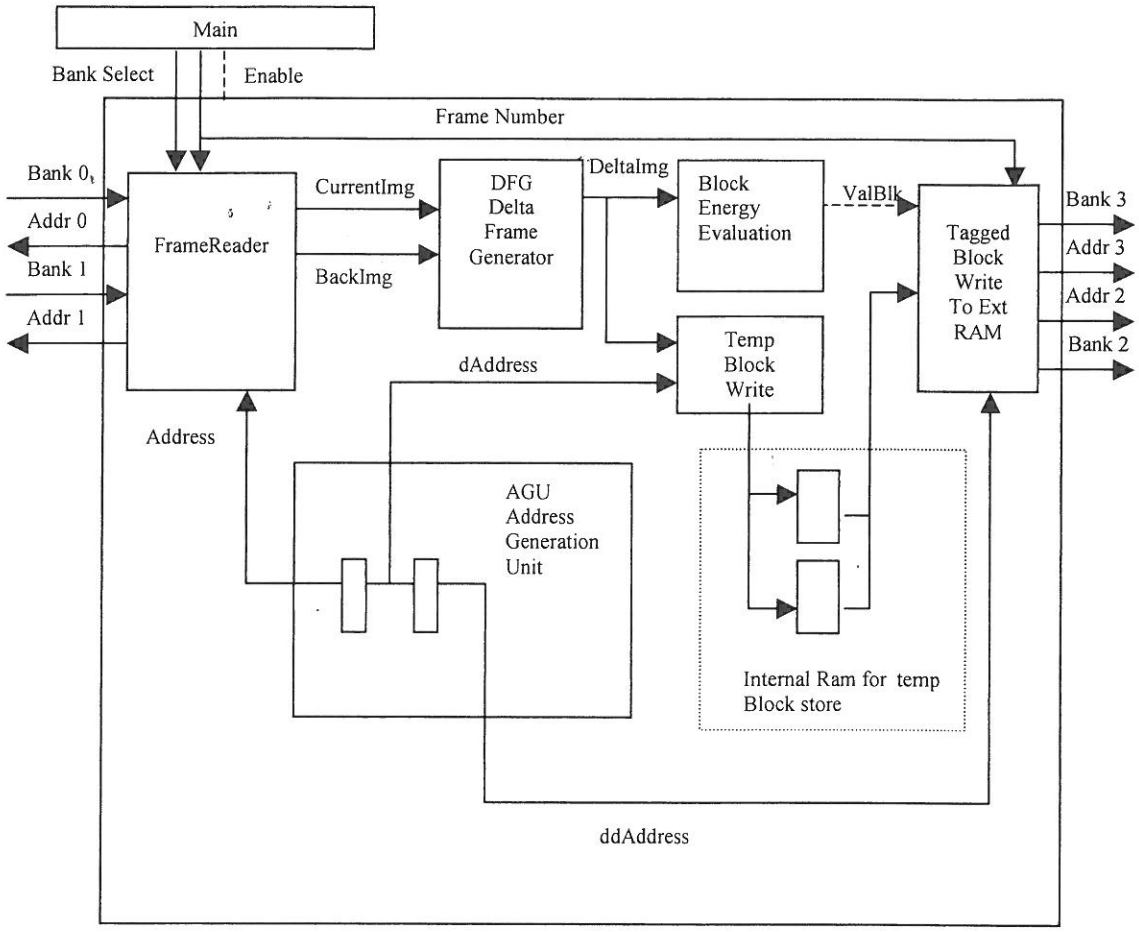

Figure 4. The pixel processor core architecture.

The following pipeline stages execute within the FPGA, four pixels at a time, on each block in sequence:

1. Concurrent read of background and current pixel
2. Subtract current/background to obtain absolute pixel difference
3. Temporarily save each pixel of block data to internal RAM (64 bytes of Block Store required). If the sum of the 64 pixel delta values within the block is greater than a pre-defined energy threshold (performed by the Block Energy Evaluation Unit), write the block data to external RAM bank 2 or 3. Use two internal RAM banks to store alternate block data until it is written to external RAM if tagged. Store frame number, tagged block location data, time etc for subsequent object definition and classification.

The Address Generation Unit (AGU) provides both the external and internal RAM addresses.

4.5 System Performance

The delta frame generation, block partition and tagging have been implemented and demonstrated using the RC1000-PP FPGA development system and Handel-C hardware programming language. A system clock frequency of 33MHz is used. External SRAM cycle time is 17ns. Pixel data is stored and processed 4 pixels (32-bits) at a time. FPGA frame tagging execution time for a 512*512 pixel frame is therefore 30nSec * 65,536 = 1.966mSec. FPGA frame tagging and host current frame writes are performed sequentially in the proposed system.

5. Comparison of FPGA vs DSP Processor Implementation

In order to fully understand the increased flexibility and performance, which the RC1000-PP Virtex-based system platform offers, compared to the software programmable DSP solution [1], a benchmarking test was carried out. This involved determining the maximum processing time required to implement the first two stages of the object identification process, on an image frame of 512x512 pixels. In addition to comparing the best processing performance of the TMS320C44 based system and the RC1000-PP Virtex based system, initial experiments were also carried out to estimate the best processing performance for the algorithms using TMS320C62x and TMS320C64x devices, Texas Instrument's cutting edge flagship processors and cores, which are based on a highly paralleled ALU. Table 1 illustrates the results.

Platform	Xilinx Virtex (100MHz Clock)	TMS320C44 (30 MHz Clock)	TMS320C62x (300 MHz Clock)	TMS320C64x (1.1 GHz Clock)
Processing Time (mSec)	1.966	77.4	3.7	3.15

Table 1 Comparison of processing times for first two stages of object detection, 512x512 pixels per frame
(Calculations for TMS320C6x devices assume use of SRAM with 3 nSec access times)

Results show that the FPGA-based implementation is significantly faster than the currently implemented TMS320C44 based system. Additionally, results show that the FPGA system is also superior in terms of processing power requirements than even the high-end TMS320C6x devices. This is primarily due to the ease in which the FPGA device can be used to parallel and pipeline execution of the basic quad pixel based processing (i.e. subtraction, multiplication, addition etc.). Using faster SRAMs and higher FPGA clock speed would increase performance further. Current research is focusing on a similar comparison of the processing platforms for the more complex image processing algorithms, which exist in the later processing stages, and an examination, of how much of the complete objection identification and tracking system can be integrated onto a single FPGA-based platform.

6. Conclusion

We have described the application of FPGA and distributed RAM to image object detection and compared performance with DSP processor-based alternatives. Image delta frame generation, block partition and tagging have been implemented and demonstrated using the RC1000-PP FPGA development system and Handel-C hardware programming language. Results demonstrate the performance advantages of the proposed solution to high I/O bandwidth and computationally intensive processing applications compared to DSP processor implementation. Current and future work includes similar implementation of the object definition and object classification processing stages.

Acknowledgements
This research is funded by Xilinx Ireland.

References

1. L. Kilmartin, M. O Conghaile, "Real Time Image Processing Object Detection and Tracking Algorithms", Proceedings of the Irish Signals and Systems Conference, NUI, Galway, June 1999, pp. 207-214

2. M. Fahy, M. Siyal, 'An image detection technique based on morphological edge detection and background differencing for real-time traffic analysis', Pattern Recognition Letters 16 1321-1330, 1995

3. The Programmable Logic Data Book, Xilinx, 1999.

4. H. Styles, W. Luk, "Customising Graphics Application : Techniques and Programming Interface", IEEE Symposium on Field Programmable Custom Computing Machines (FCCM00), April 2000.