2006 10th International Workshop on Cellular Neural Networks and Their Applications, Istanbul, Turkey, 28-30 August 2006

A Control System for a Cellular Processor Array

David R. W. Barr, Stephen J. Carey, Alexey Lopich and Piotr Dudek

School of Electrical & Electronic Engineering
The University of Manchester, PO Box 88, Manchester, M60 1QD, United Kingdom
e-mail: d.barr@postgrad.machester.ac.uk; s.carey@manchester.ac.uk; a.lopich@postgrad.manchester.ac.uk
p.dudek@manchester.ac.uk

Abstract - **Presented in this paper is a system that controls the SCAMP-3 cellular processor array vision chip, and provides an interface between it and a host system. This system can be used in real-time image processing, or computer vision applications. The system includes a sequencer for issuing instructions to the array processor, a configurable analogue interface and read-out circuitry, a system controller, which enables system operation and communication with the host, and a suite of software components, which include libraries, simulator, compiler and user interface. The presented hardware is a stand-alone vision system, which can be used in the development of PC-based or embedded applications.**

Index Terms **– Control Systems, Processor Arrays, SIMD**

I. INTRODUCTION

Processing an image traditionally requires several stages of preliminary computation. It is these stages that are usually the most computationally intensive, having significant contribution to the overall limit on the throughput of an image processing system. A sequential machine will perform the same operation on each pixel of the image in turn, producing a bottleneck that becomes more substantial as the dimensions of the image increase. A solution therefore is to process many pixels in parallel, using a processor array. Vision chips provide pixel parallel operations, but require peripheral interfacing hardware. Several systems that interface to various vision chips have been described in the literature [1]-[5].

The SCAMP-3 vision chip [6] is a 128x128 cellular processor array. The array consists of 16,384 analogue processing elements (APE), each coupled with a photo-sensor that captures light focused onto the array. The image data is stored as an analogue signal within each APE, ready to be operated on via a set of instructions. These instructions include inverting, addition, division and nearest neighbour communication. An APE contains 8 analogue registers and a digital flag bit. The SCAMP-3 vision chip is a Single-Instruction Multiple-Data (SIMD) device, illustrated in Figure 1. A single Instruction Code Word (ICW) is broadcast to every element of the array, and (depending upon the state of the local activity flag bit) is executed by all the APEs. As each APE can contain different values in its corresponding registers, there are multiple data streams processed with a single instruction. A specific sequence of ICWs dictates the image algorithm being performed. The current generation of SCAMP vision chips must have a sequence of ICWs supplied externally. The processor array also requires several clocks and bias voltages, and a mechanism for reading-out the processed data, which implies that an external control system is required.

This paper presents an entire image processing system, from the sensing/processing device (the SCAMP-3 chip) and its peripheral driving hardware (a control system and instruction sequencer) to the host controlling system (typically a desktop PC). The system allows the user to load an algorithm to the hardware, execute it, and return the processed data to the host.

II. SYSTEM OVERVIEW

The SCAMP system comprises five distinct components, shown in Figure 2. These are the SCAMP-3 vision chip,

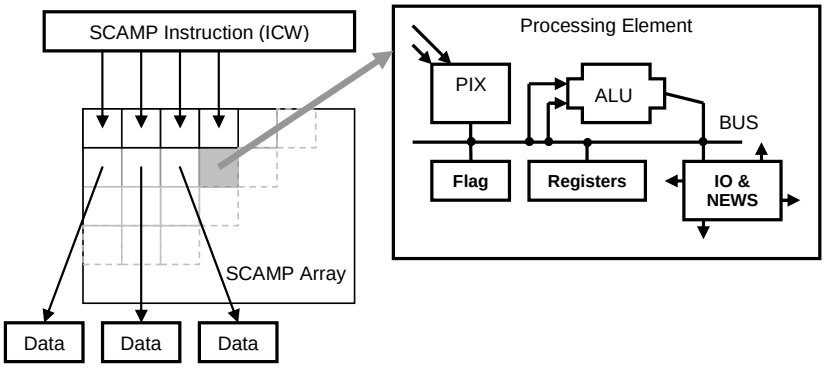

Fig 1. An overview of SCAMP-3 vision chip architecture. An example of SIMD architecture and an Analogue Processing Element (APE) are shown.

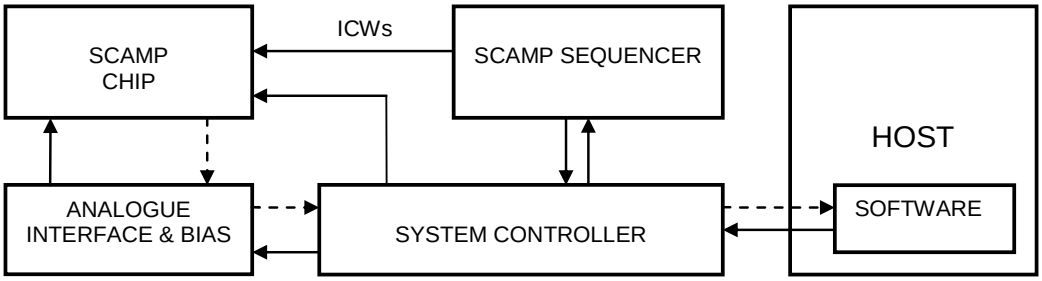

Fig 2. The connections between the 5 system components. The dotted arrow shows the path of processed data through the system

System Controller, SCAMP Sequencer, analogue interface (consisting of amplifiers, ADC & DACs, bias and power supplies), and a host software interface. The main requirements of the system are to control program execution (the sequence of ICWs delivered to the processor array) and to provide an interface to transfer processed data from the array (a read-out routine). This provides the user with a platform which can be used for both the development of image processing applications, or as an autonomous system. Typically, a user will construct and simulate an algorithm in software, and then send it to the hardware, where it will be executed. The system controller interfaces with the host and configures the sequencer with the ICWs to be delivered to the processor array. When the executing algorithm requests that processed results be returned to the host, a system controller routine reads out data from the array. This data could be digital or analogue – in which case it is then converted to an 8-bit digital word. The system controller packages up the data and sends it to the host. Flexibility has been achieved with all power supplies and bias voltages being configurable at run-time, and the digital components of the system have been implemented on an FPGA configured by the host at system start-up.

III. THE SCAMP ICW SEQUENCER

The SCAMP-3 chip executes a sequence of digital instructions that constitute an algorithm. This sequence of ICWs could be non-incremental, as the programmer could use loops and conditions within the algorithm. Thus, the delivery of ICWs to the SCAMP-3 chip is facilitated via a dedicated sequence controller (the SCAMP Sequencer), implemented using an embedded 8-bit microprocessor. A SCAMP algorithm op-code is 64 bits long, and consists of two parts. The first 17 bits are an instruction for the embedded microprocessor, and the remaining 47 bits comprise the ICW, which the SCAMP-3 processor array uses to perform analogue image processing. The entire 64-bit word is stored in a program memory, in a "dual instruction stream" configuration, illustrated in Figure 3. This creates a situation where the microprocessor is implicitly controlling the sequence of ICWs being executed by the SCAMP-3 chip, facilitating conditions, jumps and numerical operations within algorithms. The microprocessor executes its

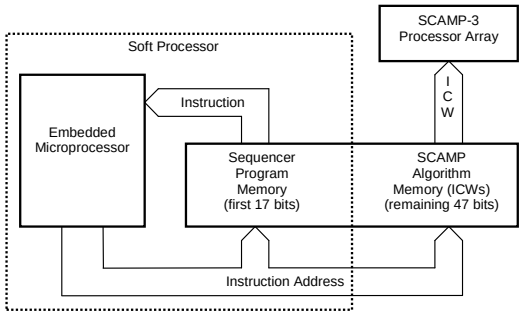

Fig 3. Illustration of "dual instruction stream" memory, with one stream being executed by the sequencer, and the other by the array processor

Line Number	Instruction	Target Processor	Comments
23	…	…	…
24	**A ← B**	SCAMP-3	Copy the contents of array register B into array register A
25	**_load s0, 0**	Microprocessor	Initialise a counter, with the value 0
26	**NEWS ← A**	SCAMP-3	Copy the contents of array register A into the NEWS register
27	**A ← WEST**	SCAMP-3	Copy the contents of western neighbour's NEWS register into A register
28	**_add s0, 1**	Microprocessor	Increase the counter by 1
29	**_compare s0,10**	Microprocessor	Check to see relationship between counter and 10
30	**_jump c, 26**	Microprocessor	If counter < 10, the carry bit is set, so jump to line 26
31	**OUT A**	SCAMP-3	Counter = 10, so send the contents of array register A to host
32	…	…	…

Fig 4. An excerpt from a conventional image processing algorithm

Memory Address	Microprocessor Instruction	SCAMP-3 Instruction
23	…	…
24	`No Operation`	`A ← B`
25	`_load s0, 0`	`No Operation`
26	`No Operation`	`NEWS ← A`
27	`No Operation`	`A ← WEST`
28	`_add s0, 1`	`No Operation`
29	`_compare s0, 10`	`No Operation`
30	`_jump c, 26`	`No Operation`
31	`No Operation`	`OUT A`
32	…	…

Fig 5. A representation of the dual instruction streams, when loaded into memory

Memory Address	Microprocessor Instruction	SCAMP3 Instruction
23	…	…
24	`_load s0, 0`	`A ← B`
25	`_add s0, 1`	`NEWS ← A`
26	`_compare s0, 10`	`A ← WEST`
27	`_jump c, 25`	`No Operation`
28	`No Operation`	`OUT A`
29	…	…

Fig 6. An optimised dual instruction stream program

stream of instructions, whilst simultaneously (via its program counter, which points to a memory location) the ICWs are delivered to the processor array.

The "dual instruction stream" architecture can be further illustrated with the aid of an example. The code excerpt in Figure 4 follows the single instruction stream approach to programming. The purpose of the excerpt is to take an image stored in array register 'B', shift it 10 pixels to the right and send the resulting image back to the host. This is achieved by copying the data register 'B' into the 'NEWS' register (making it available to direct neighbours) and then replacing the data register's contents with its western neighbour's 'NEWS' register value. As every APE executes the same instruction, the effect is to shift the entire image to the right, one column at a time. A loop is implemented that repeats this operation 10 times.

This style of programming results in both processors (the microprocessor and the SCAMP-3 chip) having to perform "No Operation" instructions, shown in Figure 5. This is undesirable as it increases the size of the algorithm, and slows down the execution speed of image processing. A more optimal solution is to execute two instructions in parallel. The sequencer allows this due to the separate nature of the instruction streams. Figure 6 illustrates how this can be done, almost halving the program size, and better utilizing both processors, without changing the outcome of the algorithm.

IV. SCAMP SYSTEM DIAGRAM

The SCAMP Sequencer is solely responsible for issuing ICWs to the processor array. The system overall is controlled by a second 8-bit embedded microprocessor, termed the System Controller, which executes a program called "Operating System" (OS). When the controller is started, it executes a small pre-loaded program that loads the OS, which is delivered to the system by the host. This provides a flexible system which is easy to modify and customize via firmware updates. The System Controller is responsible for maintaining the Communications Interface, configuring analogue hardware, controlling the SCAMP Sequencer and providing additional control over the SCAMP chip, including reading out data from the processor array. The Communications Interface allows data to be transferred either by USB2.0 or a 12-bit parallel

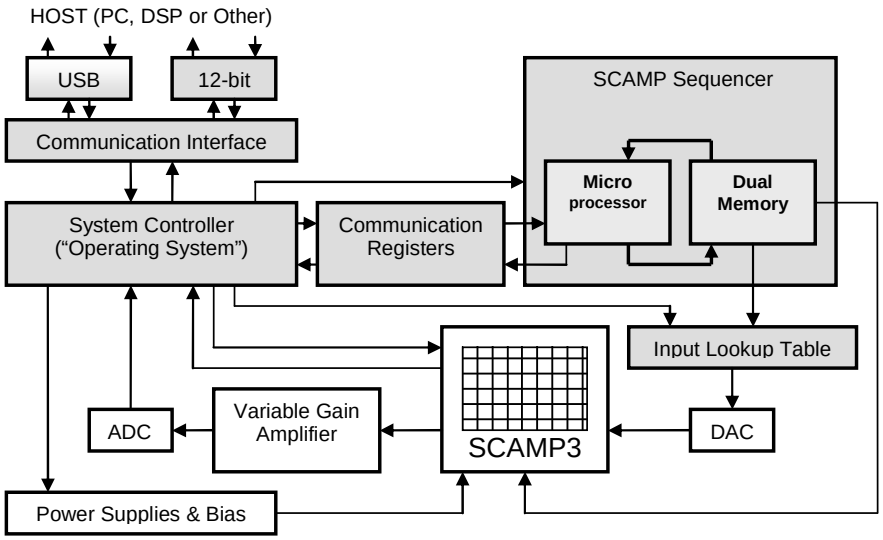

Fig 7. A functional overview of the system. Shaded components are implemented on an FPGA.

connection. Take for example the downloading of a SCAMP algorithm to the system. The host sends a packet to the system containing the image processing algorithm. The System Controller stops any currently executing SCAMP algorithm, by stopping the SCAMP Sequencer. It then fills the dual-stream memory and resets the SCAMP Sequencer, allowing it to start executing. Another example is when the host configures the analogue hardware. A packet is sent containing the configuration data, which is interpreted and distributed by the System Controller. The most important feature of the System Controller is its ability to read data out from the array. For example, when a SCAMP algorithm makes a request to send data to the host, the System Controller intervenes, takes control over the SCAMP chip and acquires the relevant data. The System Controller can perform flexible read-out of the array [7] either by scanning the array to access individual pixels or by defining regions of pixels. Different operations can be performed on selected regions, such as summation of array values, or performing digital OR operations. The System Controller can also communicate directly with the SCAMP algorithm, and vice-versa, via a set of Communication Registers. For example, it is possible for the SCAMP algorithm running on the SCAMP Sequencer to request a read-out action by the System Controller, which then places the read-out data in a Communication Register, subsequently read in by the SCAMP Sequencer. This allows feedback into the image-processing algorithms. This mechanism also permits the host to send parameters to the algorithms whilst they are executing.

The SCAMP chip can return analogue and digital data from the processor array. The system is capable of reading out different forms of data to the host system. These include full and windowed ("region of interest") analogue frames, full and windowed digital frames in two ways (1 pixel per byte, 8 pixels per byte) and global results such as identifiable scalars and pixel coordinates. All image output can be read out from different regions of the processor array, at varying resolutions. Each transfer can be tagged with an identifier to allow the host to differentiate it when an algorithm returns multiple data types. The frame-rate of the system can be set by an internal pulse which is user-configured or externally triggered. If the algorithm requests, the System Controller will suspend the SCAMP Sequencer from further execution until the pulse has occurred. The flexible nature of the System Controller allows addition of custom functionality, enabling additional processing to be carried out away from the SCAMP algorithm. This could be used to customize data read-out from the array or for example, perform pixel searches and return coordinates [7].

Fig 8. SCAMP3 vision system; a Spartan 3 FPGA daughterboard is fitted to the underside of the PCB

V. SYSTEM IMPLEMENTATION

The vision system is shown in Figure 8. The SCAMP sequencer, OS controller and all additional digital logic are implemented on a Xilinx Spartan3 XC3S400 FPGA. A third party development board [8] houses both the FPGA and the USB communication hardware. The OS controller and SCAMP sequencer both use Xilinx PicoBlaze soft microprocessors [9]. The entire digital design uses just 15% of the slice registers available, 62% of block ram and 6% of available multipliers.

From a hardware perspective, the most critical feature is the interface for the SCAMP-3's analogue readout. SCAMP-3 permits the aggregation during readout of the contents of a register across a pixel group within the array. Such aggregation may be between 1 (no summing) and 16384 (the entire array). Hence, SCAMP3's current mode readout requires an analogue front-end capable of handling variations of 84dB in peak signal. In addition, a signal to noise ratio of at least 48dB is required (i.e. 1 bit in an 8 bit system) under all readout conditions, with single pixel readout of peak signals of +/- 1.7µA clearly being the most demanding. In summary, a dynamic window of 132dB is required within a system reading out analogue data at >1MSample/s. This was fulfilled by means of 3 layers of variable gain as shown in Figure 9. Two front-end amplifiers are incorporated with the input switched between the two according to the gain required. This provides a dedicated amplifier for single pixel readout with the maximum possible front-end gain. A separate amplifier provides gain for the less demanding instances where higher numbers of pixels (with higher currents) are read. Within the feedback path of this amplifier, a second level of variable gain is supplied. Finally, the third level of gain is provided by a variable gain amplifier with gain between -14dB and +34dB to scale the output across the 2V range of the ADC.

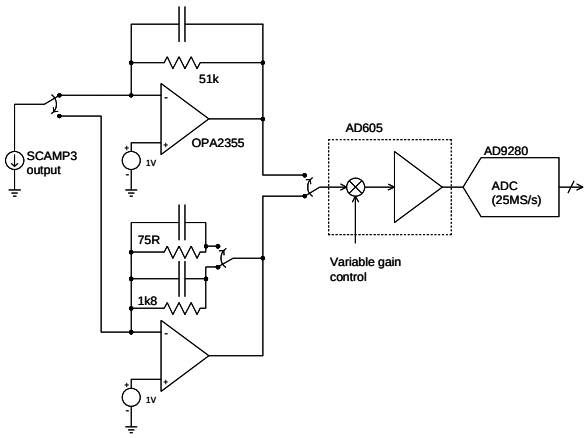

Fig 9. Schematic of Variable Gain Amplifier

A significant noise source of the readout system is the contribution from amplifier voltage noise. This noise contribution is proportional to the sum of the amplifiers input capacitance [10]; this includes the capacitance added by the current source and the analogue switch. A theoretical analysis of this noise contribution reveals that it alone is predicted to contribute 0.5bits to noise (rms). Hence, eight samples are taken for every pixel value stored reducing the effect to 0.18 bits of noise. Overall, the measured output noise level of the system is around 0.24 bits.

Within the current system, analogue readout is currently limited to a maximum speed of 37 frames per second (fps). Readout of binary images is possible at up to 434 fps. With a dedicated firmware implementation on the FPGA, substantial improvements in frame rate should be feasible.

To input analogue data to SCAMP-3's APEs, a 12-bit parallel DAC is provided. This provides a means of loading an analogue value (in reality a current) to a register. Since the input to SCAMP-3 defining the analogue value is non-linear, a look-up-table is incorporated that ensures the linear 8-bit digital scale within the system controller is correlated to a linear analogue range within SCAMP-3. This look-up-table (Figure 7) is created automatically during an OS calibration routine at start-up.

This system is designed to be powered from a USB2.0 interface (drawing 420mA). A switched power supply efficiently converts 5V to the 3.3V required for the Xilinx Spartan-3 and SCAMP-3 ICs. Additionally, 23 digital to analogue converters are incorporated to provide software configuration of bias voltages and supplies to the SCAMP IC and its system. The SCAMP-3 IC itself draws a maximum of 100mA, with typical algorithms (such as those described in this paper) consuming <5mA.

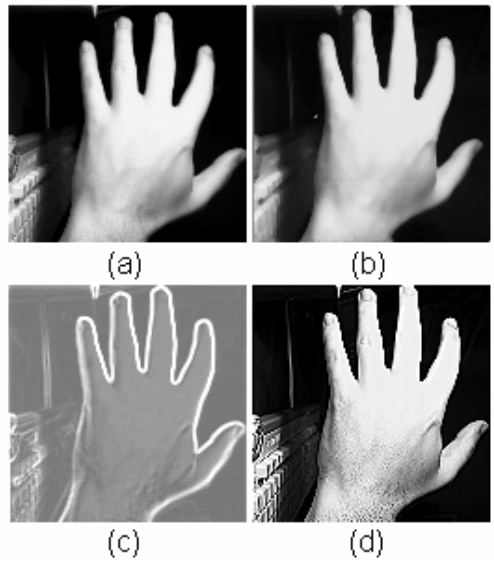

Fig 10. SCAMP only image processing. A selection of commonly used filters is shown: (a) No Filter, (b) Median Filter, (c) Sobel Filter, (d) Sharpen Filter.

VI. SOFTWARE TOOLS

Software is the primary development platform for image processing algorithms. It was anticipated the user will want to use the SCAMP system in their own applications; therefore a set of drivers and an Applications Programming Interface (API) were built. This API encapsulates all of the configuration and low-level communication, abstracting it away from the programmer. The API adheres to the ANSI C++ standard and is available for both LINUX and WINDOWS platforms. There is also software available to replace the host computer with a DSP.

A development tool 'ScampGUI' has been developed using the API which provides a means to configure, test and inspect almost all the system. ScampGUI provides frame and statistic viewers, performs noise calculations and records movies. A second tool 'ScampSIM', is an integrated development environment for creating image processing algorithms that run on the SCAMP system. ScampSIM provides a detailed behavioural simulation of the analogue SCAMP processors. ScampGUI and ScampSIM can be combined to create a "live" development environment, where the changes made in ScampSIM are downloaded to the system, providing real-time feedback. This reduces development time considerably.

VII. EXPERIMENTAL RESULTS

The examples presented in this section demonstrate different features of the system as a whole to perform image processing. The first example is SCAMP only image processing. Figure 10 shows the system returning full frames. The algorithms shown there use only the basic communication capability of the system, i.e. the SCAMP chip is performing the entire algorithm, using the default system settings. No additional input or feedback is being used.

The next example, Figure 11, illustrates how the user can interact with an executing algorithm. The algorithm performs a simple binary threshold and sends digital frames to the host. The threshold level is set via a slider control in the host software. The host sends a packet to the System Controller, which asynchronously updates a Communications Register between the System Controller and the SCAMP Sequencer. The algorithm samples this port to get the threshold value,

Fig 11. A digitally thresholded image. The threshold can be set externally, via the host.

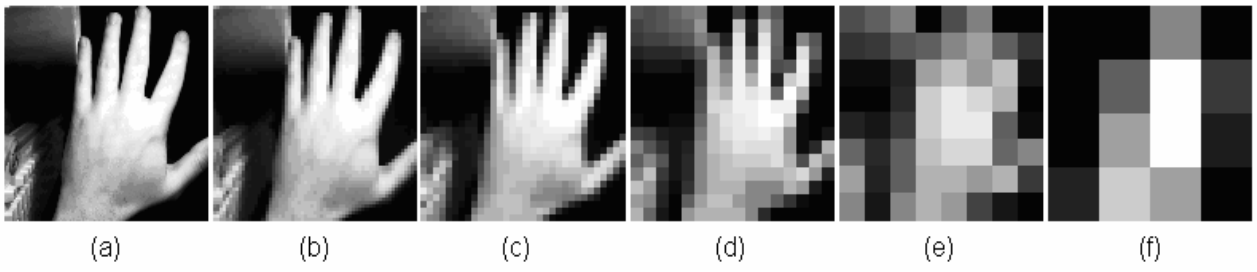

Fig 12. Multiple Resolution Readout. The control system is being configured to read out from the processor array chip with varying parameters. (a) 128x128, (b) 64x64, (c) 32x32, (d) 16x16, (e) 8x8, (f) 4x4

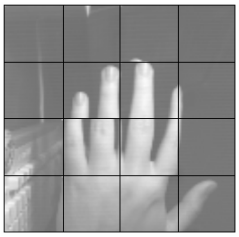

Fig 13. A Re-assembled image, comprising of regions that have changed the most

which is then inputted to the SCAMP-3 chip (via the look-up table).

Figure 12 shows how the System Controller can be used to control the SCAMP's read-out. The SCAMP chip simply captures the image and performs no additional processing. By changing the regional read-out size, and the amplifier gain, multi-resolution read-out is possible. Essentially this algorithm is performing pixel binning, grouping the regions pixels and returning their summation. This is useful when performance is critical due to the lower bandwidth requirements when returning lower resolution frames, or to obtain some global measures.

Figure 13 illustrates a number of the features in the system being used to implement a primitive compression algorithm. This algorithm is intended as a demonstration of the system's capabilities, rather than a practical compression algorithm. The algorithm returns in full resolution only the regions of the image that have changed the most from one frame to another. The SCAMP-3 chip calculates the absolute difference between two frames. This results in a map that has lighter areas for movement, and darker areas for stationary pixels. The SCAMP-3 chip does not perform any more processing at this point, and the System Controller is configured to read out regional summations, in this case similar to that in Figure 12(f), of the absolute difference map. These 16 readings are used to determine which region has changed most (i.e. has the highest overall sum of absolute difference). The System Controller is then configured to read-out in full resolution the selected region only. The host software receives these regional fragments once per frame and re-assembles an image by drawing the fragments in their correct locations. This gives the impression that the algorithm is returning full frames at full speed, but uses lower bandwidth.

In Figure 13, some discontinuities can be noticed, as the system is making a rough judgement to the area that has changed the most. A trade-off is made between accuracy and frame-rate. By increasing the resolution at which motion is detected and returning a number of smaller regions per frame, the discontinuities become less obvious, but the algorithm has more regions to compare when finding which one has changed the most.

VIII. CONCLUSIONS

This paper has presented a complete stand-alone image processing system based on the SCAMP-3 vision chip. The SCAMP-3 chip is used to reduce the processing time required, by processing pixels in parallel. This system consists of five major components; each one has been described, with particular attention paid to the SCAMP Sequencer component and its programming paradigm. These components form a system which is a flexible platform to develop and execute real-time computer vision algorithms for various applications, and is portable for use within embedded systems. Examples of the systems capabilities have been highlighted.

ACKNOWLEDGEMENTS

This work has been supported by the EPSRC; grant numbers: EP/D503213 and EP/D029759.

REFERENCES

[1] L. Carranza, et al., "ACE 16k based stand-alone system for real-time pre-processing tasks". VLSI Circuits and Systems II, May 9-11 2005
[2] A. Elouardi, et al., "A smart sensor-based vision system: Implementation and evaluation," Journal of Physics D: Applied Physics, vol. 39, pp. 1694-1705, 2006.
[3] N. Ouerhani, et al., "A real time implementation of the saliency-based model of visual attention on a SIMD architecture". Pattern Recognition. 24th DAGM Symposium. Proceedings, 16-18 Sept. 2002, pp. 282-9
[4] C. C. Wells, E. Duncan and D. Renshaw, "An FPGA based prototyping platform for imager-on-chip applications". Proceedings. 2004 IEEE International Conference on Field- Programmable Technology
[5] A. Zaràndy, "ACE box: High-performance visual computer based on the ACE4k analogic array processor chip," in Proc. Eur. Conf. Circuit Theory Design, Espoo, Finland, Aug. 28–31, 2001, pp. 361–364.
[6] P. Dudek, "Implementation of SIMD vision chip with 128x128 array of analogue processing elements", ISCAS 2005.
[7] P. Dudek, "A Flexible Global Readout Architecture for an Analogue Vision Chip", IEEE International Symposium on Circuits and Systems, ISCAS 2003, Bangkok, Thailand, vol.III, pp.782-785
[8] "XEM3001v2 Product Brief" http://www.opalkelly.com/
[9] "PicoBlaze Product Brief" http://www.xilinx.com
[10] J.G. Graeme, "Photodiode Amplifiers: Op-Amp Solutions", McGraw-Hill, New York 1995