IAPR Workshop on CV - Special Hardware and Industrial Applications OCT.12-14, 1988, Tokyo

TAGIPS, AN ADAPTABLE PARALLEL PROCESSOR FOR IMAGING APPLICATIONS

Pekka Hänninen, Jouko Viitanen, Juha Salo and Jarmo Takala

Tampere University of Technology
PO.BOX 527, 33101 Tampere, Finland

ABSTRACT

The new generation 32 bit signal processing chips have architectural features that make them suitable for image processing purposes. Although the speed of a processor board with one PE does not meet the requirements of real time imaging, with some additional circuitry it is possible to construct such a board. In this paper we present a fast parallel processor for a PC bus. It uses a single 32 bit signal processing chip connected together with special convolution processors and reconfigurable logic cell arrays.

INTRODUCTION

The parallelization of image processing tasks has been studied extensively. A large number of special parallel architectures have been developed within the last few years. In our research of practical image processing systems the emphasis has been especially on certain design criteria: low cost of the system is obvious, both for the end-user equipment, and for the equipment used in development laboratories designing industrial applications of image processing. This requirement no longer needs to contradict with another important requirement, high speed. Recent developments in VLSI allow the design of fast but low-cost systems for a wide application range. Ideally, the same equipment and programming tools that are used in developing a system, can be used in the final product.

The next criteria for an image processing system is an efficient programming environment. This requirement has traditionally been overlooked, dictated by the exotic special processor architectures used. A common approach is to offer a number of fixed routines and management tools for constructing the user 'program' using these routines. This approach offers a minimum amount of flexibility. We prefer more common development tools found in any low cost programming environment.

To satisfy these criteria, we opted to use standard DSP components with readily available development tools, and industry standard PCs as the host for the image processor with efficient user interfaces. Special processors were selected for the most computationally intensive operations found in image processing. Some important architectural features are sometimes ignored in image processing systems. A common approach is to have a general purpose computer connected to a fast accelerator processor. However, the high bandwidth requirement of image processing may not be met: the image frame transfer time may be a significant part of the processing time (I/O overhead usually is ignored in benchmark ratings!). This problem is emphasized if slow PC expansion busses are used. Therefore our design limits image data traffic to minimum: processing is done as much in place as possible. The key feature is the capability of the processor to store images to its local memory directly from a flash A/D converter, whereafter only the interesting parts are processed. When the transfer of a full image to or from the host is required the maximum speed of the bus is achieved by the use of direct memory access logic and the full data width of the bus. Our selection for the main processing element has a maximum data bandwidth of 2*33 Mbytes/s over its two busses that can be used simultaneously.

ARCHITECTURE OF TAGIPS

GENERAL

The architecture of TAGIPS image processor was designed to meet two goals: the circuitry had to be simple enough to fit on one PC board and the board had to be fast enough to support imaging routines at a speed of several frames per second as well as to capture the frames to be processed from incoming video signal or digital data. The programming of the system had to be simple and efficient.

The processor chip for TAGIPS was chosen to be Texas Instruments TMS320C30 /9/. With an instruction cycle of 60ns it is fast enough for storing images digitized from real time video. Also its internal features, such as a DMA- controller and fast serial busses support distributed processing capabilities. To enhance the power of the processor we added special cascadable convolution processors to the board. The speed of a single convolution processor is up to 10 million 32 point /16 bit convolutions per second. They can also be used for edge detection and FFTs.

Run-time reconfigurable logic cell array (RRLCA) devices were added to provide adaptable connections to external busses. In addition to bus interfacing these RRLCAs can be configured for various low level imaging tasks such as in the hierarchical chamfer matching (HCMA) method for (template) pattern matching /6/../8/, median operations in fir median hybrid (FMH) /1//2/ filters and thinning algorithms based on logical decisions. Some ideas of the implementation of these methods with our hardware are presented later in this paper.

The block diagram of TAGIPS processor is shown in fig 1.

CPU

The new 32 bit CPU, TMS 320C30 has a linear address space of 16 megawords. It is sufficient in most 2-D imaging tasks. Also some special features such as fast floating point arithmetic and internal DMA-controller come in handy in image processing. In the processor chip the internal processing units operate in parallel thus speeding up the throughput of the chip up to 33 MFLOPS.

The processor also contains two serial busses that can operate without any additional circuitry. The speed of a serial port is up to 16 Mbits/s, so they can be connected with suitable buffers to LANs or to other TAGIPS boards.

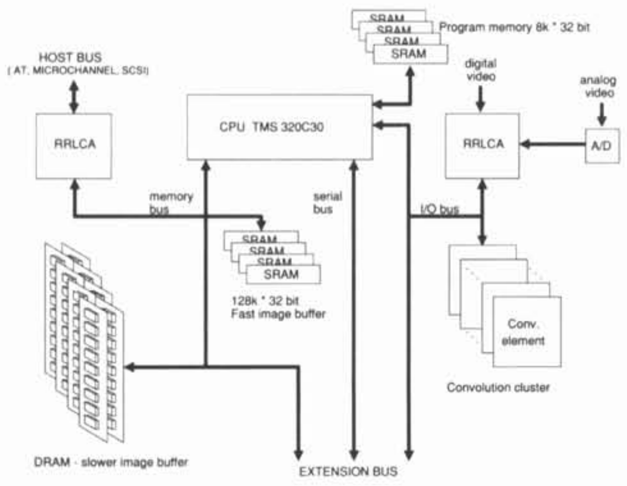

Figure 1. TAGIPS block diagram

Multiple TAGIPS boards can also be connected to common memories via the board's extension bus.

The program development tools for the processor contain a C-compiler provided by the manufacturer. That allows the user to write programs easily as well as convert already existing programs with minor changes to the new processor. Together with possibilites of writing assembly code within the C programs and a simulator / debugger, the program development is flexible.

In our project an interactive programming tool called IEDIT has also been constructed for the processor. IEDIT is an interpreter joined into a commonly known editor µE-MACS. The user interfaces to the processor by writing statements in C-like fashion and executing the statements with a control sequence within the editor. This way the development phase of methods and programs can be easily speeded up.

DIGITIZATION LOGIC

With its instruction cycle of 60 nanoseconds the processor can directly load pixels from digitized video through a flash A/D video converter on the board. When digitization/synchronization is performed under program control, the region of intrest (ROI) can be easily selected and thus the unneccessary storing of the whole image can be avoided.

The A/D converter is followed by some additional logic to pack the 8 bit pixels into one 32 bit machine word. This way only one read in about six machine cycles is necessary to keep up with synchronous digitization of 512 * 512 * 8 bit image with a line sweep time of 64 µs. The digitization can be performed with the internal DMA controller with minimal attention by the main processor. The program-controlled synchronization allows interfacing to various image sources with different video standards, and also to CCD cameras, and scanners.

CONVOLUTION PROCESSOR

Convolutions, correlations and related operations are very often used in the preprocessing of images. Although the TMS320C30 itself can perform convolutions quite well, the availability of simple-to-connect convolution processors made it possible to releave the main processor of the board from the burden of performing the very lowest level calculation intensive activities. Also such a coprocessor can be used for calculation of 2-D Fourier transform and its inverse, as well as to perform correlation calculations in gray scale pattern matching, and edge detection. The processor board contains an extension connector for several cascaded convolution processors.

An application example for convolution processors is given later in chapter:
'Application development for TAGIPS'.

RRLCAs

Run-time reconfigurable logic cell array (RRLCA) /12/ devices have made it possible to find adaptable hardware level solutions to calculation intensive tasks. Also RRLCAs can be used to save space on a processor board when one reconfigurable component can serve in a place of several field programmable logic arrays (PLA, PAL).

In TAGIPS the RRLCAs are mainly used in simplification of the board design: For example RRLCA is an ideal component in defining the connections to external busses and devices. When we connect to different devices the only thing on the board that needs to be changed is the 'program' for the RRLCA. This is performed dynamically in a couple of milliseconds with the internal DMA-controller of the TMS chip in the case of TAGIPS.

RRLCAs can also be used as reconfigurable processors in such calculation intensive tasks that are simple enough to be fitted into the hardware of the onboard RRLCAs.

Dynamically programmable RRLCAs are a newly introduced type of VLSI circuitry. They are targeted to areas where field programmable logic arrays have been used. Their use as reconfigurable computing blocks is an area that has not been studied until quite recently. So far, the capacity of the available circuits only allows small routines to be fitted in hardware. Image preprocessing tasks commonly are quite simple being continuously repeated over large amount of data before any changes in circuitry might be needed; thus the transfer of the 'program' requiring couple of milliseconds in the beginning of the repeat sequence does not present a problem. Also most of the preprocessing algorithms that are used in image processing consist of same type of operations thus allowing the functions for RRLCAs to be written in a library.

When RRLCAs become more efficient in programmability and configuration speed they can be expected to present a major part in the development phase of the computing equipment. These new kind of dynamically gate-level 'micro'programmable computers will undoubtly be very effective in many areas where calculation intensive tasks are processed although at present time their use is limited to simple but often repeated structures. The existing silicon compilers that use a high level language as their hardware description language (for instance Pascal in /11/) could be used for programming code development for them; however, as larger programs can be fitted in, besides the actual data processing operations, also the automatic synthesis of the control structure of the implementation must be performed. Asynchronous ones, like in /11/ the extended timed Petri net structures seem to be good candidates for the control structure.

APPLICATION DEVELOPMENT FOR TAGIPS

Although many of the programs written for earlier SVAI and TAMIPS /3/../5/ image processors can simply be transferred to TAGIPS, the new architecture will also require further development of methods and programs to take full advantage of the board's capabilities. Application development for pattern matching with hierarchical structures using RRLCAs have been reported in /8/. Furthermore RRLCA:s can be used even more efficiently in sorting data items as needed for median hybrid filters reported in /1//2/. Also the use of convolution processors parallelly with other simultaneously executing units of TAGIPS requires more work.

PROGRAMMING THE CONVOLUTION PROCESSOR

The use of the convolution processors allow parallel execution of several different levels of imaging tasks: convolution elements can each be configured for separate imaging routines to be run parallelly. For example one part of convolution processors could be performing FIR type of convolution calculations while other parts of the cluster could be employed in edge detection and correlation calculations. The main processor of the board could concurrently employ its DMA controller to image digitization as well as to the output of the calculation results to the host bus; the processor could furthermore employ its ALU in a higher level imaging task such as pattern recognition.

Simple convolutions up to 16 by 16 mask sizes can be performed at the speed of 20 Megapixels (16 bit pixels) per second divided by the square root of the amount of mask pixels. Thus performing a simple 3 by 3 convolution with a single convolution element to a 512 by 512 image will take approximately 40 ms being equal to one full frame time with interlaced 50 Hz video signal. Since there are eight convolution elements in the system , the convolutions up to a mask size of 64 points of 8 bits could be performed with eight parallel executing processing elements thus speeding up the convolutions with a factor of eight. This is not always possible, since the I/O will present a bottleneck after more than two simultaneously executing elements are added when execution is performed with 4 bit coefficients. With sixteen bit coefficients all eight convolution elements can be involved in separate tasks without I/O overloading.

The convolution element can also be used for performing 2D Fourier transforms. Two algorithms, namely the Prime Number Transform (PNT) and the Chirp-Z Transform /10/ can be used for that purpose. Although the processor itself is still under construction/testing, it can be estimated that the transform of 512 by 512 image can be performed in around 250 ms.

GRAY LEVEL PATTERN MATCHING

With fast correlation calculation in the convolution processor of TAGIPS, significant improvements in execution speed of the hierarchical chamfer matching method (HCMA) can be achieved. HCMA is a method for speeding up template-matching -type of operations, detailed descriptions of which are in /6/../8/. With the convolver, the binarization phase of the image can be discarded as well as the calculation of the cost of each pixel for pattern matching. Instead the matching is performed directly with the gray-scale image using the more common cross-correlation techniques of template matching.

Matching patterns in a gray-scale image presents two serious problems: the cost of calculation is tremendous and the lighting of the scene presents unpredictable changes. However, the cost of the calculation can be reduced with hierarchical structures based on increasing resolution of the image being matched, and slow spatial changes in the 'DC' component due to lighting of the image can be reduced with simple edge detecting masks.

The matching procedure starts with a low resolution scene and moves towards higher resolution when the ROI has been found. This approach is suitable for TAGIPS since the digitization of a real image can be controlled and the ROI can be spotted in digitization thus improving the performance and lowering the need for memory space. The actual matching is performed on model points that are taken from circular paths of the model. This scanning method is useful in determining the rotation angle of the object at the same time as the translational position is searched for. The method also allows another speedup in calculation since dissatisfactory results from inner scan- paths result in discarding the current scan position. Convolution processors have two coefficient banks so the next model point circle can be loaded with the DMA-controller at the same time as calculation of an inner path is at hand. This way the time needed for the transfer to the next scanning level can be reduced.

We can give rough estimates from simulations and previous experiences of the execution speed of TAGIPS. For gray scale pattern matching the estimated calculation time for a scene with a size of 256 by 256 pixels and three levels of a pyramid (quadtree, 256*256, 128*128 and 64*64 at the other levels) with 40 valid model points and no inner circle exclusion is about 10 ms. The total trial counts for model positions at each pyramid level starting from the lowest level and moving towards ROI are 256, 128 and 36. (These parameters are identical to those presented in /8/) The time estimate does not count the time needed for the digitization and high-pass filtering of the image since they can be executed almost simultaneosly. The number of model points in this example is quite low, but the inner circle exclusion will speed up the execution with larger number of model points. Thus positive recognition can be carried out with a delay of a few full video frames.

PROGRAMMING THE RRLCAs

The program development system for the logic cell arrays that we have used so far, supports manual editing of the interconnection network and the logic equations of the programmable logic blocks; a silicon compiler - type of integrated system is still due to come. The capacity of the 100 block RRLCA allows small tasks to be implemented in hardware. Most appealing ones are those that are slow to execute on a sequential Von Neumann processor: simultaneous comparisons of several words (finding minimum, maximum, median), bit-serial, word-parallel operations, fast asynchronous iteration loops, and operations that can be pipelined (successive logical operations, additions etc. without intermediate accumulator storing). Our example is the calculation of a hybrid FIR-median filter.

CALCULATION OF A FIVE POINT MEDIAN WITH RRLCA

Median based hybrid filters are good in filtering noisy images./1//2/ These filters offer less costly way to filtering compared to true median filters. These FMH filters (Fir Median Hybrid) work usually with small masks of sizes 5 by 5 or 3 by 3. Calculation starts at small area averages (usually 1-3 points) continuing to calculation of median of these averages and the center pixel. This way the element count in median is usually reduced to five. With small number of points traditional bubble sort has been found most effective in sorting the elements for median calculation. Although the pixel count is only five, TMS320C30 requires several tens of instruction cycles to sort them due to slow branching instruction.

Implementing the median with RRLCAs is most effecively done with a modified method of calculating the median; Median is calculated for each bit level of the pixels being processed and the non-median values are forced to maximum or minimum throughout all the bits of the word depending on the median value. The procedure is repeated for each bit level and finally the median is the one(s) that has not been rejected. In RRLCAs the calculation can be thus conducted in N clock cycles when the values are N bit pixels. This median calculation method is presented in figure 2.

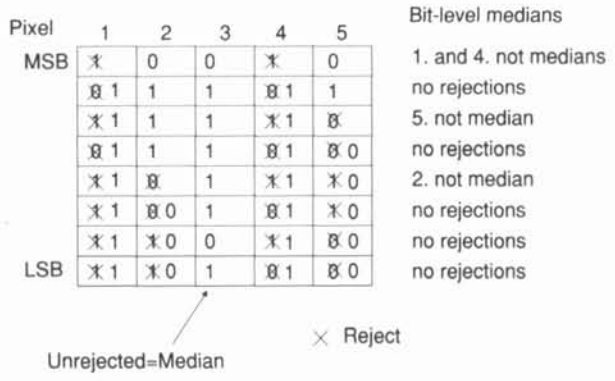

Figure 2. Calculation of a five point median

CONCLUSION

The requirement for speed and low cost need not to be contradictory in designing a multi-purpose image processor. We have shown that in user friendly PC-environment it is possible to program a high speed and yet low cost add-on image processor with low programming effort and hardware knowledge from the user. Furthermore such a board can be constructed to meet the requirements based on high speed imaging hardware, and with good understanding of the hardware it is possible to construct application routines capable of executing tasks far beyond the scope of any current low cost image processing system. In TAGIPS image processing system the employment of special convolution prosessors and reconfigurable logic blocks as well as the full employment of the main processor of the board will lead to maximum performance of several GOPS. This speed is achieved via parallelization of the different levels in imaging task.

REFERENCES

1. P. Heinonen, Y. Neuvo: New Median Type Filters for Image Processing. ISCAS'85 Conference, Kyoto, Japan, June 5-7 1985.

2. A. Nieminen, P. Heinonen, Y. Neuvo: A New Class of Detal-Preserving Filters for Image Processing. IEEE PAMI, Vol. PAMI-9, no.1 January 1987.

3. J.Viitanen, P.Vänni: The TAMIPS Multiprocessor, International Conference on Parallel Processing, Chicago, August 1985.

4. J. Viitanen, O. Pulkkinen: A Fast Digital Image Processor, Computers And Artificial Intelligence, Vol 5, no.1, 1986.

5. J. Viitanen, J. Salo, P. Vänni, P. Hänninen and G. Campbell: Developing Applications for The Parallel TAMIPS Image Processor, The 8th International Conference on Pattern Recognition, Paris, France, October 28-31 1986.

6. J. Viitanen, P. Hänninen, R. Saarela and J. Saarinen: Hierarchical Pattern Matching with Efficient Method for Estimating Rotations. IEE and SPIE conference IECON'87, Cambridge, Massachusetts, USA, November 3-6 1987.

7. J. Viitanen, P. Hänninen, R. Saarela and J. Saarinen: An Efficbent method for Image Pattern Matching. International Conference on Parallel Processing for Computer Vision and Display, Leeds, U.K., January 12-15 1988.

8. J. Viitanen: Image Pattern Recognition Using Run-time Reconfigurable Processor Architecture. Submitted to ICCV'88, Second International Conference on Computer Vision, Tarpon Springs, Florida, USA, December 5-8 1988.

9. Ray Simar and Mike Hanes, Texas Instruments Inc. : CMOS DSP Chip Packs Punch of a Supercomputer, Electronic Design, March 19, 1987.

10. Hossein Yassaie: Discrete Fourier Transform with the IMS A100, IMS A100 Application Note 2.

11. Krzysztof Kuchcinsiki and Zebo Peng: Parallelism Extraction from Sequential Programs for VLSI Applications, Microprocessing and Microprogramming 23 (1988) pp. 87-92, North-Holland.

12. Programmable gate arrays, Product data, Xilinx Inc. U.S.A. 1986.