ASSOCIATIVE APPROACH TO REAL TIME COLOR, MOTION AND STEREO VISION

Avidan Akerib and Rutie Adar
A.S.P. Solutions Ltd.
Robomatix Headquarters,
Hataasiya St., P.O.B. 2092, Industrial Zone,
Raanana 43100, Israel

ABSTRACT

This article presents a new methodology, based on the Associative Signal Processing (A.S.P) approach to real time parallel image processing. The architecture is fully programmable and can be programmed to implement a wide range of color image processing, computer vision and multi-media algorithms at much faster than video rate. The approach is based on an array of thousands of processors, each is nothing but an "intelligent" memory word that can identify itself to a value and change its content accordingly. Benchmark results show that when assigning an "intelligent" word (processor) to each pixel in the image, computational power of several hundred billion instructions per second is obtained. A chip based on this approach was developed by A.S.P Solutions Ltd. The chip called **XIUM™** includes 1024 processors, each with 72 "intelligent" bits, has computational power of 1 BIPS and cloud identifies at a rate of 20 billion patterns per second. A commercial chip with 2 BIPS performance - The **XIUM™-II** , is now on the final stage of development.

INTRODUCTION

Real time vision involves interpretation of a changing scene that must be updated as video frames arrive at a rate of 30 per second. Each video frame contains an average of half million pixels. The amount of data for each pixel may vary from 8 bits (monocular monochrome image) to 32 bits, for tasks dealing with stereo color images, or 32 bit color separations algorithms for high dense color printers.

Carrying out vision tasks while keeping up with the input scan rates, requires enormous computation power. *Rosenfeld and Weems[1]* estimated it at 100 billion instructions per second, plus or minus two orders of magnitude. Although there are many vision tasks that do not require such frequent updating, a very high computation rate is still required. Within the last decade many architectures have been developed , in system level (*Connection Machine[2]*) or in chip level (The Multi-Media Vision Processor- MVP[3]). All are based on parallelism of serial computers (or DSPs) i.e. increasing performance by partitioning the memory to blocks and assigning each block to a different processors or DSP. The major disadvantage of this approach is the price and power consumption. Hence, specific applicationsare still not commercially to be developed by general purpose components.

In the following, there is a presentation of breakthrough technology, developed by A.S.P. Solutions Ltd. The technology is based on the associative solution that meets the above requirements with low power consumption and low price. The A.S.P. architecture carries out any vision algorithm at faster than video rate, providing for the first time economy solution for special purpose applications with general purpose architecture.

WHAT IS ASSOCIATIVE PROCESSING?

The Classical approach to associative processing, as described by *Foster[4]*, uses a few primitives to implement all arithmetic and logic operations. These functions operate on all bits of all words of associative memory at the same time. *Akerib & Ruhman[5]* adapted this approach to computer vision and proposed a system designated ARTVM- Associative Real Time Vision Machine.

To understand the associative approach, let us assume that pupils are sitting randomly in a classroom. The teacher wants to change the seating arrangements so that the tall pupils are sitting at the back and the short pupils are sitting in the front rows. Using the classical serial method, the teacher would have to turn to each individual pupil and ask him/her to change their place, if necessary. Using the associative method, the teacher asks all the pupils at the same time the following question:

"Who has difficulty seeing?"

After the appropriate pupils raise their hand, the teacher will give the following instruction:

"All the pupils who have raised their hands should change places with those pupils who are sitting in front of them"

Once they have changed places the teacher will again ask the question and give the same instructions over and over again until no hands are raised.

This example shows that the problem was solved through a sequence of questions and answers. Repeating this process until the convergence condition is satisfied, is the basis for associative processing. The time complexity is not dependent on the amount of data. (The student in the class). On the other hand it depends on some physical measure that is specific to the required results. The complexity of the above example is the distance between the taller and the shorter student.

0-7803-2431-5/95 $4.00

A deeper understanding of the associative approach is given by the following example:
Suppose an image size of 512 X 512 pixels, each of 8 bit lengths, is stored in the memory. The required operation is to find in 32 bit precision the exponent of each pixel (e^{-p} where p is 8 bit pixel value). In order to implement the task with a serial computer, the CPU has to read every pixel, execute the operation (through a table-look-up or mathematical co-processor) and then write the results into a 32 bit frame buffer. This processes takes at least 262,144 machine cycles

The ASP (Associative Signal Processing) approach solves the problem in a different way. If the pixels will be stored in an "intelligent" memory that can identify itself, then a control unit (a teacher) will carry out a sequence of questions and answers (without accessing the memory) as follows:

FOR I=0 TO 255
all the pixels with a gray level value of I
have to raise a flag
all the pixels which raise a flag will be
changed to OUTPUT[I]
END

It is easy to see from this example, that the computation time is not dependent on the amount of pixels. In any case 256 questions are asked and 256 answers are given. The questions and answers are carried out simultaneously to all pixels. Hence the problem was solved by 512 machine cycles (instead of 262,144). If we assume a clock rate of 20 MHz, the above algorithm will be accomplished in 25 micro seconds for all the 512 X 512 pixels. Hence the algorithm executes at a rate of 10 billion floating point operations per second .It should be noted that every question or answer to the "intelligent" memory occurs concurrently and takes only one machine cycle. This is the major property of associative processing, enabling it to implement any parallel truth table that is the building block for all the arithmetic and logic operations.

THE XIUM™ CHIP ARCHITECTURE

Referring to Figure 1, the core of the **XIUM™** chip is an array of "intelligent" (associative) cells, with 1024 words, each of 72 bits, that are parallel by bit as well as by word. The main command is **compare**. The *CW* (Comparand/Write) register is matched against all words of memory simultaneously and agreement is indicated by setting the corresponding *Tag* bit. The **write** command operates in a similar manner. The contents of the *CW* are simultaneously written into all words indicated by the *Tag*. The **read** command is normally used to bring out a single word, the one pointed to by the *Tag*.

It is easy to see that the **compare** and **write** commands are of the type:

IF *condition* **THEN** *action*,

that is well suited for parallel truth table implementation. This property facilitates all logical and arithmetic functions to all the 1024 words in parallel. Hence the **XIUM™** chip may be regarded as an array of 1024 simple processors, one for each word in the "intelligent" memory.

Communication between processors is carried out a bit at a time via the *Tag* register by means of the **ShiftUp(n) ShiftDown(n)** commands. The number of shifts (n) applied determines the distance or relation between source and destination. When this relation is uniform, communication between all processors is simultaneous. Fortunately, most vision algorithms, including operations over a neighborhood, only require a uniform communication pattern.

Bringing digital data in to or out of the processors, is available through the FIFO, which consists of a 24-bit shift register attached to each word. A color image (24 bits) or stereo image (16 bits) can be shifted into the FIFO as it is received and digitized, without interfering with associative processing. When the FIFO is full or when there is an interrupt (such as vertical blanking), the frame is transferred into the processors, a bit slice at a time, (48 cycles) via the *Tag* register. During the next frame time, both input and output proceed in parallel with processing without interference.

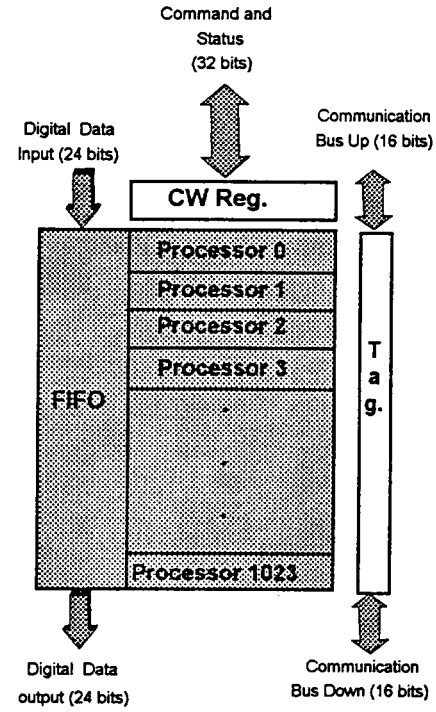

Fig. 1. XIUM™ block diagram

THE B.L.AS.T.™ ARCHITECTURE

The **B.L.AS.T.™** (Bit Logic Associative Technology) is a single board PCI Bus compatible (under development). The board includes 8 **XIUM™-II** chips for varies standard video compression (full CODEC) applications. Since the board is fully programmable, it could be also used for wide range of real time image processing applications that require a massive amount of operations. (up to 16 BIPS), such as color inspection, 3D measurement, virtual reality, motion detection, 3D graphics, high resolution animation, automatic morphing and desk top publishing accelerator. The **B.L.AS.T.™** is divided into three parts: a) the video unit, b) the processing unit and c) the controller unit. **The video unit** is to include a triple video digitizer, and a triple video display (24 bit) used for digitizing or displaying real time color TV images (PAL/NTSC). At the input section there will be a MUX to select the type of image to be accepted - PAL/NTSC or to direct 24 bit digital image. The memory storage used to store a high resolution, full frame, 24 bits. **The processing unit** will include 8 **XIUM™-II** chips. Since every chip includes 1024 processors, it makes available to process 8K pixels in parallel. The pixels to be processed are selected from the frame buffer through a programmable region of interest selector. The frame storage has two inputs and two outputs. One input to accept a new image from the camera and the second to accept the results from the processing unit. One output goes to the display monitor while the second output goes to the processing unit. All inputs and outputs can be located at different regions in the frame storage. **The controller unit** will include a program memory to store all the associative commands (questions and answers) and a micro controller to manage the program memory and to communicate with the host (PC) computer.

ARITHMETIC IMPLEMENTATION

Referring to Fig. 2, suppose that the **XIUM™** "intelligent" cells (1024 words each of 72 bits) contain two data vectors, **A** and **B**, each 1024 numbers long and with a precision of **M** bits. We wish to replace the vector field **B** by the sum **A + B**. The ASP operations are carried out sequentially, a bit slice at a time, starting with the least significant bit. In each step, the three slices $\mathbf{A_i}$, $\mathbf{B_i}$ and $\mathbf{C_v}$ (i=0,1,...,M-1, and $\mathbf{C_v}$ is the carry slice) are asked in parallel for an input combination of the applicable truth table and is followed by a parallel replacement of **B** and **C** with an output combination. The addition takes only **8M** machine cycles for all the data It follows that for a machine cycle of 50 nanoseconds (clock frequency of 20 MHz) an 8-bit addition in the **XIUM™** will takes 3.2 microsecond or 3.2 nanosecond per word. The ASP architecture is flexible such that a 4-bit addition will take only 1.6 ns. per word, and an increment of 1 will take only 0.4 ns. per word.

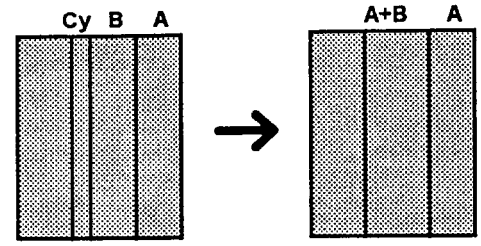

Fig. 2. Vector addition

Associative subtraction operates on the same principle and also takes **8M** cycles. It is easy to extend addition to **multiplication ($5.25M^2$+M cycles)** and **division ($17M^2$+13M+1 cycles).**

High level implementation ("C" language) for associative vector addition, is very simple. The statement looks like this:

```
VECTOR A(0,8), B(8,8), C(16,32), F(60,1);
C = A + B;
F = A==B;   /* mark in F all pixels which
          have the same value in A and B */
C = A+ShiftUp(B,1) ;   /* Vector B is
          shifted up 1 position and added to A*/
F = A>128;   /* thresholding (all values
          above 128 are marked in F) */
A+=5;          /* add a constant to a vector */
B++;           /* increment a vector */
C+=A*B ;       /* multiply accumulate */
```

The declaration vector (using class in C++) lets the compiler perform the operation "+" on vectors. When the compiler recognizes the expression "+", it calls the ASP associative addition function, to execute this operation associatively. The first parameter of the vector is the bit starting position in the associative processor within the 72 bits, while the second parameter is the precision. In this case the vector A starts in bit position A with a precision of 8. The vector length is determined by the number of processors in the chip. For example, in one **XIUM™** chip the length is 1024, and in the **B.L.AS.T.™** computer, the length is 8K. The execution time remains the same in all cases. Using the same principle, the programmer can use all the expressions and statements of regular "C" language.

VISION ALGORITHMS

The flexibility and speed of the **XIUM™** and **B.L.AS.T.™** allow the implementation of a broad range of vision functions in real-time. They include low level algorithms such as histogram generation, color convolution, edge detection, thinning, stereo matching and motion estimation. Mid-level functions can also be implemented, including morphological operations, contour tracing and labeling, Hough transforms, saliency mapping, and such geometric tasks as the convex hull and Voronoi diagram. The ASP simulator was used to test the associative algorithms and to verify their complexity. The tested was for a Single **XIUM™-II**

chip. Since the associative architecture is scalable, the performance for **B.L.AS.T.™** board (8 chips) will be exactly 8 times better. 5X5 convolution takes 45 ns./pixel., (about 15ms. for full screen) Edge detection is executed in 35 ns./pixel, Likewise, computation of stereo disparity by the *Grimson[6]* method over a range of ±15 pixels, including disambiguation and out-of-range test, is completed in 230 ns./pixel. This stereo performance was only attained by virtue of an array algorithm for counting labeled pixels over a neighborhood. Motion estimation for S.I.F. resolution and searching region of 16 ±7 pixels takes 30 ms. Curve propagation, thinning and contour tracing take 4 ns./pixel per iteration. The linear Hough transform takes 50 ns./pixel for a resolution of 16 in direction and distance from the origin. DCT and Quantization takes 14 micro second per 8X8 block.

PERFORMANCE EVALUATION

Two methods were selected for comparative evaluation of the **B.L.AS.T.™** performance. In the first method we compared our architecture to an SIMD array of up to 256 high performance processors (Inmos T800/Intel 860), and found the **B.L.AS.T.™** to have a speed advantage of 4.5 orders of magnitude. The speed advantage was lowest for neighborhood arithmetic operations of higher precision, such as convolution (factor of 97), and reached a peak for neighborhood logic operations such as curve propagation (factor of 2500). The second method was the Abingdon cross benchmark for which test results were available on several of the better known vision architectures. The **B.L.AS.T.™** was found to lead by 3.8 orders of magnitude in price-performance.

REFERENCES

[1] C. Weems, E. Riseman, A. Hanson, A. Rosenfeld, "*IU Parallel Processing Benchmark*", Proc. Comp. Vision &Pattern Recogn.pp 673-688, 1988.

[2] W.D. Hillis, "*The Connection Machine*", MIT Press, 1985.

[3] Texas Instruments, "*Multimedia Video Processor MVP*", Technical Brief, 1994.

[4] C. C. Foster, "*Content Addressable Parallel Processors*", Van Nostrand Reinhold Co., 1976, chs. 2 & 5.

[5] A.J. Akerib, S. Ruhman, S. Ullman, "*Real Time Associative Vision Machine*", Artificial Intelligence and Computer Vision, Y.A. Feldman and A. Bruckstein , Elsevier Science Publishers B.V. !991.

[6] W.E.L. Grimson, "*A Computer Implementation of a Theory of Human Stereo Vision*", Phil. Trans. Royal Soc. of London, Vol. B 292, pp 217-253, 1981.