A RECONFIGURABLE DATA-FLOW ARCHITECTURE FOR A CLASS OF IMAGE PROCESSING APPLICATIONS

A.Sinha, *Member IEEE*, S.Neogi and K.Maiti
R & D Center, Himachal Futuristic Communication Ltd.,
286, Udyog Vihar, Gurgaon –122016, INDIA
amitabha_sinha@ieee.org

Abstract ---**This paper aims to device an architecture which uses capability of asynchronous concurrency of the data flow architecture as well as spatial parallelism of SIMD machines for a class of image processing applications using reconfigurable processing elements (RPEs). Overall processing speed is enhanced by a) concurrent functioning of the RPEs and b) replacing software execution of signal processing functions by hardware approach using FPGAs as RPEs. Thus, a hybrid architecture, which functions as a data flow machine at a functional level and exploits the capability of handling spatial parallelism by incorporating a modified SIMD concepts is presented.**

Index terms ---**Control Unit (CU), Signal Processing Instructions (SPIs), Processing Elements (PEs), Bit stream memory module (BSMM), Interconnection Network (ICN).**

1. INTRODUCTION

Intensive and complex computations are required for image processing algorithms on enormous amount of data. The real time processing requirement of this huge amount data will be far below the processing speed of the fastest available uni-processor system. It is observed that a large class of image processing algorithms exhibit spatial parallelism and is most suitable for SIMD machines. Imaging Architecture based on SIMD concept has been reported in [1][2][3] and a re-configurable SIMD architecture has been reported in [4]. SIMD machines employ a large number of tiny PEs working concurrently under the control of a CU. For a given algorithm, the CU will have to broadcast the simple machine instructions in lock-step fashion corresponding to a complex imaging instruction to all the PEs. In this process CU efficiency goes down to a large extent. Apart from that, the efficiency of these machines is limited by 1) Capability of the PEs, 2) Data communication between the PEs, 3) Capability of handling tree-structured algorithms.

High-speed requirements of digital image processing algorithms can be achieved by exploiting the spatial parallelism inherent in the algorithms and using a no of dedicated hardware to execute the specific functions. However incorporating all the functions in a single processing element will lead to complex, inefficient and costly solution. This problem can be handled by introducing dynamically reconfigurable [5][6] parallel architecture where PEs can be reconfigured on the fly depending on the Signal Processing Instructions (SPIs) issued to by the CU.

Data flow architectures [14][15] offers a possible way of exploiting concurrency of computations on a large scale. Highly concurrent computation in Data-flow concept is achieved by data-driven approach. In this model instruction firing is asynchronous [7][8][9]. These properties are especially suitable for Image Processing computations [10][11].

For a given imaging application, SIMD technique alone cannot handle the whole algorithm because an algorithm can be viewed as a collection of functional units working asynchronously and concurrently. However, each functional unit exhibits SIMD kind of spatial parallelism. Hence, in order to achieve substantial throughput gain, the approach should be to devise a technique such that there should be two fold concurrency: asynchronous concurrency at the top level and spatial parallelism within the functional units. Keeping this in view, this paper present a new hybrid architecture, which uses modified SIMD concepts as the processing units of the Data-flow machine at a functional level.

2. PROPOSED ARCHITECTURE

Since the proposed architecture (fig. 1) is meant for handling image-processing algorithms, it should be capable of handling both scalar and vector instructions efficiently.

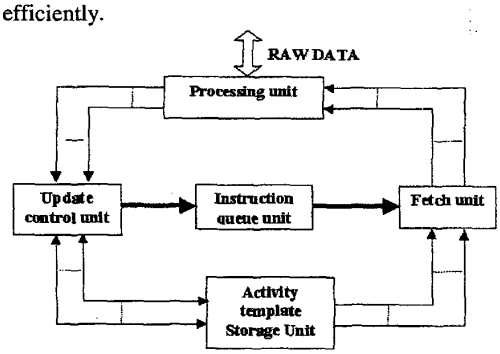

Figure 1: Architecture of the Hybrid DF-SIMD Machine

Scalar instructions are simple arithmetic (ADD, SUB), logical operators (AND, OR, etc.), relational (GREATER-THAN, LESS-THAN, etc.), decision making (SELECT, MERGE) etc.. Vector instructions include the imaging functions like FFT, SMOOTH,

EDGE-DETECTION, COSINE TRANSFORM, IMAGE RESTORATION etc.. Since these functions exhibit spatial parallelism, the basic processing unit of the proposed architecture is based on SIMD machines.

The architecture consists of five blocks: **1) Activity Template Storage Unit 2) Instruction Queue Unit 3) Fetch Unit 4) Processing Unit and 5) Update Control Unit.**

ACTIVITY TEMPLATE STORAGE UNIT (ATSU): This unit contains operation packet templates that represent the nodes in the data-flow graph. Each operation packet contains a field (S/V), which determines if the operation code is scalar or vector. It also contains op-code, slots for operands (for scalar) or pointers to the data (in case of vector) and destination address of its successor instruction. The format of the instruction template is shown below.

S/V	Op-Code	Destination Address	Pointer to data or Data1	Data2 , ...

A pointer will point to the start-address of the frame-memory. As soon as a frame arrives, the pointer value will be stored in the appropriate field of the operation packet. To determine the availability of operands or pointer, tag bits will be associated with each operand or pointer.

UPDATE CONTROL UNIT (UCU): This unit takes in the data tokens from the Processing unit (PU) and store then in its input pool and then passes them to their destination instructions in the ATSU depending on their availability. It also tests whether all the operands and acknowledge packets required to activate the destination instruction has been received and, if so, enters the instruction address into the Instruction Queue.

INSTRUCTION QUEUE UNIT (IQU): This unit stores the address of the active activity template. It is a FIFO buffer store. During execution the number of entries in the instruction queue measures the degree of concurrency present in the program.

FETCH UNIT (FU): This unit fetches templates from ATSU depending on the address in IQU and sends a complete operation packet to PU by placing it into the Operation Packet Queue (OPQ).

PROCESSING UNIT: This unit (fig. 2) consists of "OPQ", "CU", "vector processing unit (VPU)" and "scalar processing unit (SPU)". It generates one result packet for each destination field as specified by the operation packet.

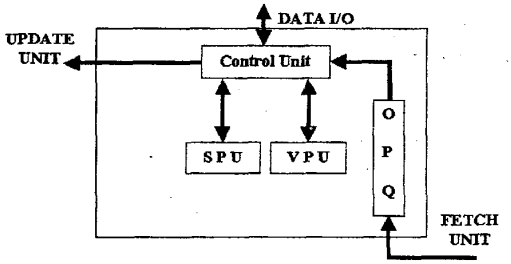

Figure 2: Processing Unit

VECTOR PROCESSING UNIT: This unit (fig. 3) consists of a 'N' numbers of RPEs, Bitstream Memory Decoder/Controller (BMDC), BSMM and an interconnection network (ICN).

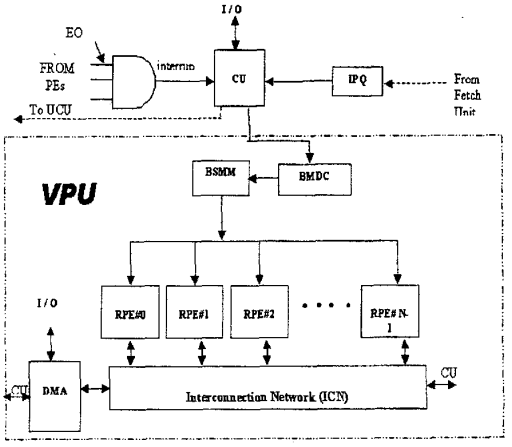

Figure 3: Vector Processing Unit (VPU)

BSMM contains the bitstream for configuring the RPEs and is controlled by the BMDC. A local memory is associated with each RPE. The issued SPIs (FFT, Hilbert transform, cosine transform, wavelet transform etc.) to the VPU are decoded by the BMDC, which then downloads the corresponding configuration bit-stream from the memory BMM to the RPEs to configure them to execute the corresponding SPI. The data sets for the RPEs are stored in their local memories through DMA. Concurrent to the execution of SPI on RPEs, the CU configures another part of the RPEs for execution of the next SPI. This process n overlaps configuration time of the RPEs with the latency of an SPI thus reducing CU idle time.

After execution of an SPI, the RPEs will interrupt the CU by sending End of Processing signals (EOP). Thus enabling the CU to access the final results and send the data tokens to UCU if needed.

Dual port memories connected in orthogonal fashion (Fig. 5) are used for the ICN [12].

Control Unit: This unit performs the following operations in sequence.
a) Read operation templates from OPQ, b) Detect instruction type (Scalar, Vector), c) Assign instruction (template) to the SPU or to the VPU depending on the type and d) Send result to UCU.

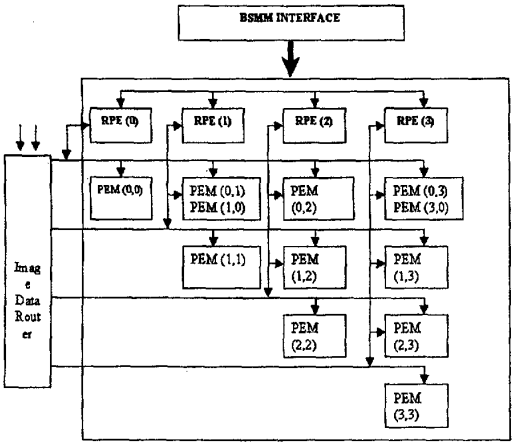

Figure 5: Orthogonal Memory Interconnection Network

3.MAPPING AN EDGE-DETECTION ALGORITHM TO DF-SIMD ARCHITECTURE

One of the commonly employed edge-detection gradient is the *Roberts gradient* [13]. It is generated by *Roberts masks*

$$R1=\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } R2=\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The Roberts gradient is given by

ROBERTS (f)=[FILTER (f; R1), FILTER (f; R2)]

At each pixel (i, j), assuming that all relevant gray values are defined, the outputs of two Roberts difference operators are

[FILTER (f; R1)](i,j) = f(i,j) - f(i-1,j+1)
and [FILTER (f; R2)](i,j) = f(i,j+1) - f(i-1,j)

Once the filtering has been accomplished, MAXNORM is applied to the resulting gradient vector to get ROBMAG. The fig.6 describes the Data-Flow approach of ROBMAG.

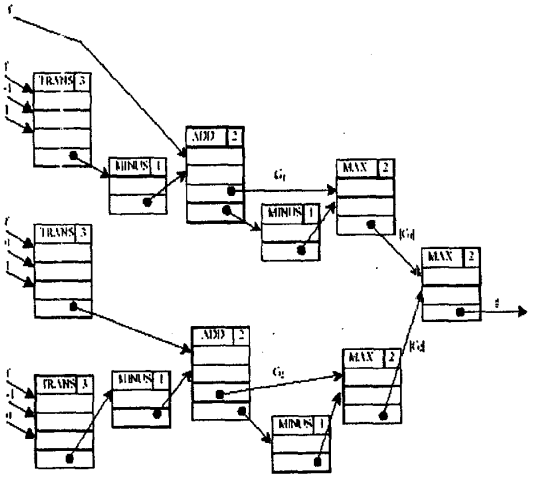

Figure 6: Data Flow Program of ROBMAG

4. TIME ANALYSIS BETWEEN THE PROPOSED AND DSP BASED ARCHITECTURE:

The motivating factor behind the design of this architecture has been an effort to reduce the execution time of a given complex digital signal / image processing functions, to meet real time requirements.

The time $T_{\text{total-FPGA}}$ required to execute one complex instruction on the proposed architecture using FPGAs is given as follows:

$$T_{\text{total-FPGA}} = t_{\text{enable}} + t_{\text{configuration}} + t_{\text{execution}} + t_{\text{communication}}$$

t_{enable} = time required by CU from receiving the Operation packet to enabling the configuration process.
$t_{\text{configuration}}$ = time for configuring the RPEs. By using the property of partial reconfigurability of the FPGAs this time can be made equal to zero.
$t_{\text{execution}}$ = time required for actual execution of the function (propagation delay time + routing delay time)
$t_{\text{communication}}$ =time required for inter-processor communication.

The corresponding time required in DSP processor based architecture is as follows:

$$T_{\text{total-DSP}} = t_{\text{instruction issue}} + t_{\text{decoding}} + t_{\text{execution}} + t_{\text{communication}}$$

$t_{\text{instruction issue}}$ = time required to issue an SPI.
t_{decoding} = time required for decoding the instruction.
$t_{\text{execution}}$ = time required for executing the high level instruction

$t_{communication}$ = time required for processor communication

Of all these times, the most contributing factor is the execution time of the SPIs.

Total time required by DSP based architecture to perform the above-mentioned ROBMAG Image Processing algorithm is,

$$T_{TOTAL\text{-}DSP\text{-}ROBMAG} = 3* t_{TRANS\text{-}DSP} + 4* t_{MINUS\text{-}DSP} + 2*t_{ADD\text{-}DSP} + t_{MAX\text{-}DSP}$$

Total time required by the proposed architecture to perform the same ROBMAG algorithm is,

$$T_{TOTAL\text{-}DFSIMD\text{-}ROBMAG} = 3* t_{TRANS\text{-}DFSIMD} + 4* t_{MINUS\text{-}DFSIMD} + 2*t_{ADD\text{-}DFSIMD} + t_{MAX\text{-}DFSIMD}$$

If there are 'N' number of RPEs in the SIMD machine the execution time of the SPIs (exploiting the spatial parallelism)

$$t_{TRANS\text{-}DFSIMD} \sim (t_{TRANS\text{-}DSP}) / K$$

Where $K < N$ due to the interprocess communication, but $K > 1$.

This will be the case of all the basic SPIs required to execute the algorithm, but the factor 'K' will vary depending on the SPI.

So, $T_{TOTAL\text{-}DSP\text{-}ROBMAG} >> T_{TOTAL\text{-}DFSIMD\text{-}ROBMAG}$

Thus the execution time of a DSP processor will be much more than the execution time of the algorithm in the proposed DF based architecture.

5. CONCLUSION

The objective is to derive a hybrid Data Flow architecture to meet the real-time requirements for a class of image processing applications. The philosophy behind the performance enhancement is based on introducing "dynamically reconfigurable computing " within the SIMD structure such that the execution of the signal processing functions can be performed at the speed of the hardware without losing the flexibility of the software. The architecture also overlaps the CU and RPE operation, thus reducing the CU idle time made possible by the partial reconfigurable property of FPGAs. Hence the architecture offers 1) Flexibility 2) Scalability 3) Flexible topology 4) PE Efficiency 5) Scheduling and synchronization at the hardware level 6) Efficient Utilization of Processing unit.

As number of RPEs increase, large number of image blocks can be compressed concurrently. Various performance studies on speedup factor, communication overhead based on Block size Vs Number of RPEs, size of IQU and OPQ, are to be investigated. Here we have adapted Static Data Flow architecture, but Dynamic Data Flow architectures have also to be explored.

6. REFERENCES

[1] S.Y.Kung , "VLSI Array Processor " ,Prentice Hall 1988

[2] H.J.Siegal, et al, "PASM: A partitionable SIMD/MIMD system for image processing and pattern recognition",IEEE Trans. Computer., Vol. C30, no. 12, pp. 934-947, Dec 1981

[3] A.Sinha, et al P.C.Jain, V.Mitra, "Asynchronous SIMD – A New Architecture For A Class Of Image Processing Applications", proc. Fifth National Conf. On Communication, pp. 287-294,IIT Kharagpur, India. 1999

[4] Hungwen Li and Quentin F.Stout, "Reconfigurable SIMD Massively parallel Computers",Proc. IEEE,Vol.79, No.4, April 1991, pp. 429-443.

[5] John Villasenor and William H. Mangione-Smith, "Configurable Computing", Scientific American Article, June 1997.

[6] John Villasenor and Brad Hutchings, "The Flexibility of Configurable Computing", IEEE Signal Processing Magazine,1998,pp 67-83

[7] J.B.Dennis, "Dataflow Supercomputers," Computer, pp48-56, Nov., 1980.

[8] Shuichi Sakai, Yoshinori Yamaguchi, Kei Hiraki, Yuetsu Kodama, and Toshitsugu Yuba. "An Architecture of a Dataflow Single Chip Processor," In Proc. 16th Annual Int'l Symposium on Computer Architecture, pp46-53. ACM, 1989

[9] Jack B Dennis, MIT "Stream Data types for signal processing," in Advanced topics in Dataflow Computing and Multithreading(Gao, Bic, Gaudiot eds.), IEEE Computer Society Press,1995.

[10] P.Netezki, "Exploiting Data Parallelism in Signal Processing on a Dataflow machine," ACM Annual Int'l Symposium on Computer Architecture pp. 262-272, 1989.

[11] Ben Lee and A.R. Hurson. "Dataflow architectures and multithreading," IEEE Computer, pages 27-38, Aug 1994.

[12] A.N. Choudhary and J.H. Patel, "A Parallel Process\-ing Architecture for an Integrated Vision System," *1988 International Conference on Parallel Processing*, August 1988, pp. 383-387.

[13] Edward R. Dougherty & Charles R. Giardina, "Matrix Structured Image Processing", Prentice Hall. 1987

[14] K. Hwang & F.A. Briggs,"Computer Architecture and Parallel Processing", McGraw-Hill, Singapore, 1985

[15] V.M.Bove and J.A.Watlington, "Cheops: A reconfigurable Data-flow system for video processing,"
IEEE Transactions on Circuits and Systems for Video Technology, 5, Apr. 1995, pp. 140-149.