Parallel Pipelined C-Slow Retimed Architecture through an Efficient Systolic Array

[1]G.Shanmugaraj, [2]Narender Malishetty, [3]G.Sethuram Rao
[1]Associate Professor, [2]Assistant Professor, [3]Assistant Professor
Velammal Institute of Technology

***Abstract* - A fully parallel pipelined array of cells is proposed for suitable real-time calculation of histograms. The cell structure builds on earlier work to now agree operating on a stream of data at single pixel per unit clock cycle. This novel cell is more fitting for interfacing to sensors of camera or to microprocessors of 8- or 16-bit data buses which are commonly used by consumer digital cameras. Image analysis created on histograms is copious and fully utilized in many consumer applications. The proposed architectures for systolic arrays and also make available a general method for mapping an algorithm to a systolic array. An array of cells to achieve the computation of n-bin histograms that takes p pixels per unit clock cycle offers to improvement a speedup factor of p. Likewise a design was proposed, but then required a sensor or processor providing four pixels per unit clock cycle to get a speedup in terms of four. Many real time embedded microprocessors contain of 8 bit data buses or 16 bit data and subsequently are able to supply one pixel per clock cycle. So as to feat this property, the proposed work is a histogram outcome using C-slow retiming to produce two sub streams of computation resulting from a dataset received at one pixel per unit clock cycle. Here proved that Arrays using the new proposed cells are obtained through C slow technique. Retiming techniques used at a faster frequency than earlier arrays.**

***keywords* - Embedded, Microprocessor, Parallel, Pipelined, Retiming.**

I. INTRODUCTION

The construction of two parallel architectures with modified data flow and systolic array is introduced. In the data flow architecture, an instruction is ready for execution when data for its operands have been made available [1]. Data availability is achieved by channeling results from previously executed instructions into the operands of waiting instructions. This channeling procedure a flow of data, triggering instructions to be effectively executed. An outcome of this is that many instructions are executed simultaneously, leading to the possibility of a highly concurrent computation [2].

In a systolic array structure, there are a enormous number of undistinguishable simple processors or processing elements (PEs). The PEs are organized in a well-ordered structure, Likewise a linear array or two-dimensional array type. Each PE has inadequate private storage and is coupled to neighboring PEs.

The next section explains the theory of data flow concept and explains one of the well-known designs of a data flow machine. This paper briefly explains the principle of C-slow retiming and. The essential result is that the proposed design affords speed-up while also facilitating easier interfacing to like camera sensors or microprocessors compared to other designs.

II. DATA FLOW ARCHITECTURE

To establish the performance of a data flow machine, a graph, called a data flow graph, is repeatedly used. The data flow graph signifies the data dependencies between distinct instructions [3]. It signifies the steps of a program and serves as an interface among system architecture and end user programming languages. The nodes present in the data flow graph, is called actors, denote the operators and are related by input and output arcs that carry tokens manner values [5]. Tokens are located on and then detached from the arcs according to convinced firing rules. Each actor requires certain input arcs to have tokens before it can be fired. When tokens are existing on all essential input arcs of an actor, that actor is enabled and thus fires. Upon firing, it eliminates one token from the mandatory input arcs, applies the itemized function to the values associated with the tokens, and places the result tokens on the output arcs.

Each node of a data flow graph can be signified as an activity template Similar to a data flow graph, a collection of activity patterns can be used for indicating a program. A specific set of actors that is used in a data flow graph (DFG). There are two kinds of tokens in the system: data tokens and Boolean tokens. To differentiate the type of inputs to an actor, solid arrows are used for taking data tokens and delineated arrows for Boolean tokens. The merge actor keeps one of the input tokens on the output arc depends on the Boolean token inward on its input. The T gate keeps the input token on the output arc on every occasion, the Boolean token arriving on its input is true, as the F gate does every time the Boolean token received on its input is false.

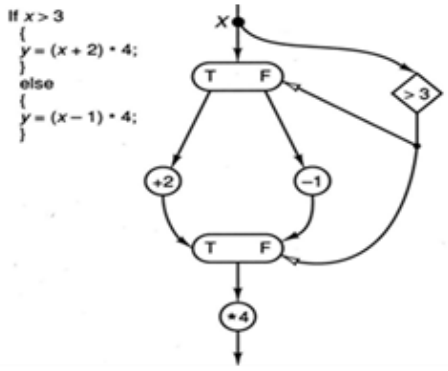


Fig.1. Represent data flow of IF statement

As an example, Fig. 1 represents a data flow graph with its corresponding templates for the following if statement: Another example is shown in Fig. 2. The Boolean input arcs to both mergers are initialized to false tokens. At the start this reason the data input token *N is to* move to the output of both mergers.

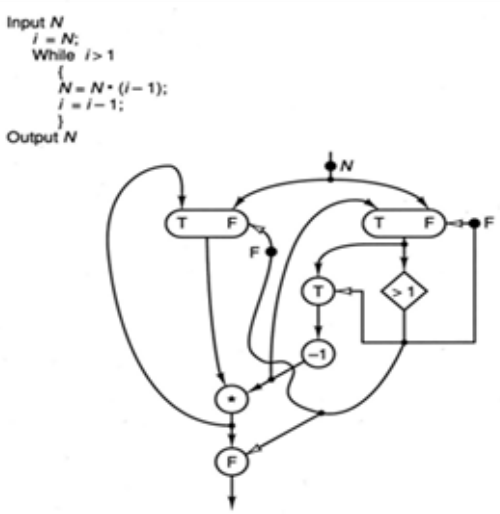


Fig. 2. Represent data flow of WHILE statement

III. STRUCTURE OF DATA FLOW COMPUTER

The structure of a data flow machine, Figure 3 represents the main elements of a data flow machine called the MIT data flow computer. The MIT computer contains of five main units: (1) the processing unit, involving of dedicated processing elements; (2) the memory unit, involving of instruct-ion cells for holding the instructions and its operands (3) the arbitration network, which delivers instructions to the processing elements for execution; (4) the distribution network for shifting the result data from processing elements to its memory unit; and (5) the control unit, which manages all other units [6].

An instruction cell grips an instruction containing of an operation code (opcode), operands, and its destination address. The instruction is enabled when all the essential operands and control signals are established. The arbitration network directs the enabled instruction as an operation packet to the appropriate processing element. Once the instruction is executed, the outcome is sent back via the particular distribution network to the specified destination in memory. Each result is directed as a packet, which entails of a value and a destination address [7].

Proposed data flow machines can be segregated into two groups: static and dynamic. In a static data flow machine, an instruction is enabled whenever all the essential operands are acknowledged and another instruction is waiting for the outcome of this instruction; otherwise, the instruction remains disabled. In other words, each arc in the data flow graph can carry at most a single token at any instance [8]. The multiply instruction is not been enabled until its previous result was used by the add instruction. Often, this constraint is imposed through the use of acknowledgment signals.

Wavefront array: The construction of a wave front array with a data stream is shown in Figure 4 Given two *n*-by-*n* matrices *A* and *B*, the matrix *A* can be decayed into columns A_i and other matrix *B* into rows B_j [9]. Thus

$$C = A1*B1 + A2*B2 + ... + An*Bn \quad (1)$$

Then the matrix multiplication can be carried out by the subsequent *n* iterations (1):

$$C(k) = C(k\text{-}1) + Ak*Bk \quad \text{recursively, for } k = 1, 2, ...,$$

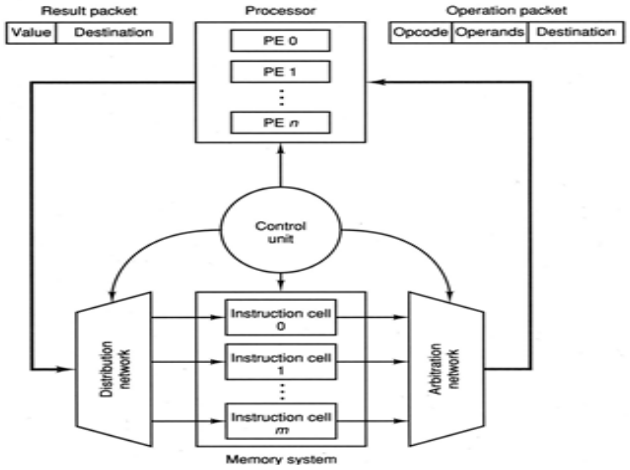


Fig. 3. Elements of data flow machine

In general, for a wavefront type of array with two *n*-by-*n* matrices (2), we have the following:

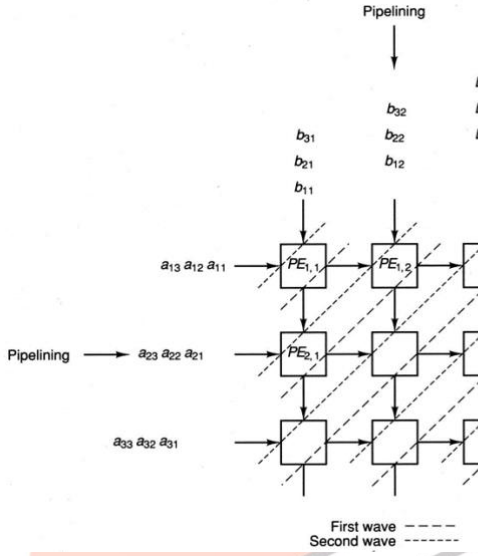


Fig. 4. Indicates processing elements of wavefront array

$$P = n^2, \quad (2)$$

$$T = 3n\text{-}2, \quad (3)$$

when the result stays in the array, $T = 4n\text{-}2$, (3) when the result is moved out of the array

As an example, consider the following algorithm which denotes multiplication of two 2-by-2 matrices *A* and *B*

```
for (k=1; k<=2; k++)
for (i=1; i<=2; i++)
for (j=1; j<=2; j++)
C(i,j)=C(i,j) + B(k,j) * A(i,k);.
```

Fig. 5 specifies the interconnections among the PEs. In this Fig. 5, every index element is shown as a three tuple (*k*,*i*,*j*). Note that for both index elements (*k*,*i*,1) and (*k*,*i*,2), the same value of *A*(*i*,*k*) is utilized that is, the value *A*(*i*,*k*) can be piped on the specified *j* direction of the PE. Likewise, values *B*(*k*,*j*) and *C*(*i*,*j*) can be piped on *i* and *k* directions, correspondingly. Based on these facts, the process can be rewritten by introducing buffering variables A^{j+1}, B^{i+1}, and C^{k+1}, as follows: it represents the

```
for (k=1; k<=2; k++)
for (i=1; i<=2; i++)
for (j=1; j<=2; j++)
   {
        Aj+1(i,k) = Aj(i,k);
        Bi+1(k,j) = Bi(k,j);
        Ck+1(i,j) =  Ck(i,j) + Bi(k,j) * Aj(i,k);
                                                }
```

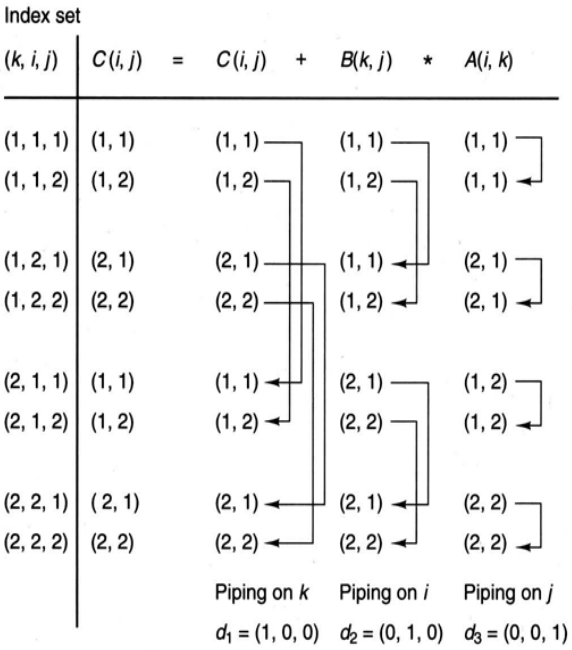


Fig. 5. Signifies the interconnections between the PEs.

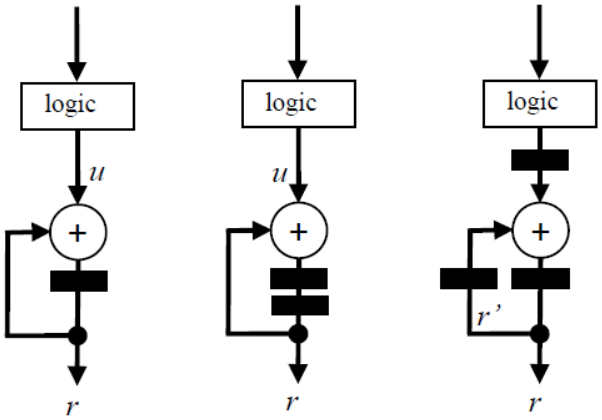
It represents the interconnections between the PEs. At time 1, *b*11 and *a*11 enter PE1,1, which holds variable *c*11. Then each PE completes a multiply and an addition operation; that is, *c*11=*c*11+*a*11**b*11. At time 2, *a*11 leaves PE1,1 and enters PE1,2. Simultaneously *a*12 enters PE1,1, *b*12 enters PE1,2, *b*11 enters PE2,1, and *b*21 enters PE1,1[6].
Yet again, each PE performs a multiply and an addition operation. Thus

*c*11=*c*11+*a*12**b*21, *c*12=*c*12+*a*11**b*12, *c*21=*c*21+*a*21**b*11.

This process remains until all the values are totaled.

A. Discussion on the C-slow effects

Fig. 6.a demonstrates that the process of re-timing reduces the main critical path delay occurs from the cost of a binary adder and the related logic to being either at the time of the binary adder or the logic time whichever is lengthier [4]. The downside is that the register count may rise expressively (by a factor of *C*) (B. Smith and E. H. Miller)



a. C-slow effects b.C-*slow retimed* c. Systolic C-slow retimed

Fig. 6. Represents a data flow graph for IF statement

B. C-slow retimed histogram processing cell

The cell construction above specified the Logic block has been preserved except for the element that C-slowing by a factor of two repeats the pipeline registers poignant data left to right in the original design. [10]. It should be valued that the structure looks very much like an illustration of Fig. 6.b It follows that, the separation of the computation into two streams does need the use of the extra adders. The critical path delay for the cell is either the comparison followed by the block of Logic or the adder [11].

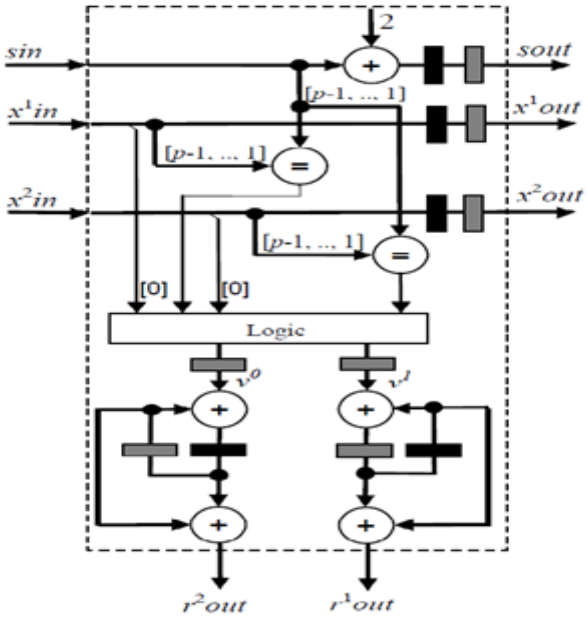


Fig. 7. C-slow retimed cell structure of two data items

IV. RESULT AND ANALYSIS

Although the C-slow cell is only about 25% faster than the traditional standard pipelined cell the real benefit comes when the cells are settled as an array. A pipelined array accepting 2 data items per clock cycle calculates the histogram in $n/2 + m/2$ clock cycles with each and every cell processing two bins; $m/2$ is the latency. The C-slow cell in Fig. 6.c needs two data items per clock cycle and Fig. 7. C-slow retimed cell internal structure processing the two data items simultaneously.

V. CONCLUSION

A new set of arrays of cells calculates *m*-bins histograms on streams of single pixel per clock cycle at over 80% of the recital of a pipelined array, employed on streams of two pixels per clock cycle. This is due to arrays of C-slow cells attaining rapid clocks than previous traditional pipelined arrays. The proposed array is subsequently better suited for when camera sensors or microprocessors are partial to supply single pixel per clock cycle.

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[3] H. Poor, *An Introduction to Signal Detection and Estimation*. New H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985.

[4] J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.

[5] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[6] M. Young, *The Technical Writers Handbook.* Mill Valley, CA: University Science, 1989.

[7] J. O. Cadenas, R. S. Sherratt, P. Huerta and W. C. Kao, "Parallel pipelined arrays for real-time histogram computation in consumer devices*," IEEE Trans. Consumer Electron.*, vol. 57, no. 4, pp. 1460- 1464, 2011.

[8] K. Yoon, C. Kim, B. Lee and D. Lee, "Single-chip CMOS image sensor for mobile applications," *IEEE J. on Solid State Circuits*, vol. 37, no. 12, pp. 1839-1845, 2002.

[9] C. Leiserson, F. Rose and J. Saxe, "Optimizing synchronous circuits by retiming," 3rd Caltech Conf. on VLSI, 1993.

[10] R. S. Sherratt and Oswaldo Cadenas, "A double data rate architecture for OFDM based wireless consumer devices," *IEEE Trans. Consumer Electron.*, vol. 56, no. 1, pp. 23-26, 2010

[11] Richard Hughey, Programming Systolic Arrays, Proc. Int. Conf. Application-Specific Array Processors, IEEE Computer Society, Aug. 4-7, 1992.