DIGITAL SYSTEM DESIGN FOR FADDEEV'S ALGORITHM

Tushar Chauhan **Abhishek Chauhan**,
SRM University, NCR Campus, Modinagar

Abstract-
In this project we are implementing Nash's systolic implementation and Chuang an He,s systolic implementation for Faddeev's algorithm. These two implementations have their own advantages and drawbacks. Here in this project report we first see detail of Nash implementation and then we will go for Chaung and He's implementation.

1. INTRODUCTION:

A systolic array is composed of matrix-like rows of data processing units called cells. Data processing units (DPU) are similar to central processing units (CPU)s, (except for the usual lack of a program counter, since operation is transport-triggered, i.e., by the arrival of a data object). Each cell shares the information with its neighbours immediately after processing. The systolic array is often rectangular where data flows across the array between neighbour DPUs, often with different data flowing in different directions. The data streams entering and leaving the ports of the array are generated by auto-sequencing memory units, ASMs. Each ASM includes a data counter. In embedded systems a data stream may also be input from and/or output to an external source.

2. Systolic Architecture:

A systolic array is composed of matrix-like rows of data processing units called cells. Data processing units (DPU) are similar to central processing units (CPU) s, (except for the usual lack of a program counter, since operation is transport-triggered, i.e., by the arrival of a data object). Each cell shares the information with its neighbours immediately after processing. The systolic array is often rectangular where data flows across the array between neighbour DPUs, often with different data flowing in different directions. The data streams entering and leaving the ports of the array are generated by auto-sequencing memory units, ASMs. Each ASM includes a data counter.
In embedded systems a data stream may also be input from and/or output to an external source.

An example of a systolic algorithm might be designed for matrix multiplication. One matrix is fed in a row at a time from the top of the array and is passed down the array, the other matrix is fed in a column at a time from the left hand side of the array and passes from left to right. Dummy values are then passed in until each processor has seen one whole row and one whole column. At this point, the result of the multiplication is stored in the array and can now be output a row or a column at a time, flowing down or across the array.

Systolic arrays are arrays of DPUs which are connected to a small number of nearest neighbour DPUs in a mesh-like topology. DPUs perform a sequence of operations on data that flows between them. Because the traditional systolic array synthesis methods have been practiced by algebraic algorithms, only uniform arrays with only linear pipes can be obtained, so that the architectures are the same in all DPUs. The consequence is, that only applications with regular data dependencies can be implemented on classical systolic arrays. Like SIMD machines, clocked systolic arrays compute in "lock-step" with each processor undertaking alternate compute | communicate phases. But systolic arrays with asynchronous handshake between DPUs are called wavefront arrays.

Applications

An application Example - Polynomial Evaluation

Horner's rule for evaluating a polynomial is:

$$y = (...(((a_n * x + a_{n-1}) * x + a_{n-2}) * x + a_{n-3}) * x + ... + a_1) * x + a_0$$

A linear systolic array in which the processors are arranged in pairs: one multiplies its input by and passes the result to the right, the next adds and passes the result to the right:

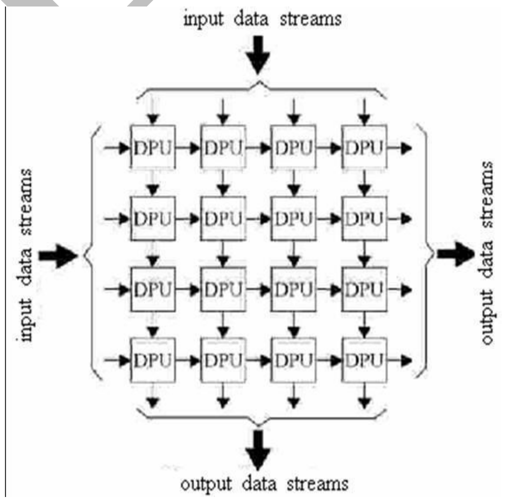
Advantages and Disadvantages

Pros

- Faster
- Scalable

Cons

- Expensive
- Highly specialized for particular applications
- Difficult to build



3. Systolic architecture implementation in VHDL:

This is a form of **pipelining**, sometimes in more than one dimension. Machines have been constructed based on this principle, notable the **iWARP**, fabricated by Intel.

• ‘Laying out algorithms in VLSI’

- efficient use of hardware
- not general purpose
- not suitable for large I/O bound applications
- control and data flow must be regular
- The idea is to exploit VLSI efficiently by laying out algorithms (and hence architectures) in 2-D (not all systolic machines are 2-D, but probably most are)
- Simple cells
- Each cell performs one operation (usually)

Definition 1.

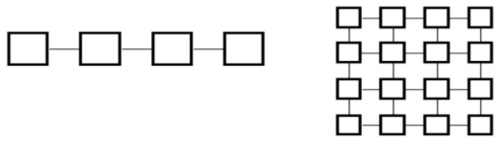
- sys·to·le (sîs¹te-lê) noun
- The rhythmic contraction of the heart, especially of the ventricles, by which blood is driven through the aorta and pulmonary artery after each dilation or diastole.
- [Greek sustolê, contraction, from sustellein, to contract. See systaltic.]
- — sys·tol¹ic (sî-stòl¹îk) adjective
- American Heritage Dictionary

Definition 2.

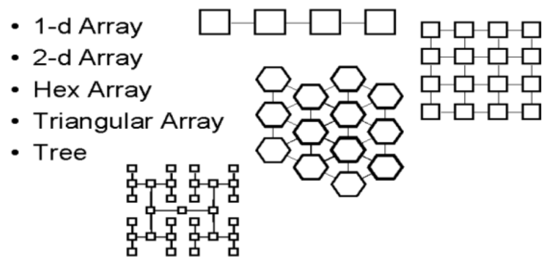
• Data flows from memory in a rhythmic fashion, passing through many processing elements before it returns to memory.

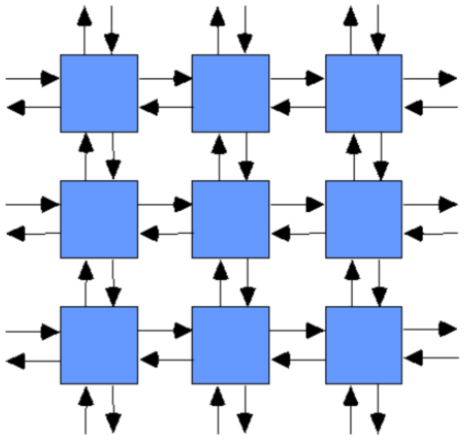
Definition 3.

• A set of simple processing elements with regular and local connections which takes external inputs and processes them in a predetermined manner in a pipelined fashion.



Different types of systolic array: =





Applications of systolic Array: =
• Matrix Inversion and Decomposition.
• Polynomial Evaluation.
• Convolution.
• Systolic arrays for matrix multiplication.
• Image Processing.
• Systolic lattice filters used for speech and seismic signal processing.
• Artificial neural network.
• Robotics (PSU)
• Equation Solving (PSU)
• Combinatorial Problems (PSU)
Features of Systolic Arrays: =
• A Systolic array is a computing network possessing the following features:
– Synchrony,
– Modularity,
– Regularity,
– Spatial locality,
– Temporal locality,
– Pipelinability,
– Parallel computing.

4. Gaussian Elimination for matrix computation-

The process of Gaussian elimination has two parts. The first part (Forward Elimination) reduces a given system to either triangular or echelon form, or results in a degenerate equation, indicating the system has no unique solution but may have multiple solutions (rank<order). This is accomplished through the use of elementary row operations. The second step uses back substitution to find the solution of the system above.

Stated equivalently for matrices, the first part reduces a matrix to row echelon form using elementary row operations while the second reduces it to reduced row echelon form, or row canonical form.

Another point of view, which turns out to be very useful to analyze the algorithm, is that Gaussian elimination computes matrix decomposition. The three elementary row operations used in the Gaussian elimination (multiplying rows, switching rows, and adding multiples of rows to other rows) amount to multiplying the original matrix with invertible matrices from the left. The first part of the algorithm computes an LU decomposition, while the second part writes the original matrix as the product of a uniquely determined invertible matrix and a uniquely determined reduced row-echelon matrix.

5. Faddeev's algorithm-

One general purpose algorithm, useful for a wide class of matrix operations and especially suited for systolic implementation, is the Faddeev's algorithm illustrated by the simple case of computing the value of ex + D, given AX = B, where A, B, C, and D are known matrices of order n, and X is an unknown matrix.

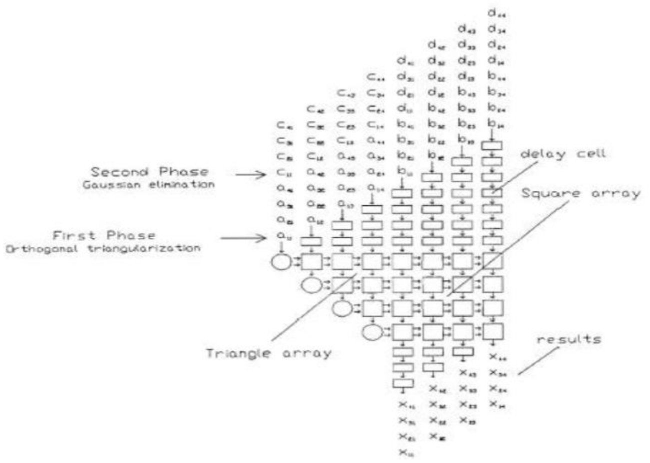
$$\begin{array}{cccc|cccc}
a_{11} & a_{12} & \cdots & a_{1n} & b_{11} & b_{12} & \cdots & b_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} & b_{21} & b_{22} & \cdots & b_{2n} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
a_{n1} & a_{n2} & \cdots & a_{nn} & b_{n1} & b_{n2} & \cdots & b_{nn} \\
\hline
-c_{11} & -c_{12} & \cdots & -c_{1n} & d_{11} & d_{12} & \cdots & d_{1n} \\
-c_{21} & -c_{22} & \cdots & -c_{2n} & d_{21} & d_{22} & \cdots & d_{2n} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
-c_{n1} & -c_{n2} & \cdots & -c_{nn} & d_{n1} & d_{n2} & \cdots & d_{nn}
\end{array}$$

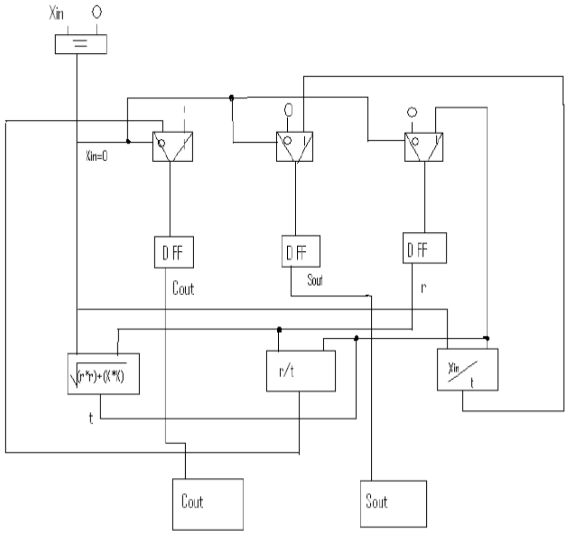
Nash implementation: =

To improve the stability of systolic implementation of Faddeev's algorithm Nash suggested a modification to Faddeev's algorithm by replacing the Gaussian elimination process used to triangularize the coefficient matrix A with orthogonal triangularization.

For clarity, it is useful to divide their algorithm into a two-phase procedure. In the first phase A is triangularized by series of given rotation (simultaneously applied to B); in the second phase, the diagonal elements of the resulting triangular matrix are use as pivoting elements in the Gaussian elimination procedure on c and where columns of c will be zeroed out and D will become the result. Note that for the Gaussian elimination process the work properly, it is necessary that these pivoting elements be non-zero, hence the requirement that A be full ranked, i.e. at least one of its square sub matrices of order n has non-zero determinant.

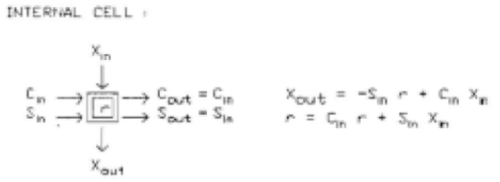
Nash systolic implementation, shon in the figure 1, consists of a triangular array and its right extension, a square array. The triangular array based on Kung's design for orthogonal trianguarization, performs given rotations an A (first phase) and ordinary Gaussian elimination on C (second phase). For higher efficiency in performing given rotations, cells macrocodes of figure are slightly modified in to those are also shown. Furthermore the added processing of ordinary Gaussian elimination requires the extra code that also shown. The square array simply extends the corresponding processing's to B and D thus consists only of square cells.



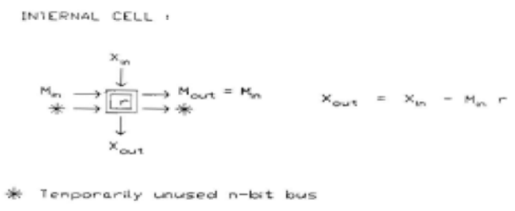


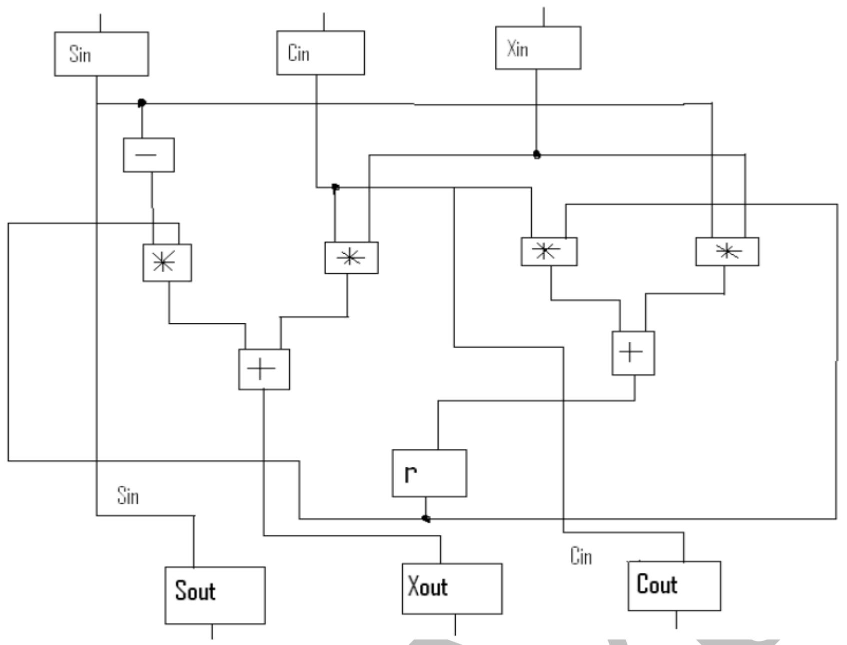
Square cell: =

In first phase it can be given as:



In Second phase:


INTERNAL CELL :
X_{in}
M_{in}
$M_{out} = M_{in}$
X_{out}
$X_{out} = X_{in} - M_{in}\ r$
* Temporarily unused n-bit bus



6. Chaung and He's implementation in detail and its VHDL coding-

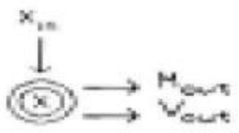
Another systolic implementation of Faddeev's algorithm proposed by Chuang and He, significantly improves upon the previous array. As shown in Figure 2, exist many similarities between the two arrays' design. To compute from nash implementation , both systems use a triangular array for the triangularization of A and the annulment of C, and a square array for extending the corresponding processing to B and D. The input data flow each system in a similar fashion. For the processing of the lower half of the input data flow (i.e. matrices C and D), both employ ordinary Gaussian elimination.

However, Chuang and He's system processes the upper half of the input data flow (i.e. matrices A and B) using Gaussian elimination with neighbor pivoting instead of the Givens transform.1 9 Hence, while numerical accuracy is somewhat inferior, this implementation is less expensive in terms of processing time and hardware complexities. Because the square root operation is not used, the array avoids the bottleneck problem created by the boundary cells of the Nash's array. And since the rightward data flow essentially consists of only one operand, Mout the pin counts of boundary cell and internal cell are correspondingly reduced to 3n and 4n, respectively.

Since it is obvious that different phases processing are required for the upper half and the lower half of the data flow, two separate sets of micro programs for boundary cells and internal cells are needed, as shown in Figure 9 and 10. The first set, the pivoting functions,
Performs Gaussian elimination with neighbor pivoting on A and B, while the second set, the non-pivoting functions, performs regular Gaussian elimination on C and D and is essentially the same as the functions of Nash cells in Figure 7.

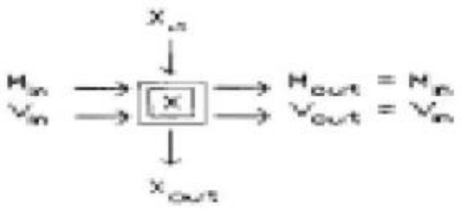
As the data flow is pipelined through the array, each boundary cell stores an input data element and sends a multiplier Mou t rightwards to modify the input data that o u enter the internal cells of the same row.

BOUNDARY CELL :



```
if $|X_{in}| \geq |X|$ then
  begin
    $V_{out} \leftarrow 1$
    if $X_{in} \neq 0$ then
      $M_{out} \leftarrow -X/X_{in}$
    else $M_{out} \leftarrow 0$
    $X \leftarrow X_{in}$
  end
else
  $V_{out} \leftarrow 0$
  $M_{out} \leftarrow -X_{in}/X$
```

INTERNAL CELL :



```
if $V_{in}$ then
  begin
    $X_{out} \leftarrow X + M_{in} * X_{in}$
    $X \leftarrow X_{in}$
  end
else
  $X_{out} \leftarrow X_{in} + M_{in} * X$
```

7. Conclusion

In this paper work and project I have good experience to implementing Nash and Chuang algorithm. There is lot of difficulties with Nash algorithm and it gives simulation errors. Chuang and He's algorithm are simple and can be simulate. But still in this also there is some timing issue. Although these algorithms are good and working fine this are not practical means they are not synthesizable.

REFERENCE

1. IEEE Recommended Practice for Instrumentation: Specifications for Magnetic Flux Density and Electric Field Strength Meters - 10 Hz to 3 kHz, IEEE Std 1308-1994, 1995. Performance."(n.d.).Federal Highway Administration. Retrieved October 8, 2006, from
2. W. J. Dally, "Interconnect-limited VLSI architecture," in Proceeding of IEEE International Conference on Interconnect Technology, pp.15–17, 1999.
3. J. D. Meindl, ''Beyond Moore's law: The interconnect era,'' Computer Science and Engineering. pp. 20-24, 2003.
4. Ahmed Shebaita and Yehea Ismail, "Multiple threshold voltage design scheme for CMOS Tapered Buffers" IEEE Transactions on circuits and Systems-II, Vol. 55, Page(s): 21 - 25 , January 2008.
5. Ahmed Shebaita and Yehea Ismail "Lower power, lower delay Design scheme for CMOS Tapered Buffers" , Design & Test Workshop (IDT), Page(s): 1 - 5 , 2009
6. N. C. Li, G. L. Haviland and A. A. Tuszynski, "CMOS tapered buffer," IEEE I.S.SC., vol. 25, no. 4, pp. 1005-1008, 1990.
7. B. S. Cherkauer and E. G. Friedman, "A unified design methodology for CMOS tapered buffers," IEEE Transactions on VLSI Syst., vol. 3, no. 1, 1995.
8. K. Roy, S. Mukhopadhyay, and H. Mahmoodi " Leakage current mechanism and leakage reduction techniques in DSM CMOS circuits", IEEE Proceeding., vol 91 No. 2, Feb. 2003.