

VLSI Architectures for the Finite Impulse Response Filter

KAM HOI CHENG AND SARTAJ SAHNI, MEMBER, IEEE

Abstract—We review the various VLSI architectures that have been proposed for the finite impulse response filter problem. In addition, new architectures are proposed and improved designs for some of the earlier architectures are developed.

I. INTRODUCTION

VLSI architectures for a variety of problems have been proposed by several authors. A bibliography of over 150 research papers dealing with this subject appears in [1]. In this paper, we are concerned solely with the finite impulse response (FIR) filter problem. The input to this problem is a $1 \times w$ vector $A = (a_1, a_2, \dots, a_w)$ and a $1 \times (n + w)$ vector $X = (x_0, x_1, \dots, x_{n+w-1})$. The output is a $1 \times (n + 1)$ vector $Y = (y_0, y_1, \dots, y_n)$ where

$$y_i = \sum_{j=1}^w a_j x_{i+j-1}, \quad i = 0, 1, \dots, n. \quad (1)$$

The FIR problem is of interest to VLSI designers as (1) is difficult to solve in real time for wide-band signal processing applications. Furthermore, the FIR problem has relatively favorable sensitivity properties with respect to round-off errors. Consequently, most application needs can be satisfied using fixed-point arithmetic of moderate precision. Additionally, the problem may be solved by algorithms requiring a simple control and flow of data. This facilitates highly parallel and pipelined computation.

VLSI architectures for this problem have been proposed earlier in [2]–[6]. The design of [3] and [4] uses a unidirectional chain of processors as in Fig. 1(a). Here data flow is from left to right. The design of [2], [5], and [6] employs a bidirectional chain of processors as in Fig. 1(b). In this, data flows both from left to right and from right to left.

In this paper, we examine each of the above two architectures. In addition, we consider broadcast chains [Fig. 1(c)] as used in [7]–[11] (among others) for the matrix multiplication, back substitution, LU decomposition, and the adaptive recursive filtering problems. A broadcast line has the property that data put on this line and becomes

available at all PE's on the line in $O(1)$ time. Furthermore, the systolic ring architecture [Fig. 1(d)] used in [3] for the LU decomposition problem is also examined.

In evaluating our designs, we assume that the VLSI system will be attached to the host processor using a bus as in Fig. 2. The evaluation of a VLSI design should take the following into account.

1) Bus bandwidth—How much data is to be transmitted between the host and the VLSI system in any cycle? This figure is denoted by B .

2) Speed—How much time does the VLSI system need to complete its task? This time may be decomposed into the times T_C (time for computations) and T_D (time for data transmissions both within the VLSI system and between the host and the VLSI system).

One may expect that by using a very high bandwidth B and a large number of processors P , we can make T_C and T_D quite small. So, T_C and T_D are not in themselves a very good measure of the effectiveness with which the resources B and P have been used. Let D denote the total amount of data that needs to be transmitted between the host and VLSI system. The ratio

$$R_D = B * T_D / D$$

measures the effectiveness with which the bandwidth B has been used. Clearly, $R_D \geq 1$ for every VLSI design. As an example, consider the multiplication of two $n \times n$ matrices. The host needs to send $2n^2$ elements to the VLSI system and receive n^2 elements back. So, $D = 3n^2$. With a bandwidth of n , T_D must be at least $3n$. T_D will exceed $3n$ if the bandwidth is not used to capacity at all times.

Let C denote the time spent for computation by a single processor algorithm. The ratio

$$R_C = P * T_C / C$$

measures the effectiveness of processor utilization. Once again, we see that $R_C \geq 1$ for every VLSI design. Consider the problem of multiplying two $n \times n$ matrices A and B to get X . Each element of X is the sum of n products. We shall count one multiplication and addition as one arithmetic (or computation) step. Hence, $C = n^3$. If $P = n$, then $T_C \geq n^2$.

In evaluating a VLSI design, we shall be concerned with T_C and T_D and also with R_C and R_D . We would like R_C and R_D to be close to 1. Finally, we may combine the two

Manuscript received May 28, 1985; revised July 29, 1985. This work was supported in part by the National Science Foundation under Grant MCS-83-05567.

K. H. Cheng is with the Department of Computer Science, University of Houston, Houston, TX 77004.

S. Sahni is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 8406184.

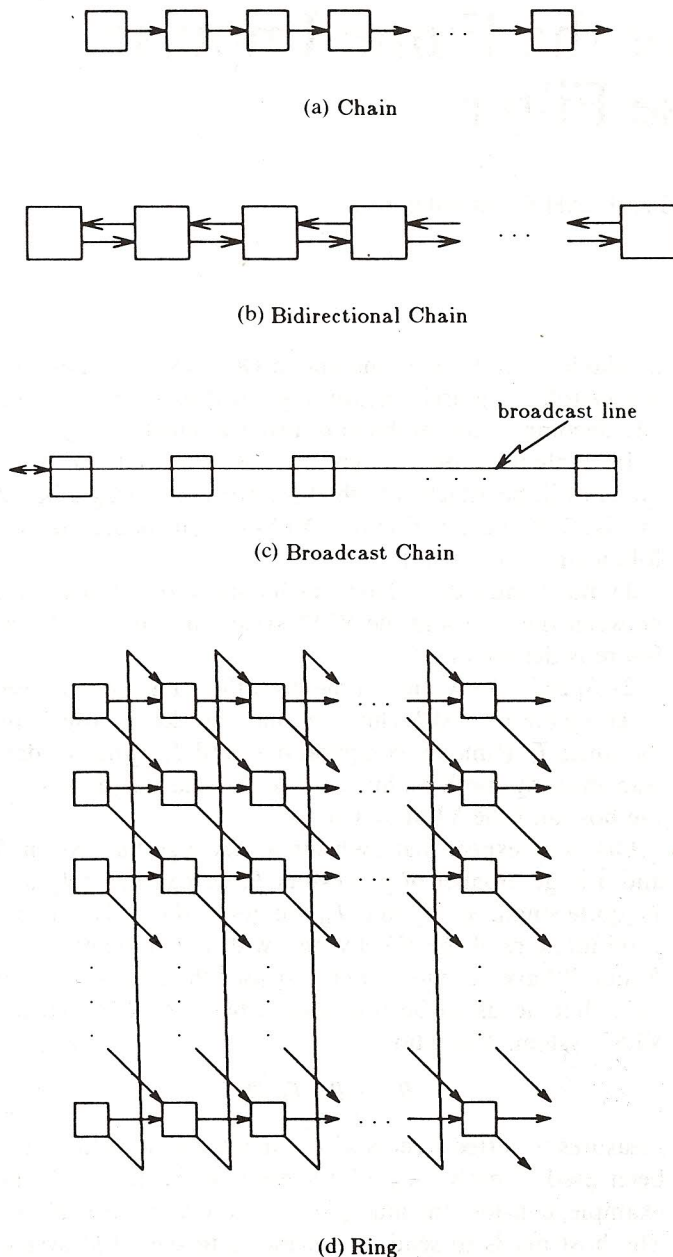


Fig. 1. Different VLSI architectures. (a) Chain. (b) Bidirectional chain. (c) Broadcast chain. (d) Ring.

efficiency ratios R_C and R_D into the single ratio $R = R_C * R_D$. A design that makes effective use of the available bandwidth and processors will have R close to 1.

The efficiency measure R as defined here is the same as that used in [8] and [9] to evaluate VLSI designs for matrix multiplication and back substitution. This measure is also quite similar to that proposed in [7]. In fact, the two measures become identical when $T_C = T_D$.

For each of the designs considered in this paper, we compute R_C , R_D , and R . In several cases, our designs have lower efficiency ratios than all earlier designs using the same model. In comparing different architectures for the same problem, one must be wary about overemphasizing the importance of R_C , R_D , and R . Clearly, using $P=1$ and $B=1$, we can get $R_C = R_D = R = 1$ and no speedup at all. So, we are really interested in minimizing T_C

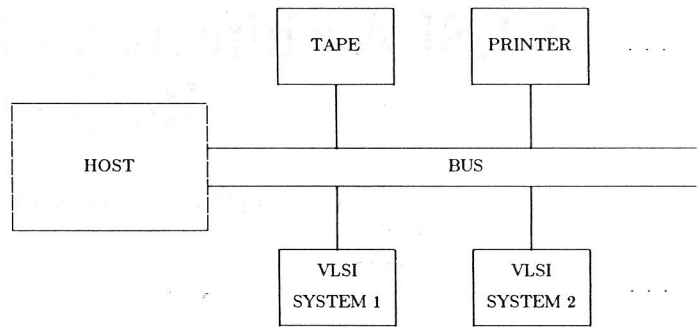


Fig. 2. Connecting to a host computer.

and T_D while keeping R close to 1. Some of our designs reduce T_C at the expense of R .

II. FINITE IMPULSE RESPONSE (FIR) FILTER

A. The Problem

Input: A $1 \times w$ vector A and a $1 \times (n+w)$ vector X .

Output: A $1 \times (n+1)$ vector Y that satisfies (1).

Parameters: $C = (n+1)w$, $D = 2(n+w)+1$.

B. Broadcast Chain

The broadcast capability permits a very straightforward and efficient solution to the FIR filter problem. We provide two such solutions. The first solution uses w PE's and has a throughput of $T_C = n+w$. The VLSI architecture is shown in Fig. 3(a) and the algorithm executed by the system is given in Fig. 4. Each PE has three registers, A , X , and Y . $Z(i)$ denotes register Z of PE(i), $Z \in \{A, X, Y\}$.

The algorithm consists of two FOR loops. The first initializes the A register of each PE such that $A(i) = a_i$, $1 \leq i \leq w$. Following this initialization, the system goes through $n+1$ steps in which the following happens.

1) Each PE receives the same x value from the broadcast line.

2) Each PE sends its Y value to the PE on its right. PE(1) initializes its Y register to zero. PE(w) outputs its Y register.

3) Each PE updates its Y register to $Y + AX$.

The first two events occur concurrently, while the third occurs after the first two have completed. It is easy to see that the first w Y 's output by PE(w) are garbage and the next $n+1$ Y 's output are y_0, y_1, \dots, y_n as given by (1).

In computing the performance figures for this VLSI system, we note that $B=2$ as the system requires at most one input and one output to occur concurrently. Observe that there is no Y input to the system as PE(1) simply initializes $Y(1)$ to 0 internally. So, we get $P=w$, $B=2$, $T_C = n+w$, $T_D = n+2w+1$, $R_C \sim 1+w/n \sim 1$, $R_D \sim 1+w/(n+w) \sim 1$ and $R \sim 1$.

The throughput of the VLSI system may be reduced to $o(n/k)$ by using k independent subsystems as in Fig. 3(b). Each subsystem has w PE's. Subsystem b computes y_i , $[(n+1)/k]b \leq i < [(n+1)/k](b+1)$, $0 \leq b \leq k$. The per-

