BIT LEVEL SYSTOLIC LATTICE FILTER SYNTHESIZER

B.B. Madan, S.R. Parker and M. Zubiar

Dept. of Electrical & Computer Engineering
Naval Postgraduate School
Monterey, CA. 93943

C.A.R.E.
I.I.T., New Delhi
INDIA-110016

ABSTRACT

This paper is concerned with the design of systolic architectures for a lattice filter synthesizer, incorporating bit-level pipelining. These architectures are derived by first of all representing the required computations by means of a signal flow graph. Next, we make use of the so called "delay transfer" and "gain transfer" rules to 'systolize' this SFG at bit-level. The resulting structures turn out to be linearly connected arrays of carry save adders involving only local nearest neighbour data paths.

SUMMARY

A lattice filter synthesizer has many important applications in areas such as speech synthesis, geo-physical signal processing, and linear prediction, etc. Because of the large number of computations required to implement a lattice filter structure, single processor architectures may not be able to achieve high enough data rates for real time applications. The solution lies in introducing concurrency by incorporating many processing elements operating in parallel and/or pipelined configuration.

With the advent of VLSI technology, the emphasis has shifted towards integrating a large number of such computing elements onto a single chip. However, this places some special constraints on the architecture. A multi-processor architecture which is being conceived for VLSI implementation, should preferably have:

- 1) Nearly identical processing elements to reduce design cycle time.
- 2) Only local nearest neighbour data paths for interconnecting these processing elements.

The second requirement is important because the data paths require chip area, propagation time, and energy.

Systolic array architectures proposed by Kung & Leiserson [1] meet the above two requirements and are therefore suitable for VLSI implementation. Systolic architectures

have been proposed for a number of problems e.g. vector transformation, matrix multiplication and digital filtering etc., to name a few. Also, a number of formal techniques have been proposed for designing systolic arrays for a given problem [2], [3]. We have found Kung's [2] Signal Flow Graph (SFG) cut-set approach to be particularly useful for designing systolic architectures for signal processing problems. This is because the computations of most signal processing algorithms can be represented by an SFG.

In order to introduce greater amount of concurrency and consequent resource utilization, one needs to introduce pipelining at bit level. A number of bit-level systolic array designs are available in the literature [4], [5]. However no formal methodology for designing bit-level systolic architectures has been proposed so far. Recently we have reported a formal technique for designing bit-level systolic architectures [6].

In this paper we apply the technique developed in [6] to obtain bit-level systolic arrays for the lattice filter synthesizer. The application of this technique to the problem at hand is briefly described in the following steps:

Step 1: A lattice filter synthesizer structure is represented by an SFG as in Fig. 1. The k_i 's are the filter coefficients and are assumed to be known.

Step 2: The filter coefficients can be represented by m-bit binary number i.e.

$$k_i = \sum_{j=0}^{m-1} k_i^j 2^j, \quad k_i^j = \{0,1\}$$

The filter structure can now be redrawn as in Fig. 2.

Step 3: Form cut-sets as shown by the broken lines in Fig. 3.

Step 4: Identify the arcs (edges) of the SFG which are delay free. Such edges imply global data paths and need to be avoided. Therefore,

delays (storage reg.) need to be introduced into these edges. With the addition of a delay, the variable associated with this edge is pipelined.

Step 5: Apply delay scaling and delay transfer rules [2] so as to be able to loan delays to delay free edges.

Step 6: Apply gain transfer rule [6] to convert multiple bit shifts implying global communication to single bit shifts requiring only local communication.

By following these steps, two different architectures can be obtained. Both of these architectures consist of locally interconnected arrays of carry-save adders. One of the designs is shown in Fig. 3. By suitably incorporating additional delays, a second architecture has also been obtained which makes it possible to time share a single 1-bit adder between m -stages and thereby reduce the area requirement at the expense of slightly slower throughput. Hence, depending on the requirement, we can make a tradeoff between the speed and the area. The final paper contains both these architectures.

REFERENCES

1. H.T. Kung and C.L. Leiserson, "Systolic arrays (for VLSI)", in Introduction to VLSI Systems, by C.A. Mead and L.A. Conway, Addison-Wesley, 1980.
2. S.Y. Kung, "On supercomputing with systolic wavefront array processors", Proc. IEEE, Vol. 72, pp. 867-884, July 1984.
3. D.I. Moldovan, "On the design of algorithms for VLSI systolic arrays", Proc. IEEE, Vol. 71, No. 1, Jan. 1983, pp. 113-120.
4. J.V. McCanny et al., "Bit level systolic arrays", Proc. 15th Asilomar Conf. on Cir. Sys. and Comp., Nov. 1981.
5. P.R. Capello and K. Steiglitz, "Completely pipelined architectures for digital signal processing", IEEE Trans. Vol. ASSP-31, No. 4, pp. 1016-1023, Aug. 1983.
6. B.B. Madan, S.R. Parker and M. Zubair, "A technique for the design of systolic arrays with bit level pipelining", submitted to IEEE Trans. Cir. and Sys.

Fig. 1 Lattice filter synthesizer SFG

Fig. 2 Lattice filter structure with bit-level coefficient representation

Fig. 3 Bit level systolic structure for lattice filter synthesizer