PIPELINING OF PARALLEL MULTIPLEXER LOOPS AND DECISION FEEDBACK EQUALIZERS

Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455
Email: parhi@ece.umn.edu

ABSTRACT

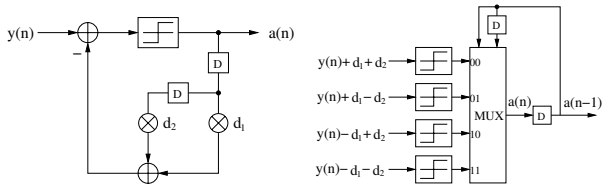
High speed implementation of a DFE (decision feedback equalizer) requires reformulation of the DFE into an array of comparators and a multiplexer loop. The throughput of the DFE is limited by the speed of the multiplexer loop. This paper proposes a novel look-ahead computation approach to pipeline multiplexer loops. The proposed technique is demonstrated and applied to design multiplexer loop based DFEs with throughput in the range of 3.125 - 10 Gbps.

1. INTRODUCTION

Decision feedback equalizers (DFEs) are widely used in various communication systems. High speed applications requires reformulation of a DFE into an array of comparators and a multiplexer loop [1, 2]. The throughput of the DFE is limited by the the speed of the multiplexer loop. For example, a 2-tap DFE in Fig. 1(a) can be reformulated as an array of 4 comparators and a 4-to-1 multiplexer loop. The iteration bound of this structure is T_{mux} (the computation time of a 2-to-1 multiplexer), which is about 0.2 ns in 0.13 μm CMOS technology.

By using the same method to the 2-tap DFE, a 6-tap DFE can be implemented by using 64 comparators and a 64-to-1 multiplexer loop. Fig. 2 shows the block diagram of the 64-to-1 multiplexer loop. It requires 32 instances of 2-to-1 multiplexer A, 16 instances of B, 8 instances of C, 4 instances of D, 2 instances of E and 1 instance of F blocks. There are a total of 63 2-to-1 multiplexers. The iteration bound of the 64-to-1 multiplexer loop is T_{mux}. However, to achieve this bound, either multiplexer F or the delay which is directly connected to the output of F will have a large fan-out. The computation time of multiplexer F is about twice that of a multiplexer which is not heavily loaded, resulting in an iteration bound of $2T_{mux}$, which is about 0.4 ns. Retiming and unfolding techniques cannot solve the problem since the iteration bound is a fundamental limit [3]. If the 6-tap DFE is implemented in this way, it can at most achieve a throughput of 2.5 Gbps. Therefore techniques must be developed for further pipelining and parallel processing of nested multiplexer loops for implementation of DFEs for speeds beyond 2.5 Gbps.

THIS RESEARCH WAS CARRIED OUT WHILE THE AUTHOR WAS WITH BROADCOM CORP., IRVINE, CA.



(a) A 2-tap DFE

(b) The reformulated 2-tap DFE

Fig. 1. A 2-tap DFE and its reformulated form

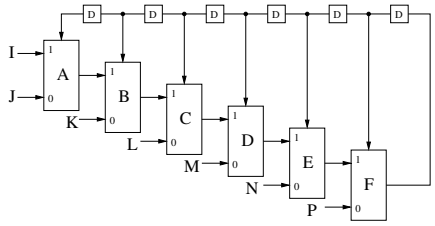


Fig. 2. A 64-to-1 multiplexer loop

In the past, look-ahead computation approaches were successfully applied to pipeline feed-forward adaptive lattice filters [4], linear time-invariant recursive filters [5], and add-compare-select operations in Viterbi decoders [6, 7, 8, 9]. Look-ahead computation was also used in multiplexer-based fast adders [10] and algorithms containing quantizer loops and piecewise-linear devices [1, 2]. However, the use of look-ahead computation in nested multiplexer loops is not straightforward, and has so far not been studied. This paper explores the problem of pipelining nested multiplexer

loops by using look-ahead computation.

The rest of the paper is organized as follows. In section 2, we study the problem of pipelining and look-ahead design for nested multiplexer loops. In section 3, the proposed approach is applied to the high speed design for DFEs with multi-Gigabit throughput.

2. PIPELINING NESTED MULTIPLEXER LOOPS BY LOOK-AHEAD COMPUTATION

This section extends the look-ahead techniques to pipeline nested multiplexer loops. We start with 2-stage pipelining of a 2-to-1 multiplexer loop. Next we study the problem of pipelining a 4-to-1 multiplexer loop which contains nested feedback loops. It is shown that pipelining by a certain form of look-ahead can successfully pipeline a nested multiplexer loop by an arbitrary number of stages.

2.1. 2-to-1 Multiplexer Loop

Fig. 3(a) shows a 2-to-1 multiplexer loop. The output, a_n of the circuit is given by:

$$a_n = A_n a_{n-1} + B_n \bar{a}_{n-1}. \tag{1}$$

The iteration bound of the circuit is T_{mux}. From (1), we have the output at time $n-1$ described by:

$$a_{n-1} = A_{n-1} a_{n-2} + B_{n-1} \bar{a}_{n-2} \ . \tag{2}$$

To pipeline the 2-to-1 multiplexer loop, we need to create additional delays in the loop by using look-ahead, which can be done by substituting equation (2) into equation (1). We can obtain:

$$a_n = f_{n,1} a_{n-2} + f_{n,2} \bar{a}_{n-2}, \tag{3}$$

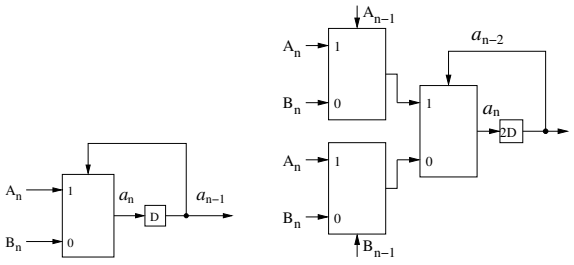
where $f_{n,1} = A_n A_{n-1} + B_n \bar{A}_{n-1}$ and $f_{n,2} = A_n B_{n-1} + B_n \bar{B}_{n-1}$. The resulting look-ahead structure is shown in Fig. 3(b). Its iteration bound is $\frac{T_{mux}}{2}$. It improves the iteration bound by a factor of 2. The hardware overhead is only two 2-to-1 multiplexers.

2.2. 4-to-1 Multiplexer Loop

Fig. 4(a) illustrates a 4-to-1 multiplex loop. Its output can be expressed as:

$$\begin{aligned} a_n = \quad & (A_n a_{n-2} + B_n \bar{a}_{n-2}) a_{n-1} \\ & + (C_n a_{n-2} + D_n \bar{a}_{n-2}) \bar{a}_{n-1}. \end{aligned} \tag{4}$$

As shown in Fig. 4(a), there are three 2-to-1 multiplexers and two delays. Assume the computation time of the inner multiplexer is 0.4 ns (i.e., it is highly loaded) and the other two multiplexers each have a computation time of 0.2 ns. This would be true if this 4-to-1 multiplexer was part of a much larger nested multiplexer loop. The iteration bound of the 4-to-1 multiplexer loop is then 0.4 ns.



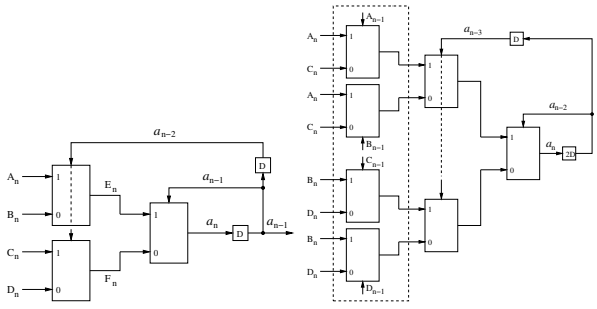
(a) Original loop (b) Pipelined design

Fig. 3. A 2-to-1 multiplexer loop and its 2-stage pipelined design

To improve the iteration bound, we can apply look-ahead to equation (4) by substituting its version at time $n-1$ into itself, we have:

$$\begin{aligned} a_n = \quad & [f_{n,1} a_{n-3} + f_{n,2} \bar{a}_{n-3}] \, a_{n-2} \\ + \quad & [f_{n,3} a_{n-3} + f_{n,4} \bar{a}_{n-3}] \, \bar{a}_{n-2}, \end{aligned} \tag{5}$$

where $f_{n,1} = A_n A_{n-1} + C_n \bar{A}_{n-1}$, $f_{n,2} = A_n B_{n-1} + C_n \bar{B}_{n-1}$, $f_{n,3} = B_n C_{n-1} + D_n \bar{C}_{n-1}$, and $f_{n,4} = B_n D_{n-1} + D_n \bar{D}_{n-1}$. The resulting look-ahead architecture is shown in Fig. 4(b). It consists of a 4-to-1 multiplexer loop and a 1-level look-ahead network as shown in the dashed box. The topology of the 4-to-1 loop is the same as that of the original 4-to-1 multiplexer loop except that the inner loop now contains two delays. The iteration bound of the architecture becomes 0.2 ns. The hardware overhead is only 4 2-to-1 multiplexers.



(a) Original loop (b) Look-ahead design

Fig. 4. A 4-to-1 multiplexer loop and its 2-stage look-ahead design

The look-ahead technique can be easily extended to larger

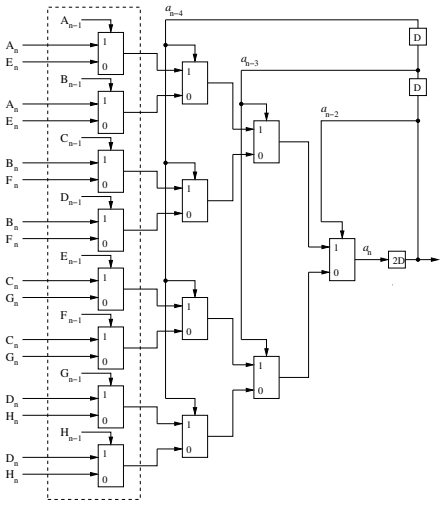


Fig. 5. 2-stage pipelining of an 8-to-1 multiplexer loop

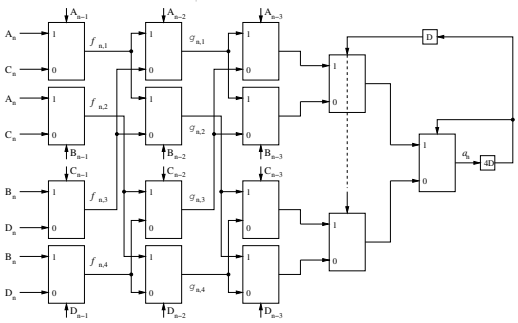


Fig. 6. 4-stage pipelining of the 4-to-1 multiplexer loop

multiplexer loops. Fig. 5 illustrates a 2-stage pipelined design of an 8-to-1 multiplexer loop. Its iteration bound is 0.2 ns and the hardware overhead is 8 2-to-1 multiplexers. The look-ahead technique can also be used to achieve multi-stage pipelining of multiplexer loops. Fig. 6 shows a 4-stage pipelined design for the 4-to-1 multiplexer loop. Its iteration bound is only 0.12 ns. Like the 2-stage pipelined design, it also has a regular structure. It consists of a 4-to-1 multiplexer loop and a 3-level look-ahead network. Each level contains 4 2-to-1 multiplexers. In general, for an L-to-1 multiplexer loop, its M-stage pipelined design consists of an $(M-1)$ level look-ahead network and an L-to-1 multiplexer loop. The difference between the original L-to-1 multiplexer and the L-to-1 multiplexer loop in the look-ahead design is that the number of delays of the innermost loop in the latter becomes M. Each level of the $(M-1)$ level look-ahead network consists of L multiplexers. Thus, the total overhead is $(M-1)L$ multiplexers and the total number of multiplexers in the M-stage pipelined L-to-1 multiplexer loop is $ML-1$. The iteration bound is $\frac{(log_2 L+1)T_{mux}}{M+log_2 L-1}$.

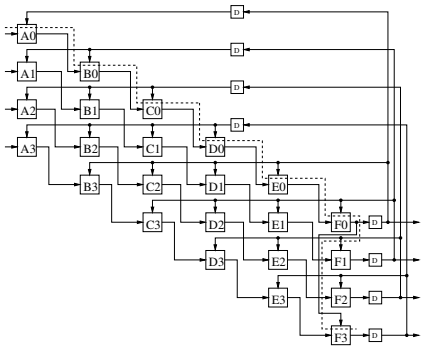


Fig. 7. A 4-unfolded design for a 3-stage pipelined 64-to-1 multiplexer loop

3. APPLICATIONS TO MULTI-GIGABIT DFE DESIGN

The proposed pipelining technique has been applied to design a 4-unfolded 6-tap DFE for a backplane receiver application requiring a throughput of 3.125 Gbps. The required clock period for the DFE is 1.28 ns. The throughput of the DFE is limited by a 4-unfolded 64-to-1 multiplexer loop. Before using the look-ahead technique, the 4-unfolded DFE has an iteration bound of 1.6 ns ($4 \times 2T_{mux} = 4 \times 2 \times 0.2 = 1.6$ ns) and a maximum throughput of 2.0 Gbps (which includes 0.4 ns clock setup/hold time and margin).

To meet the design requirement, we used a 3-stage pipelined 64-to-1 multiplexer loop. The iteration bound is $\frac{log_2 64+1}{3+log_2 64-1} \times 0.2 = 0.175$ ns. Fig. 7 shows a 4-folded design for the 3-stage pipelined 64-to-1 multiplexer loop. Its iteration bound is $4 \times 0.175 = 0.7$ ns. However, it has a long critical path as shown by the dashed line in the figure. The critical path consists of 5 normally loaded 2-to-1 multiplexers and two heavily loaded multiplexers. The computation time is $5 \times 0.2 + 2 \times 0.4 = 1.8$ ns. It does not meet the design requirement of shorter than 0.88 ns (with a margin of 0.4 ns). Thus, retiming is needed to reduce the critical path of the circuit. After applying the retiming algorithm [11] to the 4-unfolded 3-stage pipelined 64-to-1 multiplexer loop, we obtain the retimed circuit as shown in Fig. 8. The retimed circuit has seven critical paths as shown by the dashed lines. The computation times of critical paths 1 and 2 are 0.8 ns (i.e., $4 \times 0.2 = 0.8$ ns). The computation times of critical paths 3, 4, 5, and 6 are also 0.8 ns ($2 \times 0.2 + 0.4 = 0.8$ ns). Critical path 7 consists of two heavily-loaded multiplexer and its computation time is also 0.8 ns. The receiver with the pipelined 6-tap DFE can now be clocked with a clock period of 1.28 ns with enough operating margin (0.48 ns) for the clock setup/hold etc, achieving a rate of 3.125 Gbps. The total number of 2-to-1 multiplexers is $4 \times (64 \times 3 - 1) = 764$. The parameters for the 3.125 Gbps DFE design are listed in Table 1.

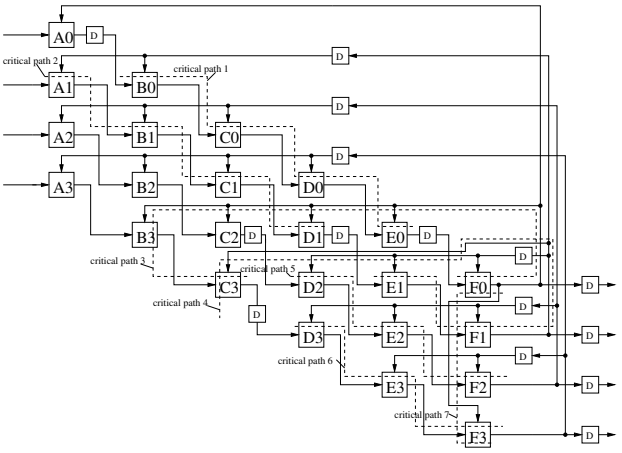


Fig. 8. Retimed 4-unfolded look-ahead for the 64-to-1 multiplexer loop

Table 1 also shows that the same technique can be used to design 5Gbps and 10Gbps 6-tap DFEs. For the 5 Gbps design, the unfolding level is 8 and the required pipelining level is 5. The iteration bound of the 8-unfolded 5-stage pipelined 6-tap DFE is 1.12 ns ($8*\frac{(log_2 64+1)\times 0.2}{5+log_2 64-1}$ ns), and the achievable least critical path after retiming is 1.2 ns. The number of multiplexers needed is $8 \times (64 * 5 - 1) = 2552$. For a 16-unfolded DFE, we can achieve a throughput of 10 Gbps with pipelining level 15. The number of multiplexers is $16 \times (64 * 15 - 1) = 15344$. From the table, we can see the hardware overhead is linear with the pipelining level.

Table 1. 3.125, 5 & 10 Gbps 6-tap DFE Design

Speed	Unfolding Level	Pipe-lining Level	Clock Period (ns)	Iteration Bound (ns)	Actual Least Critical Path (ns)	Number of Muxes
3.125G	4	3	1.28	0.7	0.8	764
5G	8	5	1.6	1.12	1.2	2552
10G	16	15	1.6	1.12	1.2	15344

4. CONCLUSION

This paper has extended the look-ahead computation technique to pipeline nested multiplexer loops. This approach is suitable for high-speed dedicated implementations for nested multiplexer loops. The hardware overhead is linear with the level of pipelining. The proposed technique has been applied to design multiplexer loop based DFEs with multi-Gigabit throughput.

5. REFERENCES

[1] S. Kasturia and J. H. Winters, "Techniques for high-speed implementation of nonlinear cancellation," *IEEE J. Select. Areas Commun.*, vol. 9, no. 5, pp. 711-717, June 1991.

[2] K. K. Parhi, "Pipelining in algorithms with quantizer loops," *IEEE Trans. on Circuits and Systems*, vol. 37, no. 7, pp. 745-754, July 1991.

[3] K. K. Parhi, *VLSI Digital Signal Processing System Design and Implementation*, John Wiley & Son, Inc., New York, 1999.

[4] K. K. Parhi and D. G. Messerschmitt, "Concurrent cellular VLSI adaptive filter architectures," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 1141-1151, Oct. 1987.

[5] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters, Part I and Part II," *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 1099-1135, July 1989.

[6] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785-790, Aug. 1989.

[7] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877-1885, Dec. 1992.

[8] H. D. Lin and D. G. Messerschmitt, "Improving the iteration bound of finite state machines," in *Proc. Int. Symp. Circuits and Systems*, pp. 1328-1331, May 1989.

[9] K. K. Parhi, "Look-ahead in dynamic programming and quantizer loops," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 1382-1387, May 1989.

[10] K. K. Parhi, "Low-energy CSMT carry generators and binary adders," *IEEE Trans. on VLSI Syst.*, vol. 7, no. 4, pp. 450-462, Dec. 1999.

[11] C. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming," in *Third Caltech Conference on VLSI*, pp. 87-116, 1983.