LOW COST DECISION FEEDBACK EQUALIZER (DFE) DESIGN FOR GIGA-BIT SYSTEMS

Chih-Hsiu Lin and An-Yeu(Andy) Wu

Graduate Institute of Electronics Engineering, and

Department of Electrical Engineering, National Taiwan University

Taipei, 106, Taiwan, R.O.C.

ABSTRACT

This paper addresses the design of a *Decision Feedback Equalizer* (*DFE*) for gigabits throughput rate in each communication path of 10G base LX4 Ethernet system. It is well-known that the feedback loop within a DFE limits an upper bound of the achievable speed. For a *L*-tap *Feed-Backward Filter* (*FBF*) with word-length *W* and *M-Pulse Amplitude Modulation* (*PAM*) modulated signal, the authors of [1][2] reformulate the FBF as a $(\log_2 M)^L$-to-1 multiplexer. Due to the reformulation, the overhead of extra adders and extra multiplexers are as large as $(\log_2 M)^L$. The required hardware overhead should be more severe when the DFE is designed in parallel. In this paper, we propose two new approaches to implement the DFE when Giga-bit speed is required. The first approach is partial pre-computation, which can trade-off between hardware complexity and computation speed. The second approach is two-stage pre-computation, which can be applied to higher speed applications. We can reduce the hardware overhead to about $2(\log_2 M)^{(-L/2)}$ times of [1][2], and the iteration bound is $2(\log_2 W+2)/L+(\log_2 M)$ multiplexer-delays. We demonstrate the proposed architectures to the 10G base LX4 Ethernet system.

1. INTRODUCTION

Pipeline scheme is successfully employed to increase the computation speed of the communication Equalizer Design [4][5]. Another approach to achieve high speed computation is parallel implementation [6][7]. We can relax the clock time to *N*-times for *N*-parallel implementation. Exploiting both pipeline and parallel processing for high speed applications are straightforward for non-recursive computations. However, recursive computations, such as DFEs, can not be easily pipelined or processed in parallel due to the feedback loops in these filters. For a filter with a loop, retime approach [9] can be used to move the delay elements of a shorter path to a longer path in loops; then, a smaller critical path can be obtained. However, retime approach can not shorten the iteration bound. Moreover, retime approach can not achieve the iteration bound in most cases. In order to achieve iteration bound, we can unroll a loop, which is referred as the unfolding scheme [5][7], and then apply the retiming approach.

However, for gigabits throughput rate in each communication path, all the aforementioned approaches can not achieve the desired speed. Fig. 1 is a conventional DFE architecture. The critical path of this DFE is one multiplier, one slicer, and two adders as drawn in bold line. For Ethernet 10G base LX4 [11], the modulation scheme is 2-PAM; therefore, the multiplier can be replaced by one 2-to-1 multiplexer. The iteration bound reduces to one multiplexer and two adders. This critical path does not meet the required speed, *i.e.*, 0.32 ns. The authors of [1][2] reformulate the FBF as 2^L-to-1 multiplexers. The iteration bound is reduced to one multiplexer. The delay time of one multiplexer is 0.14 ns in UMC 0.18 μm process technique. Again, the computation speed of the architecture [1][2] can not provide a delay element with enough margins. Despite that unfolding approach can be employed to achieve the desired throughput rate, the overhead will be extremely large.

In this paper, both hardware cost and computation speed are taken into account. Motivated by [1][2], two new approaches are proposed in this paper. The first approach is to pre-compute and sum the partial outputs of the FBF. This approach can be used to trade off between hardware complexity and computation speed. For higher speed applications, the second approach is proposed to reformulate the FBF as two-stage pre-computation. For *M*-PAM modulations and a *L*-tap FBF with wordlength *W*, we can reduce the hardware overhead to about $2(\log_2 M)^{(-L/2)}$ times of [1][2]. The iteration bound is only $2(\log_2 W+2)/L+(\log_2 M)$ multiplexer-delays.

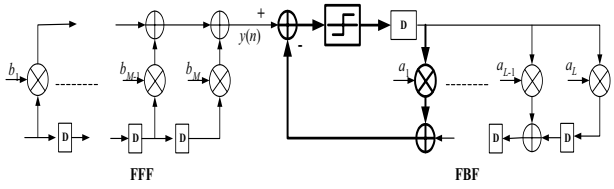

Fig. 1 The architecture of a DFE and its iteration bound.

2. REVIEW OF THE REFORMULATED FBF SCHEME [1][2]

We show a 2-tap FBF of a DFE in Fig. 2(a). The modulation scheme is 2-PAM. Fig. 2(b) is an alternative architecture by using retiming scheme. The critical path is one multiplexer delay. Furthermore, for an *L*-tap FBF, the fanout of the last stage multiplexer is 2^L-1. This will increase the delay time of this multiplexer. By using this reformulation approach, the whole overhead cost is 2^L adders and (2^L-1) 2-to-1 multiplexers. However, this architecture cannot provide margin enough for a delay element to process correctly for a 3.125Gs/b throughput rate in 10G base LX4 Ethernet system. To achieve 3.125 Gb/s, unfolding approach must be adopted. For an *N*-unfolding

implementation, the hardware cost is about N times. Therefore, this hardware cost of this architecture is very consumptive, especially when L is large.

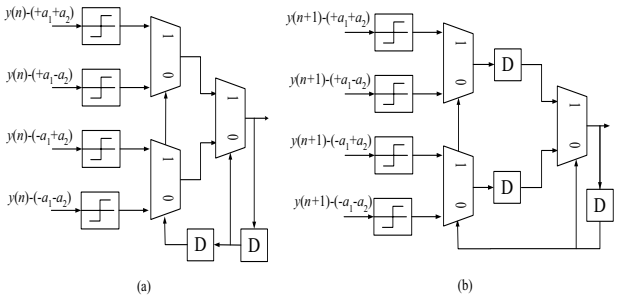

Fig. 2 (a) Reformulated 2-tap FBF of a DFE [1][2], (b) the retimed of the reformulated 2-tap FBF.

3. PRE-COMPUTATION DFE

In Fig. 1, when the transmitted signal is modulated by 2-PAM, the iteration bound is the computation delay of 2 adders and 1 multiplexer. The delay of a full adder and multiplexer is 0.28 and 0.14 ns in UMC 0.18 μm process technique, respectively. Therefore, the conventional DFE can not be employed to the application, in which the throughput rate is higher than gigabits per second. First, we discuss the partial pre-computation approach. Next, we present the two-stage pre-computation approach.

3.1 Partial Pre-Computation

Motivated by [1][2], we pre-compute and sum the partial outputs of the FBF, and then use a multiplexer to select the correct output. By doing so, extra delay elements can be used to pipeline the feedback loops. To clarify the presentation, we denote T_M, T_{add}, T_{FA}, and T_{mux} as the delay of one multiplier, adder, full adder, and multiplexer, respectively. As depicted in Fig. 1, the critical path is in the inner loop, *i.e.*, the path through multiplier a_1. When the transmitted signal is modulated as M-PAM, the multiplier a_1 can be replaced by $\log_2 M$-to-1 multiplexer. Furthermore, we can implement adders by carry-save adder, and then apply a multiplexer-based tree binary-look ahead carry generation Adder [10] to generate the output. For a wordlength W adder, the latency of a two-binary-adder by using [10] is $(\log_2 W+2)T_{mux}$. Therefore, the whole delay time can be reduced to $2T_{FA}+(\log_2 W+\log_2 M+2)T_{mux}$.

To speed up the computation of a DFE, we must reduce the critical path in a FBF. If we pre-compute N terms of a FBF, then we have extra N-delay elements to pipeline the feedback loop. An example of 2-term pre-computation is depicted in Fig. 3. The critical path can be reduced to 1/3 times. Although, the critical path is reduced to $1/(N+1)$ (N = 2 for 2-term pre-computation) times in the inner loop, the critical path is bounded by the adders in the rest of the FBF. Implementing a DFE like Fig. 3, we can use the N delay elements to pipeline the inner loop, and the critical path of the implementation can be reduced to T_{FA}, *i.e.*, 0.28 ns in UMC 0.18 μm process technique. Therefore, the throughput rate of the partial pre-computation can be up to gigabits. Moreover, we can implement a FBF in an alternative way, as shown in Fig. 4; especially when N is large enough. The critical path can be reduced to be shorter than T_{FA}.

In summary, for a small N-term pre-computation, the critical path is about $(2T_{FA}+(\log_2 W+\log_2 M+2)T_{mux})/(N+1)$. We implement a DFE like Fig. 3. For a larger N, we implement a FBF as in Fig. 4 to achieve a shorter critical path. The extra hardware costs of both architectures are 2^N adders and 2^N 2-to-1 multiplexers in N terms pre-computation DFE. In general, N must be chosen as smaller than $L/2$. This will be obvious in next section.

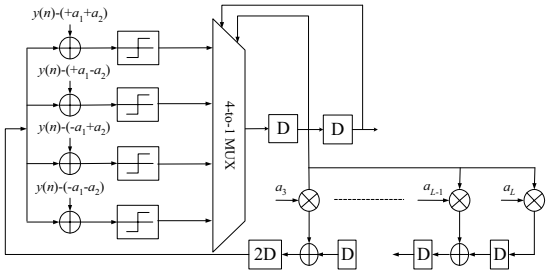

Fig. 3 The architecture of a DFE by using 2 terms pre-computation architecture

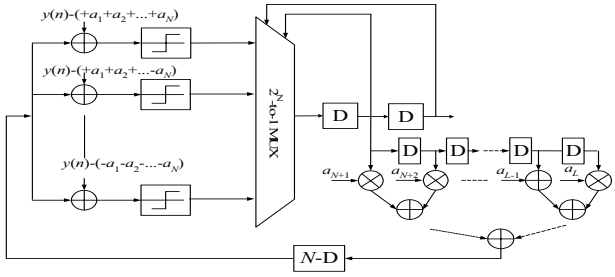

Fig. 4 The architecture of a DFE by using N terms pre-computation approach.

3.2 Two-Stage pre-Computation

In partial pre-computation method, we can trade off between hardware cost and computing speed. However, for very high-speed application, such as 10G base LX4, a small N-terms pre-computation can not provide a delay element with margin enough in UMC 0.18 μm process technique. As shown in Fig. 4, the critical path is limited by computing the sum of the outputs of a_{N+1}, …, and a_L. Therefore, if these terms are also pre-computed separately and simultaneously, we can shorten the path delay just at the price of smaller hardware cost. As illustrated in Fig. 5, the L-tap FBF is divided into two parts; One is the summation of the last terms, *i.e.*, weight a_{I+1} and a_L, called the 1st stage pre-computation. The 2nd stage sums the first I terms. The output, $y(n)$, of the FFF can be added by one of these two parts. Below, we discuss the properties of this scheme:

- *Partition scheme*: The outputs of each tap-delay line in the FBF can be partitioned into one of two parts, but leads to different iteration bound. The best grouping is to put the first terms in one group, and the rests are grouped together.
- *Asymptotic iteration bound*: For a L-tap FBF, the iteration bound of the two-stage pre-computation

method is $1/(I+1)T_{add}+(\log_2 M)T_{mux}$. Since the input number of this adder is 2, we can implement it by using a multiplexer-based tree binary-look ahead carry generation Adder [10]. The latency of a W-bit binary adder can be as short as $(\log_2 W+2)T_{mux}$. The complexity of the proposed architecture is as only $2(\log_2 M)^{(-L/2)}$ time as [1][2]. Moreover, the iteration bound of both two architectures is close to $(\log_2 M)T_{mux}$ [1][2] as L becomes large.

- *Low power*: For a communication system in which the channel changes very slowly, we can compute these sums of each stage and store in memories. In the 1st stage pre-computation, no new data is inputted when a slicer operates in data mode. We can compute the sum, store in memories. We need not update in each symbol; therefore, the computing power can be saved. In the 2nd stage, $y(n)$, the output of FFF, changes in each sample time, either when a slicer operates in training mode or data mode. Therefore, the computing power can not be saved in the 2nd stage.
- *Large fanout impairment*: As shown in Fig. 2(b), the fanout number of the multiplexer in the last stage is exponential to the tap-size of the FBF. This will increase the delay time of the last multiplexer. In the proposed architecture, the fanout number is degraded extremely compared with [1][2], so that the increased latency can be mitigated. However, each output of the 1st stage is carried into $(\log_2 M)^I$ adders. That is the fanout of 1st stage is equal to wordlength of the FBF. The large fanout will also increase the latency. In the practical design, some approaches can be used to mitigate the impairment. This will be shown in Section 4.

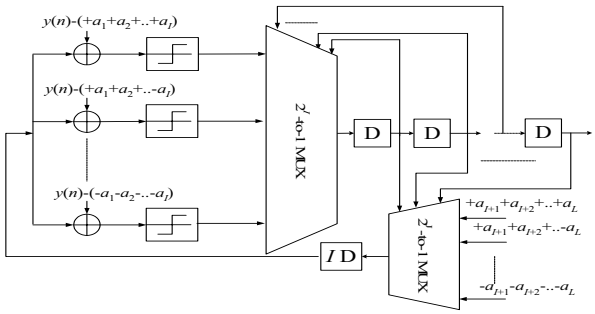

Fig. 5 Architecture of an L-tap FBF by applying Two-stage pre-computation approach.

4. APPLICATION TO Ethernet 10GBase-LX4

In 10G base LX4 Ethernet system, the communication is over 4 fibers. The transmitted signal is coded by 8B/10B scheme and modulated by 2-PAM. The actual date rate in each path is 3.125 Gb/s. That is the clock time of an equalizer must be 0.32 ns. The delay of a full adder and a delay element in UMC 0.18 μm technique process is 0.28 ns and 0.25 ns, respectively. The designed tap-size of the FFF and FBF is 8 and 6, respectively, and the word-length is 8. Therefore, it is impossible to pipeline the FFF to achieve the clock time. Parallel scheme must be used to have enough margins for a delay element.

In Section 3.1, we propose the partial pre-computation approach to increase the computation speed of a FBF. Despite that the very low extra hardware overhead of the proposed partial pre-computation scheme, the proposed partial pre-computation can not achieve the desired throughput rate of 10G base LX4 in UMC 0.18 μm process technique. Therefore, we use the two-stage pre-computation approach to design the FBF. To implement this FBF with the lowest hardware cost, I must be chosen as 3. Then, the iteration bound is 2.25 T_{mux} and the cost is 16 adders and 14 multiplexers. As aforementioned, we design the DFE by 4-unfolding implementation to provide sufficient margin for a delay element.

4.1 Architecture of a 6-Tap FBF with 4-Unfolding

We depict the 4-unfolding architecture of a 6-tap FBF in Fig. 6. The delay time of a *Vector Merge (VM)* Unit is $5T_{mux}$[10]. The overall critical path is T_{mux} as drawn in dash line. Moreover, the multiplexers A_{02}, A_{12}, A_{22}, A_{32}, B_{02}, B_{12}, B_{22}, and B_{32} suffer from the large fanout. In order to achieve the desired throughput rate, the architecture of Fig. 6 is retimed as in Fig. 7. We discuss some design efforts in detail as follows:

- *Employ inverted multiplexer*: The delay of a multiplexer with an inverted output is faster than a multiplexer without inverter output in the standard cell library. For the clarification of presentation, we call a multiplexer with an inverted output an inverted multiplexer. In UMC 0.18 μm technique process, the latency of an inverted multiplexer can be as a half as a multiplexer and is denoted as T_{muxi}. Moreover, when the selector of a multiplexer is inverted, we just exchange the two inputs. Then, the output of the multiplexer will be the same. Therefore, we replace some multiplexers with inverted multiplexers in our design. While the outputs of the multiplexers A_{02}, A_{12}, A_{22}, and A_{32} is added by the sum of 1st stage pre-computation, we do not replace these multiplexers with inverted multiplexers; otherwise, the output values will be changed. On the other hand, the fanout number of these multiplexers is 16; therefore, a multiplexer with large driving capacity is chosen.
- *Isolate large fanout by inserting buffers*: As showing in Fig. 6, the fanout of the multiplexers B_{02}, B_{12}, B_{22}, and B_{32} is also very large. To shorten the latency, we apply one inverter and buffer to increase the driving capacity, as showing in Fig. 7. By doing so, we can mitigate the degradation of the delay time in the critical path due to the large fanout. For example, the fanout of multiplexer B_{02} is multiplexers A_{02}, A_{11}, A_{20}, B_{30}, B_{21}, and B_{12}. We insert one buffer to drive A_{11} and cascade another to A_{20}, which has the largest loading capacity. This can reduce the fanout of multiplexer. Then, the critical path through B_{02}, A_{02}, VM, B_{00}, and B_{01} is reduced. Also, we apply one inverter to drive the buffer and the rest multiplexers.
- *Retiming scheme*: Although, we can reduce the latency from B_{02} to A_{02} (example above), the latency from B_{02} to A_{11} and A_{20} is increased. To solve the problem, we can

apply retime approach. For example, the latency from B_{02} to A_{11} is two multiplexers, one inverter, and one buffer delay time. Therefore, the delay element D_1 must be moved to the front of the VM Unit. Then, the latency from B_{10} to A_{12} is close to from B_{02} to A_{12}. Therefore, the effect of the delay time of the inserted buffer can be eliminated. D_2 and D_3 are retimed by using the same way. The architecture of Fig. 6 is retimed as in Fig. 7.

In summary, after some modifications, the critical path is in dash line and the latency is 0.9 ns, which is evaluated by using UMC 0.18 μm technique process. It can provide the delay element with margin enough. The extra overhead is only 64 adders and 56 2-to-1 multiplexers. The types of cells are listed in Table 1.

Table 1 The table list the types and delay time of all cells.

Cell name	A_{02}, A_{12}, A_{22}, A_{32}	B_{02}, B_{12}, B_{22}, B_{32}	B_{01}, B_{11}, B_{21}, B_{31}	*The rest inverted multiplexers*
Type	X4	X4	X2	X1

5. CONCLUSION

In this paper, we proposed two new schemes to speed up the computation of a FBF. The proposed architecture can reduce the hardware overhead extremely, and the latency is increased slightly. The partial pre-computation can trade-off between hardware complexity and computation speed. Two-stage pre-computation can be used for very high speed applications and applied to 10G base LX4 Ethernet system.

6. REFERENCE

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

[2] 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.

[3] Kamran Azadet Erich F. Haratsch, Helen Kim, et al. "Equalization and FEC techniques for optical transceivers," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 317-327, Mar 2002.

[4] P. M. Kogge, *The Architecture of Pipelined Computers*. McGraw-Hill, 1981.

[5] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filter- Part I and II, " *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 1099-1135, July 1989.

[6] J. I. Acha, "Computational structures for fast implementation of L-path and L-block digital filters," *IEEE Tran. on Circuits and Systems*, vol. 36, no. 6. pp. 805-812, June 1989.

[7] L. E. Lucke and K. K. Parhi, "Parallel processing for rank order and stack filter," *IEEE Trans. on Signal Processing*, no. 5, pp. 1178-1189, May 1994.

[8] L. F. Chao and E. Sha, "Retiming and unfolding data-flow graphs," *in Proc. of 1992 International conference on Parallel Processing, part II*, (St. Charles, IL), pp. 33-40, Aug. 1992.

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

[10] K. K. Parhi, "Fast low-power VLSI binary addition", *in Proc. Of 1997 IEEE International Conference on Computer Design (ICCD)*, pp. 676-684, Oct. 1997.

[11] *IEEE Std 802.3ae-2002* (Amendment to IEEE Std 802.3-2002)

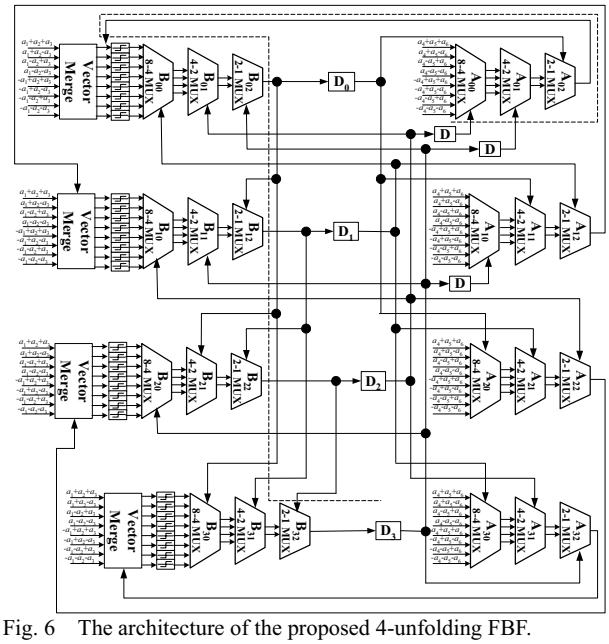

Fig. 6 The architecture of the proposed 4-unfolding FBF.

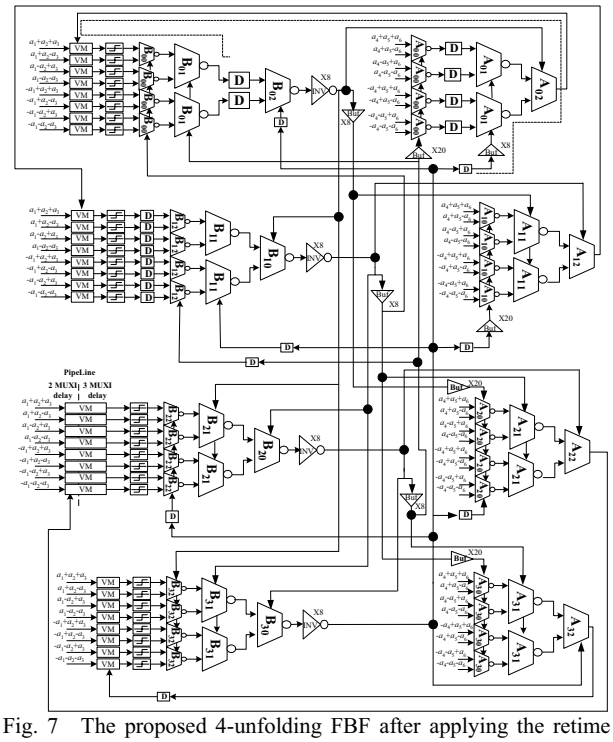

Fig. 7 The proposed 4-unfolding FBF after applying the retime approach and some modifications.