PIPELINED PARALLEL DECISION FEEDBACK DECODERS (PDFDS) FOR HIGH SPEED ETHERNET OVER COPPER

Yongru Gu and Keshab K. Parhi

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

ABSTRACT

One of the powerful yet simple algorithms to decode trellis codes as well as to combat intersymbol interference (ISI) is the parallel decision-feedback decoding algorithm. However, for high-speed applications, such as Gigabit Ethernet over copper, the implementation and design of a parallel decision-feedback decoder (PDFD) is challenging due to the long critical path in the decoder structure. Straightforward pipelined designs usually introduce significant hardware overhead. To solve these problems, in this paper, first, based on an optimized scheduling of the computations in the parallel decision-feedback decoding algorithm, a low complexity pipelined PDFD is proposed. Next, we present a retiming and reformulation technique for the decision feedback unit (DFU) in the PDFD which can remove the DFU from the critical path of the PDFD with negligible hardware overhead. Compared with similar designs in the literature, the proposed design can reduce hardware overhead by 60% while achieving similar speedup for Gigabit Ethernet systems.

1. INTRODUCTION

Many digital communication applications employ trellis code and pulse amplitude modulation (PAM). A typical example is 1000BASE-T (Gigabit Ethernet over copper), which uses a 4 dimensional (4D) 8-state trellis code combined with a 5-level PAM modulation. In the presence of ISI and additive Gaussian noise, it is well established that, to decode the trellis coded PAM signals, the maximum-likelihood sequence estimation (MLSE) implemented by the Viterbi algorithm can provide optimal performance in terms of error rate event. However, the complexity of the algorithm is exponential with the sum of the channel memory length and the trellis code memory length. Thus it is highly desirable to reduce the complexity of the detection technique while retaining near optimal performance. One of the most powerful approaches for doing so is the so-called parallel decision-feedback decoding. In this approach, an independent feedback signal is computed for each path in the Viterbi decoder, as the convolution of the sequence of symbols associated with that path, and the coefficients of the feedback filter of the decision feedback equalizer [2, 3].

For high-speed applications, such as 1000BASE-T, the implementation of a parallel decision feedback decoder (PDFD. Parallel decision-feedback decoding is also referred as PDFD) is challenging because of its long feedback loop in the decoder structure. Precomputation and look-ahead techniques based pipelined designs, such as the one proposed in [4], usually introduce significant hardware overhead, especially for a PAM modulation with a large symbol set. Thus, how to reduce the hardware overhead due to precomputation and look-ahead is also a main concern.

This paper addresses the problem of low-complexity pipelined designs for PDFDs for 1000BASE-T. First, a novel optimized scheduling for the PDFD algorithm is proposed. Next, based on the optimized scheduling, a low complexity pipelined PDFD design is proposed. In addition, a retiming and reformulation technique for the DFU in the PDFD is presented. The retiming and reformulation technique can remove the DFU from the critical path of the PDFD with negligible hardware overhead.

The rest of the paper is organized as follows. In section 2, the PDFD algorithm and its straight-forward hardware implementation are briefly reviewed. Section 3 presents an optimized scheduling for the PDFD algorithm and its corresponding pipelined PDFD design. In section 4, a retiming and reformulation technique is presented for the PDFD which can remove the DFU from the critical path.

2. PDFD ALGORITHM AND ITS STRAIGHTFORWARD IMPLEMENTATION

1000BASE-T systems employ a 4D trellis code combined with 5-level PAM (PAM5) modulation [1]. One of the preferred algorithms to decode the trellis code is the PDFD al-

THIS RESEARCH WAS SUPPORTED IN PART BY THE NATIONAL SCIENCE FOUNDATION BY THE GRANT NUMBER CCF-0429979.

gorithm [1, 4]. The PDFD algorithm can be used to combat ISI as well as to decode the 4D trellis code for 1000BASE-T. Fig. 1 illustrates the straight-forward scheduling of the PDFD process. At time n, it first computes ISI estimates. Next, the well-known Viterbi algorithm is applied to the ISI-free data. The Viterbi algorithm begins with the computation of 1D branch metrics. Then the 1D branch metrics are added up to obtain 4D branch metrics. Finally, they are used to update state metrics and survivor paths. This process is repeated at next iteration. The computations are performed in a serial way, resulting in a straightforward architecture where all its constituent units, which include a DFU (decision feedback unit), a 1D BMU (branch metric unit), a 4D BMU, an ACSU (add-compare-select unit) and an SMU (survivor memory unit), are on the critical path, as shown in Fig. 2. For details about the straightforward PDFD, the readers are referred to [5].

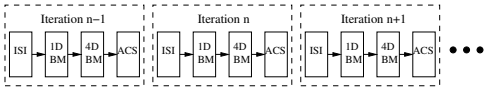


Fig. 1. The straightforward scheduling of computations in the PDFD algorithm

For 1000BASE-T, the PDFD needs to operate at a speed of 125 MHz. It is difficult to meet the speed requirement due to the long critical path. In [4], a pipelined PDFD design was proposed based on precomputation and look-ahead techniques but it introduces significant hardware overhead due to the use of precomputation. Compared with the straightforward design, the area of the pipelined design in terms of gate count is almost doubled [4]. Furthermore, the hardware overhead is proportional to the number of levels in the PAM modulation. If we use the same techniques in [4] to applications with PAM modulation with a larger symbol set, the overhead can be more significant. Thus it is of interest to develop techniques to speedup the straightforward PDFD with less increase in hardware overhead.

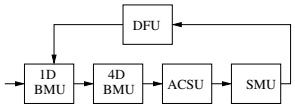


Fig. 2. The straightforward PDFD architecture

3. PROPOSED HIGH SPEED LOW COMPLEXITY PDFD ARCHITECTURE

In this section, based on an optimized scheduling of the PDFD algorithm, a pipelined PDFD architecture is proposed.

Fig. 3 shows the proposed scheduling. Right after finishing the computation of 1D branch metrics for iteration n, we can begin to pre-compute the branch metrics for the next iteration $(n + 1)$ since we already know the two possible candidate 1D symbols for each wire. The real 1D branch metrics can be selected upon the completion of the ACS operation of iteration n. This process is repeated at the next time as illustrated in the figure.

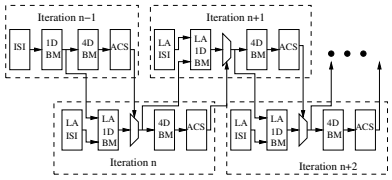


Fig. 3. An optimized scheduling of computations in the PDFD algorithm

Fig. 4 shows the proposed PDFD architecture corresponding to the optimized scheduling in Fig. 3. The sequential computation path in Fig. 2 now becomes two concurrent computation paths in the proposed design. One path consists of the look-ahead DFU (LA DFU), the look-ahead 1D BMU (LA 1D BMU) and the 1D BM selection unit. The other includes the 4D BMU, the ACSU and the SMU.

In the following, the designs of the LA DFU and the LA 1D BMU are discussed. The architectures for the 4D BMU, ACSU, and SMU are similar to those in the straightforward PDFD implementation [5].

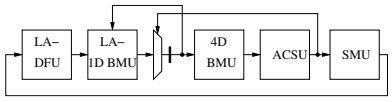


Fig. 4. High speed PDFD architecture

3.1. Look-ahead DFU

At time n, the look-ahead DFU is used to compute partial ISI estimates for code state ρ_{n+1} and wire j due to the channel coefficients $\{f_{2,j}, f_{3,j}, \ldots, f_{L,j}\}$ based on the already known survivor symbol sequence where L is the channel memory length. Assume there is a state transition between ρ_n and ρ_{n+1}, then the partial ISI estimate for ρ_{n+1} corresponding to the transition can be calculated as:

$$\tilde{u}_{n+1,j}(\rho_n) = -\sum_{i=2}^{L} f_{i,j} a_{n-i+1,j}(\rho_n). \qquad (1)$$

Since there are 8 code states and 4 wires, altogether 32 look-ahead ISI estimates need to be computed.

3.2. Look-ahead 1D BMU

The look-ahead 1D BMU computes look-ahead 1D branch metrics for transitions departing from code states $\{\rho_{n+1}\}$.

Inputs to the look-ahead 1D BMU are partial ISI estimates $\{\tilde{u}_{n+1,j}(\rho_n)\}$ due to $\{f_{2,j}, f_{3,j}, \ldots, f_{L,j}\}$ and the received sample $z_{n+1,j}$. In addition, we need to consider the ISI partial contribution due to the channel coefficient $f_{1,j}$ and the 1D symbol decision $a_{n,j}(\rho_n \rightarrow \rho_{n+1})$ associated with a state transitions $\rho_n \rightarrow \rho_{n+1}$. A speculative ISI estimate for the state transition $\rho_n \rightarrow \rho_{n+1}$ can be calculated as

$$\hat{u}_{n+1,j}(\rho_n \rightarrow \rho_{n+1}) = \tilde{u}_{n+1,j}(\rho_n) - f_{1,j} a_{n,j}(\rho_n \rightarrow \rho_{n+1})$$
$$= -\sum_{i=2}^{L} f_{i,j} a_{n-i+1,j}(\rho_n) - f_{1,j} a_{n,j}(\rho_n \rightarrow \rho_{n+1}). \quad (2)$$

Due to PAM5 modulation, there are 5 possible choices for $a_{n,j}(\rho_n \rightarrow \rho_{n+1})$, and in turn 5 possibilities for $\hat{u}_{n+1,j}$ $(\rho_n \rightarrow \rho_{n+1})$. Thus, in a straightforward implementation, like in [5], we need to compute $5 * 2 * 8 * 4 = 320$ look-ahead 1D branch metrics since there are eight code states and four wires and for each of the possibilities 2 look-ahead 1D branch metrics need to be computed.

However, the architecture in Fig. 4 allows us to reduce the hardware overhead by feeding back the previous 1D branch metric results (for transitions $\{\rho_n\} \rightarrow \{\rho_{n+1}\}$) to the current calculation of the look-ahead 1D branch metrics. After the completion of 1D branch metrics for transitions departing from a state ρ_n, there are only two possible choices for $a_{n,j}$ associated with the state transition $\rho_n \rightarrow \rho_{n+1}$, one $(a_{n,j}(\rho_n, A))$ from subset A and the other $(a_{n,j}(\rho_n, B))$ from subset B. In addition, the two possibilities for $a_{n,j}$ are only dependent on ρ_n. Thus, there are only two possibilities for $\hat{u}_{n+1,j}(\rho_n \rightarrow \rho_{n+1})$. Therefore, we only need to pre-compute look ahead 1D branch metrics for the 2 possibilities, resulting in high hardware reduction.

As the two possible choices for $a_{n,j}(\rho_n \rightarrow \rho_{n+1})$ are only dependent on the initial state ρ_n, the possible ISI estimates for state ρ_{n+1} are only dependent on ρ_n too. For code states $\{\rho_{n+1} = 0, 1, 2, 3\}$, as they have the same predecessor states $\{\rho_n = 0, 2, 4, 6\}$, their LA 1D branch metrics are the same. Therefore, we only need to compute LA 1D branch metrics for one of them. This is also true for code states $\{\rho_{n+1} = 4, 5, 6, 7\}$. For wire j and initial code state ρ_n, four look-ahead 1D branch metrics need to be calculated according to:

$$\hat{\lambda}_{n+1,j}(r_{n+1,j}, a_{n+1,j}, \rho_n, a_{n,j}) = (r_{n+1,j} - a_{n+1,j} + \tilde{u}_{n+1,j}(\rho_n) - f_{1,j} a_{n,j})^2, \quad (3)$$

with two (one per 1D subset for $a_{n+1,j}$) for $a_{n,j} = a_{n,j}$ (ρ_n, A) and two for $a_{n,j} = a_{n,j}(\rho_n, B)$. As there are eight code states and four wires, altogether $8 \times 4 \times 4 = 128$ 1D look-ahead branch metrics need to be computed. Compared with the design in [5], the hardware overhead for LA 1D branch metric calculation is reduced by 60% in the proposed design.

Fig. 5 shows the calculation of LA 1D branch metrics for wire j and initial code state ρ_n. The inputs are the received sample $r_{n+1,j}$, the look-ahead ISI estimate $\tilde{u}_{n+1,j}(\rho_n)$, and the two possible candidates for the transmitted symbol $a_{n,j}$ associated with the state ρ_n, obtained from the last iteration.

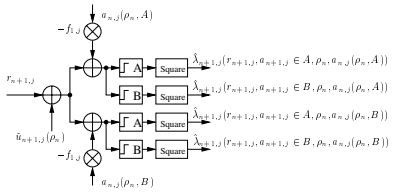


Fig. 5. Computation of look-ahead 1D branch metrics

3.3. Selection of Look-ahead BMs

For code state ρ_{n+1} and wire j, we need to select two real 1D branch metrics (one for $a_{n+1,j} \in A$ and one for B) among 16 precomputed branch metrics (four from each of 4 predecessor states of ρ_{n+1}).

Fig. 6 shows the selection for the A-type branch metric $\lambda_{n+1,j}$ $(r_{n+1,j}, a_{n+1,j}(\rho_{n+1} = 0, A), \rho_{n+1} = 0)$. The inputs are 8 eight precomputed branch metrics with two from each of 4 predecessor states, the 1D symbol decision associated with state transition $\rho_n \rightarrow \rho_{n+1}$ from the 4D BMU, and the ACSU decision $d_n(\rho_{n+1})$.

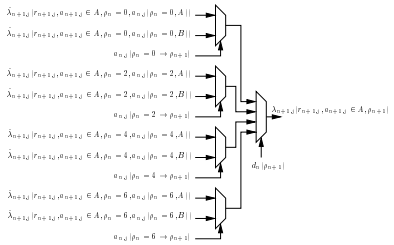


Fig. 6. Selection of look-ahead 1D branch metrics

4. RETIMING AND REFORMULATION OF DFU

From Fig. 4, we can see there are two major computation paths in the architecture. One of them consists of the LA DFU, the LA 1D BMU, and the 1D BM selection unit. The other consists of the 4D BMU, ACSU and SMU. We expect the first path dominates the computation time and is the critical path in the proposed design. In this section, we propose a technique to further pipeline the first path.

Fig. 7(a) illustrates the composite architecture for the look-ahead DFU and the SMU used in Fig. 4. As shown by the dashed line in the figure, there is a long-chain of adders

directly connected to the 1D BMU, resulting in a long critical path which limits the throughput of the architecture in Fig. 4. However, we can use the proposed retiming and reformulation technique shown in Fig. 7(b) through Fig. 7(d) to remove the DFU from the critical path.

First, we can isolate the long chain of adders from the BMU by using the retiming cutsets shown by the dotted lines in Fig. 7(b). The resulting circuit is shown in Fig. 7(c). Applying retiming again by using the cutset shown in Fig. 7(c), we can obtain the retimed DFU in Fig. 7(d). However, from this figure, we can see that the long chain of adders is now connected to the ACSU through a mux, and the DFU is still on the critical path. If we can move the multipliers before the corresponding muxes, then there are delays between the long chain of adders and the ACSU. This can be done by performing the following reformulation:

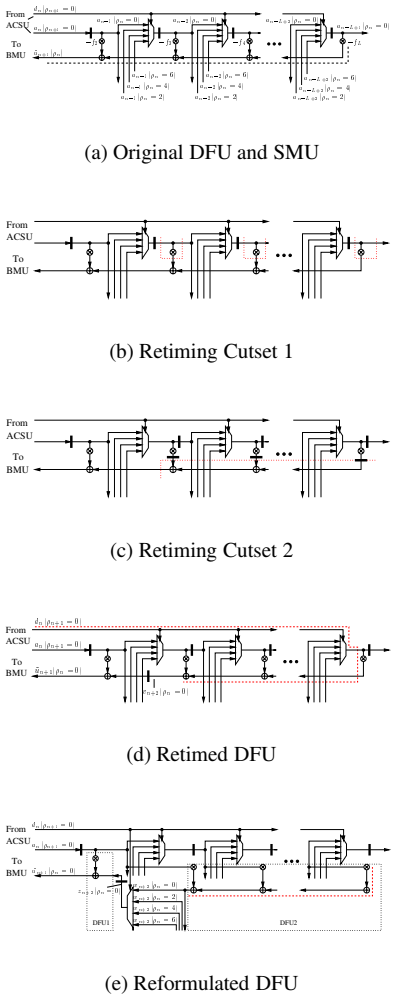
$$\begin{aligned}
&\textstyle\sum_{i=3}^{L} Sel(d_n(\rho_{n+1}=0), a_{n-i+2}(\rho_n=0),\\
&\quad a_{n-i+2}(\rho_n=2), a_{n-i+2}(\rho_n=4),\\
&\quad a_{n-i+2}(\rho_n=6)) * f_i\\
&= Sel(d_n(\rho_{n+1}=0), \textstyle\sum_{i=3}^{L} a_{n-i+2}(\rho_n=0) f_i,\\
&\textstyle\sum_{i=3}^{L} a_{n-i+2}(\rho_n=2) f_i, \sum_{i=3}^{L} a_{n-i+2}(\rho_n=4) f_i,\\
&\quad \textstyle\sum_{i=3}^{L} a_{n-i+2}(\rho_n=6) f_i), \qquad (4)
\end{aligned}$$

where $Sel(d, x_0, x_1, x_2, x_3)$ is a 4-to-1 multiplexing function and depending on d, it selects one of $x_i, i = 0, 1, 2, 3$ as its output.

The reformulated DFU is shown in Fig. 7(e). The DFU is divided into two parts, DFU 1 and DFU 2. The major part, DFU 2, which has a long chain of adders, is now isolated from both of the BMU and ACSU and is no longer on the critical path. It is possible that we can completely isolate the whole DFU by using pre-computation to DFU 1. However, it may be not necessary as after we apply the retiming and reformulation technique to the pipelined PDFD design in Fig. 4, the new critical path will be one which includes the 4D BMU, ACSU and SMU, similar to the design in [4]. However, the advantage of our design is low hardware overhead. In our design, we only need to compute 128 look-ahead 1D branch metrics, instead of 320. In addition, we only need to compute 32 ISI estimates instead of 160. The proposed design can lead to a reduction of hardware overhead at least by 60% for 1000BASE-T. If the proposed techniques are applied to applications employing PAM modulation with a larger symbol set, the complexity reduction will be even more.

5. REFERENCES

[1] M. Hatamian,et. al., "Design considerations for Gigabit Ethernet 1000Base-T twisted pair transceivers," *Proc. IEEE Custom Integrated Circuits Conference*, pp. 335-342, 1998.



(a) Original DFU and SMU

(b) Retiming Cutset 1

(c) Retiming Cutset 2

(d) Retimed DFU

(e) Reformulated DFU

Fig. 7. Retiming and reformulating the DFU

[2] P. R. Chevillat and E. Eleftheriou, "Decoding of trellis-encoded signals in the presence of intersymbol interference and noise"' *IEEE Trans. Commun.*, vol. 37, no. 7, pp. 669-676, July, 1989.

[3] M. V. Eyuboğlu and S. U. H. Quershi, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE J. Selected Areas Commun.*, vol. 7, no. 6, pp. 989-995, Aug. 1989.

[4] E. F. Haratsch and K. Azadet, "A pipelined 14-tap parallel decision-feedback decoder for 1000BASE-T Gigabit Ethernet," in *2001 Int. Symp. on VLSI Tech., Syst., and Appl.*, pp. 117-120, April, 2001.

[5] E. F. Haratsch and K. Azadet, "A 1-Gb/s Joint Equalizer and Trellis Decoder for 1000BASE-T Gigabit Ethernet," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 3, pp. 374-384, March 2001.