Architectural Techniques for Eliminating Critical Feedback Paths

Horng-Dar Lin, *Senior Member, IEEE*, and David G. Messerschmitt, *Fellow, IEEE*

***Abstract*—Circuits with feedback paths are significantly slower than comparable circuits without the feedback. The feedback also implies data dependency which voids usual parallel implementations, further exacerbating the throughput problem. This paper discusses a new high-throughput solution for systems with finite-level feedback values. As an example, we consider coding and signal processing systems for optical communications, which usually have very simple feedback. Our method uses architectural techniques, and requires no detail circuit tuning for high speedup. We demonstrate the method by realizing a 2 micron CMOS layout of a bimode 3B4B line coder. Simulation estimates that, using standard cell design, the chip achieves a coding rate of 1.4 Gb/s. Other design options are discussed.**

I. Introduction

OPTICAL communication systems are gradually adopting coding and signal processing techniques. For example, new (line) coding methods were proposed for reducing jitter [1], [2], introducing timing information [3], inserting an ancillary channel [4], facilitating FSK coherent transmission [5], [6], improving energy efficiency [7]–[9], monitoring or correcting errors [10]–[12], and limiting dc baseline fluctuations. (See [13]–[15] for more line code references, and [15] for classification of prior line codes.) Simple coding techniques like these are helpful because they can compensate nonideal characteristics of optical communications systems. Other coding applications include binary signaling of ternary or quaternary alphabets, data access control, and general user-defined applications.

In view of throughput characteristics, general coding and signal processing systems can be categorized into either feedforward or feedback systems. In a feedforward system, the input propagates through the circuits to the output along a unidirectional path. In contrast, a feedback system contains closed loops within the circuits, i.e., a feedback path exists between the output and the input.

A feedback path places a more stringent throughput limit than a feedforward unidirectional path. For example, consider a unidirectional path from the input to the output. Depending upon the circuits' state, the propagation delay may vary from a maximum τ_{max} to minimum τ_{min}. According to Fig. 1 (a), the circuit can operate correctly if the inputs are spaced at least $\Delta\tau = \tau_{max} - \tau_{min}$ apart [16]. Therefore, an upper bound on a feedforward system's throughput is $1/\Delta\tau$, which we call *latchless pipeline bound*. In a feedback loop, the propagation delay still may vary from τ_{max} to τ_{min}. The timing should always allocate τ_{max} for aligning the feedback with the next input, as shown in Fig. 1 (b). Thus, an upper bound on a feedback system's throughput is $1/\tau_{max}$, known as the *iteration bound* [17], [18]. The two bounds illustrate that the maximum achievable throughput of a feedback system is much less than that of a feedforward system with comparable complexity.

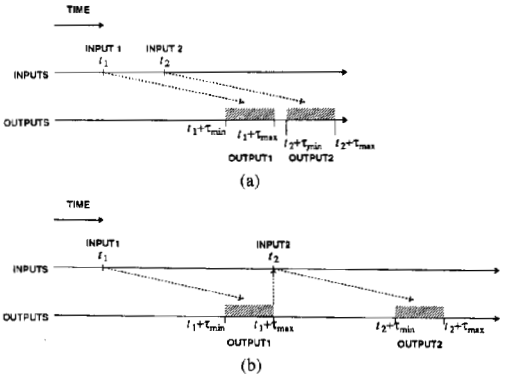


Fig. 1. The throughput upper bounds of feedforward systems and feedback systems. (a) A feedforward system can operate correctly if the output is nonoverlapping, i.e., $t_1 + \tau_{max} < t_2 + \tau_{min}$. (b) A feedback system uses the current output in the next input cycle. This implies $t_1 + \tau_{max} < t_2$.

The throughput discrepancy between feedforward and feedback systems becomes even larger in terms of available solutions. In feedforward systems, higher throughput it possible through parallelism. For example, since conventional block codes require no feedback, a coding rate higher than the $1/\Delta\tau$ bound is possible with interleaving, as shown in Fig. 2 (a), or with serial-parallel-serial conversions, as shown in Fig. 2 (b). The throughput rate of the parallel systems in Fig. 2 is ideally three times the individual rate of the block coders. However, the two well-known architectural techniques in Fig. 2 do not apply to feedback systems in general. As we will discuss in Section II, these parallel techniques lengthen the critical path and propagation delay of feedback systems, and thus counteract the advantages of parallelism.

Motivated by the above considerations, we describe in

Manuscript received June 5, 1990; revised March 5, 1991.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 91000127.

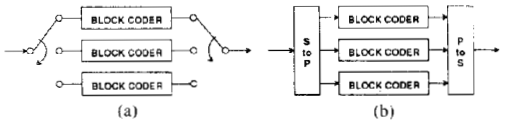


Fig. 2. Two basic architectural techniques for improving circuit throughput: (a) interleaving and (b) serial-parallel-serial conversions.

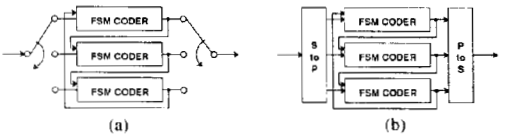


Fig. 3. Necessary modifications for maintaining the functionality of the parallel FSM coder systems in (a) interleaving and (b) serial-parallel-serial conversions.

this paper a new architectural technique that relaxes throughput bottlenecks for feedback systems in optical communications. Our technique applies to general coding and signal processing for discrete channels. The significance of the technique manifests in two ways. First, it makes feasible CMOS implementations of feedback systems at high data rates. Second, it enables applications of more sophisticated feedback systems. Both of these are important in integrating optical communications with user services.

The paper is organized as follows. Section II describes modified parallel techniques for feedback systems. After pointing out the limitation of these modified techniques, we present our new technique in detail. Finally, we demonstrate our method and explore design options with a 1.4 Gb/s 3B2T-RBS (bimode 3B4B) encoder using CMOS standard cell design.

II. Backgrounds and Modified Parallel Techniques

We can model feedback systems in optical communications as finite state machines (FSM). The feedback value is the state of the FSM, and the logic function that computes the next feedback is the state transition function of the FSM. $\tau_{\max}$ in the $1/\tau_{\max}$ bound then corresponds to the (worst-case) time for computing the next state.

A first-cut solution to improving the $1/\tau_{\max}$ bound is to start with the architectural techniques in Fig. 2. Unfortunately, replacing the block coders directly with FSM coders in Fig. 2 (a) does not generate correct coding functionality. To correct this, we must route the previous FSM's output to the next FSM's input, as illustrated in Fig. 3 (a). The input again should synchronize with the previous output. For example, as the input stream switches from the top FSM coder to the second FSM coder in Fig. 3 (a), the second FSM coder needs to wait for the top FSM's output to code the input. This implies the input rate is still $1/\tau_{\max}$, i.e., no throughput improvement at all. Similarly, modification of serial-parallel-serial conversions as in Fig. 3 (b) does not improve the throughput.

To overcome this functionality-throughput impasse, we can expand the FSM to one that processes multiple inputs simultaneously. Table I describes a one-bit-input, ternary-output FSM with a state transition diagram shown in Fig. 4. Expanding the FSM to a two-bit input FSM means making two consecutive state transitions in Fig. 4. After relabeling the arcs, we have a new FSM with the state transition diagram in Fig. 5. Because the new FSM has twice as many inputs per cycle, if the new FSM has the same $\tau_{\max}$ as the original, the throughput is $2/\tau_{\max}$, a speedup of two.

TABLE I
An Example FSM with a Binary Input and Ternary Output

Input	State	Output	Next State
0	S_0	0	S_0
1	S_0	+	S_1
0	S_1	+	S_1
1	S_1	−	S_2
0	S_2	−	S_2
1	S_2	0	S_0

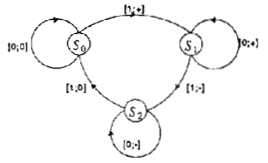


Fig. 4. An example FSM coder which takes binary inputs $\in \{0, 1\}$ and generates ternary outputs $\in \{+, -, 0\}$ according to the states. The nodes represent the states, and the arcs represent the state transitions associated with the [input; output] of their labels.

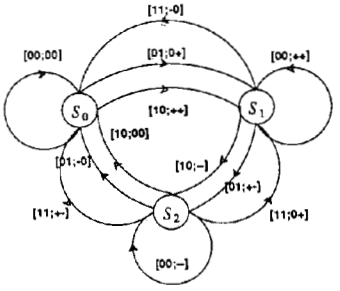


Fig. 5. The new FSM coder from expanding Fig. 4 by a factor of two. The inputs are two binary bits or a quaternary alphabet $\in \{00, 01, 10, 11\}$. The number of states remains the same.

From Fig. 5, we observe two properties that are generally true: 1) the number of states of the expanded FSM remains the same; and 2) the number of possible state transitions grows exponentially with the expansion factor. More specifically, suppose an N-state FSM has an input alphabet size A. If we expanded the FSM by K times, the new FSM still has N states, but the equivalent input alphabet size becomes A^K. The total number of state transitions, an indicator of the new FSM's complexity, is NA^K. Thus, expanding an FSM by K times increases the complexity by A^{K-1} times.

The increase in complexity inevitably lengthens the τ_{max}, and therefore counteracts the throughput improvement. In practice, two types of implementations with reasonable τ_{max} exist. The first type is to construct circuits directly from the new state transition diagram. The additional input wires and logics induce higher complexity and heavier circuit loading. As a result, τ_{max} increases, but not proportional to the expansion factor. The second is to tabulate for all combinations of K inputs in a PLA or memory. Similarly, τ_{max} increases due to additional loading. Empirical results show that for an expansion factor K, the increase in τ_{max} usually tracks between $O(\log(K))$ and $O(K)$ asymptotically. Circuit sizing can help reduce increases in τ_{max}[1].

The exponentially growing complexity and the increasing τ_{max} are two limiting factors to the expansion method. As a well-known architectural technique, the method has been successful for small throughput improvements in relatively simple FSM's. But in situations where the required speedup K is high or the alphabet size is large, we need a method with a complexity proportional to the speedup, rather than exponential, and with a τ_{max} insensitive to the increase in complexity. We will present such a method in the next section.

III. The Post-Selection Method

A. The Algorithm

Consider the operation of an FSM. As time evolves, the state transition sequence follows a path in a full tree, as illustrated in Fig. 6 (a). For example, if the first three inputs to the FSM coder in Fig. 4 are (0, 1, 0), the state transitions correspond to the darkened path in Fig. 6. Since the tree soon gets intractable after a few inputs, we prefer describing our method with trellises like Fig. 6 (b). A trellis Fig. 6 (b) results from merging identical nodes at the same tree level within Fig. 6 (a). Alternatively, we can generate the trellis by expanding the state transition diagram in Fig. 4 against the time index. The state transition sequence again maps to a path in the trellis, while each transition corresponds to extending the path by one arc from the current node to a node at the next stage.

The throughput limitation comes from data dependency between path extensions. That is, in order to extend the path, we need to know the current node position in the trellis, which in turn depends on the previous path extension. If data dependency in path extensions can be relaxed from between adjacent trellis stages to between K stages, different parts of trellis can be traversed in parallel. This way we can achieve a throughput higher than the $1/\tau_{max}$ bound.

More specifically, consider breaking trellis into blocks of length K, as shown in Fig. 7. Data dependency between processing different blocks can be relaxed by computing for all possible initial node positions for each block. Because path traversing always starts from every possible initial node position, traversing results from the previous block are not necessary. The parallelism is twofold—among different initial node positions and different blocks. Notice that the number of parallel blocks is unlimited in principle, which translates to a large speedup.

[1]Without sizing, τ_{max} increases rapidly after K grows beyond a threshold, which is when the expanded PLA overloads the precharge circuits.

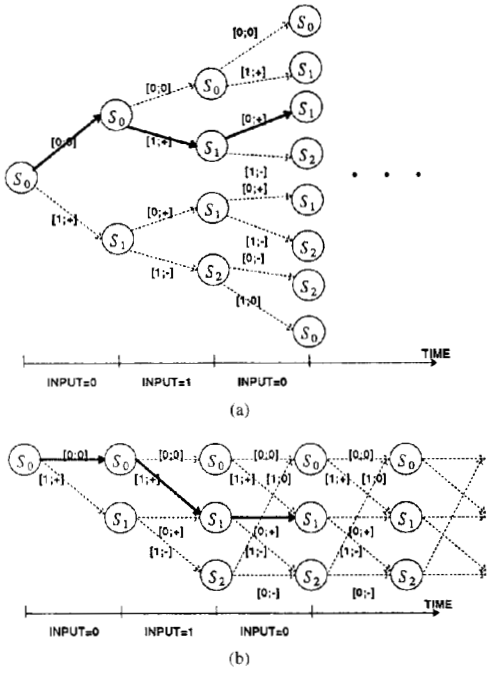


Fig. 6. Representation of possible FSM state transition sequences: (a) tree and (b) trellis. Here we use the example FSM in Fig. 4 with initial state S_0. The dark paths correspond to input (0, 1, 0).

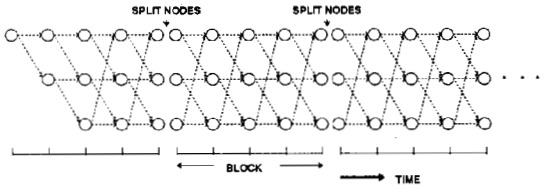


Fig. 7. The trellis in Fig. 4 is divided into blocks of length $K = 4$.

The algorithm starts with processing blocks of inputs in parallel, and generates for each block N path segments, one for each initial node, as illustrated in Fig. 8 (a). To determine the correct path, we need to link the path segments sequentially through the blocks. The linking only concerns the initial node and the ending node of each path segment, not the detail information within. Therefore, as shown in Fig. 8 (b), each path segment can be regarded as an arc between its initial node and ending node during linking. This is exactly the same formulation as the original path traversing, but now extending the path by one arc corresponds to processing one block of inputs. Because the processing in Fig. 8 (b) is the only part that contains data dependency, the throughput bound has been relaxed from $1/\tau_{max}$ to K/τ_{max}, where K is the block length.

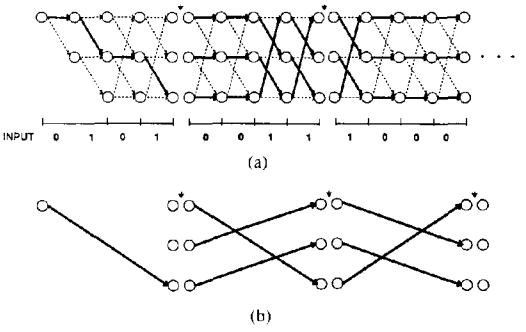


Fig. 8. An example for illustrating the algorithm. (a) The first level of processing is to find path segments for each block in parallel. (b) The second level of processing is to connect the path segments sequentially through the blocks.

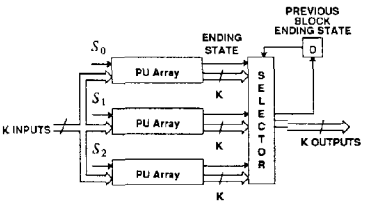


Fig. 9. The post-selection architecture for our example in Fig. 8.

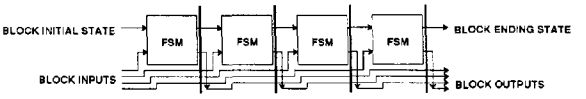


Fig. 10. The PU array for our example. Here we use four-stage pipelining, with each FSM as one stage.

We can verify that the above algorithm indeed achieves a K times throughput improvement. The throughput of block parallel processing is bounded by $1/\Delta\tau$, and can be further improved as in Fig. 2 if $1/\Delta\tau < K/\tau_{\max}$. The throughput of linking is again bounded by $1/\tau_{\max}$. Because the formulation is Fig. 8 (b) is the same as the original FSM, the $\tau_{\max}$ is also the same as the original. As each block contains results for K inputs, the throughput rate is K times better than the original.

The complexity of the algorithm depends on the implementation, which in turn depends on its architectures and level of hardware sharing. In terms of amount of computations, the algorithm requires $(N + 1/K)$ FSM operations per input for an N-state FSM with block length K. Interestingly, the overhead ratio $(N + 1/K)$ is insensitive to the speedup K, i.e., near $O(1)$. In some cases, the overhead can be reduced to (log $(N) + 1/K$) [19]. In the following, we show an architecture that fully exploits the efficiency of the algorithm.

B. The Architecture

As parallelism exists among processing of different blocks and different block initial nodes (states), the architecture can take many forms. Here we present a parallel pipelined architecture suitable for coding systems in optical communications.

The *post-selection* architecture in Fig. 9 implements the algorithm in Fig. 8. Each *processing unit array* (PU array) in Fig. 9 is a linear array that processes a block of K inputs for a specific block initial state. Each PU array generates K outputs and a block ending state, which are sent to a selector. The selector uses the previous block ending state to determine which PU array's output should be selected. That is, if the previous block ending state is S_0, then the selector chooses the output from the PU array with initial state S_0. In terms of our algorithm, the selector implements the linking process, whereas the PU arrays implement parallel block processing. While a PU array computes only for a specific block initial state, different blocks may share the PU array through pipelining.

The PU array is a multilevel logic circuit that transforms K inputs into the corresponding K outputs and block ending state for the given initial state. Although the PU array can be synthesized directly from its I/O description, the complexity is usually too high due to combinatorial inputs. A better implementation is to cascade multiple FSM's together with input and output skew, as shown in Fig. 10. Each FSM processes one stage of path extension: it takes the input of the corresponding stage and the state output from the FSM at the previous stage, and computes the new state value and the output. Notice that the longest feedforward path in the cascaded FSM is the state data path. However, because the cascade can be fully pipelined, the PU array will not become the throughput bottleneck. For example, each FSM can be decomposed into two-level AND–OR logic gates. Pipelining at the gate level is possible by inserting pipeline latches between the AND and OR gates. Although this is hardly necessary, it illustrates the real throughput limit is in the selector in Fig. 9.

The throughput of the post-selection architecture is K/τ_{sel}, where τ_{sel} is the (worst-case) propagation delay of the selector. The complexity is KN FSM's, plus pipeline latches and a selector. Therefore, we conclude the architecture indeed has a complexity proportional to the speedup, as opposed to the exponential complexity of the expansion method.

IV. Design Example

We use an example to demonstrate the method and its design options. Consider the 3B2T-RBS code, which uses the optimum 3B2T line code in Table II with *relative biphase signaling* (RBS) [1]. (The 3B2T stands for coding three binary inputs into two ternary outputs.) Relative biphase signaling (RBS) transmits a ternary signal with two binary signals in a state-dependent way. While [1] suggests implementing RBS for the 3B2T code with a four-state FSM, a simpler two-state FSM in Table III supports the same RBS rules. Therefore, the 3B2T-RBS code is equivalent to the bimode 3B4B line code in Table IV. The bimode code chooses different codewords according to the last encoded output bit, which is the feedback value

TABLE II
THE 3B2T LINE CODE IN THE 3B2T-RBS CODE

Input (binary)	Codeword (ternary)
000	02
001	12
010	01
011	22
100	11
101	10
110	21
111	20

TABLE III
RELATIVE BIPHASE SIGNALING

Input (ternary)	Codeword	
	Last Coded Bit = 1	Last Coded Bit = 1
0	10	01
1	11	00
2	01	10

TABLE IV
THE BIMODE 3B4B LINE CODE EQUIVALENT TO THE 3B2T-RBS CODE

Input	Codeword	
	Last Coded Bit = 0	Last Coded Bit = 1
000	1001	0110
001	1110	0001
010	1011	0100
011	0110	1001
100	1100	0011
101	1101	0010
110	0100	1011
111	0101	1010

or the state of the FSM coder. It has been shown that the bimode 3B4B code outperforms the 3B4B, and even the 5B6B code [1].

When implemented with a two-phase standard cell PLA, our simulation at the switch RC level estimates that the bimode 3B4B coder in 2 micron CMOS (lambda = 1 μm) runs at about a 70 MHz clock, or 210 Mb/s. ($\tau_{\max}$ = 13 ns for an $N = 2$, $A = 3$ FSM in our previous notation.) A speedup of seven ($K = 7$) can improve the coding rate to a 1.4 Gb/s optical rate.

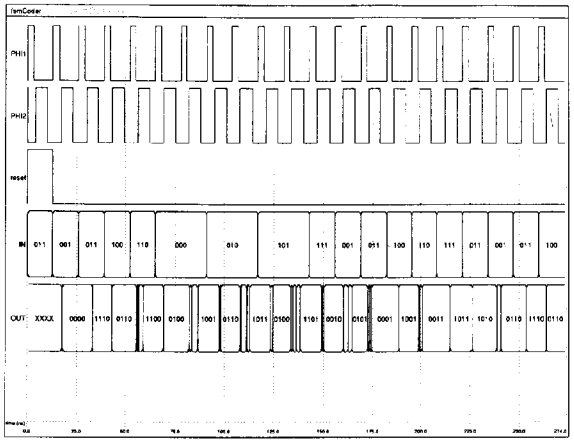
For comparison, we check the feasibility of the expansion method. When expanded by 2 and 3, the FSM becomes a bimode 6B8B coder and a bimode 9B12B coder, respectively. The bimode 6B8B coder and the bimode 9B12B coder are 3.1 and 12.1 times larger[2] than the bimode 3B4B coder, in fair agreement with the theoretical value $A = 3$ and $A^2 = 9$. Simulation shows that the clock rate drops to 50 MHz ($\tau_{\max}$ = 19 ns) for the bimode 6B8B coder and to 14 MHz ($\tau_{\max}$ = 68 ns) for the bimode 9B12B coder. The clock rate decreases because larger PLA's require longer time to precharge and evaluate, as we can see from the PHI1 (precharge) and PHI2 (evaluate) waveforms in Fig. 11 (a)–(c). Thus, the bimode 6B8B coder achieves a 6 * 50M = 300 Mb/s rate, which means a speedup of 1.4 for expansion factor 2. The bimode 9B12B coder runs at 9 * 14M = 126 Mb/s, which means no speedup at all for expansion factor 3 and higher. Thus, the limitation to the expansion method is more than just the complexity overhead. Unless we have fast precharge and evaluation circuits to keep up with the size increase of the PLA, a speedup higher than 1.4 cannot be achieved with direct FSM synthesis in this technology.

In contrast, our algorithm and post-selection architecture can achieve the seven times speedup easily without any FSM redesign. As the bimode 3B4B coder is a two-state FSM, the post-selection architecture needs two PU arrays, each of which has seven cascaded PLA's. The two PU arrays share the same input pipeline skew, as shown in the layout in Fig. 12. Our design uses Magic and Oct CAD tools with LagerIV design manager [20], and requires no physical chip editing like sizing transistors or defining feedthrough cells. Simulation estimates that the PLA clock remains at two-phase 70 MHz (13 ns). The input and output pipeline latches are also clocked at 70 MHz. The reason why we can maintain the same clock rate is that the PLA's in cascade have the same loading effects as the original PLA. The selector on the left runs faster than the pipeline clock ($\tau_{sel} < 10$ ns < 13 ns). Thus, the speedup is exactly seven times for a throughput of 7 * 3 * 70M = 1.4 Gb/s. (In fact, because we obtain the speedup through architectural techniques, the chip should always be seven times faster than the actual circuit speed of a bimode 3B4B coder.) The overall size is only 2.5 × 2.5 mm, which explains why we did not use hand design to minimize the chip area. We expect an area reduction of at least 30% with partial custom design.

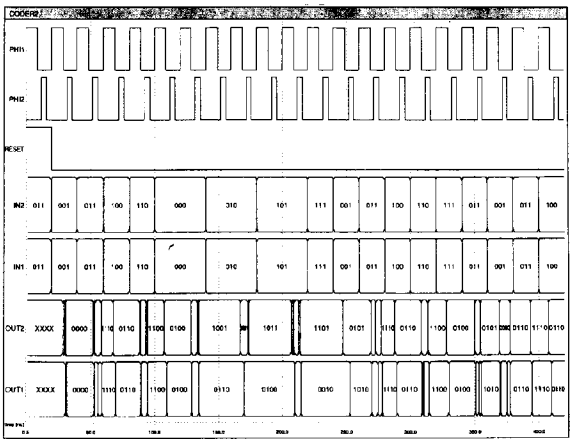
Alternative implementations with different clock rates are possible. To run at a faster clock rate, each FSM coder in the PU array can use two-level AND–OR gates with pipelining transmission gates between them. Clock rates at about 100 MHz are possible, but care must be taken in adjusting clock skew and compacting pipelined AND–OR gate cell. Although this design is more demanding, the improvement in clock speed means less cascaded stages in the PU arrays, less routing area, smaller multiplexer bit width, and less I/O pads.

Implementations at clock rates slower than the original coder's clock rate also exist. This approach is a combination of the post-selection method and the expansion method. For example, if we first expand the bimode 3B4B coder by two, the result is a bimode 6B8B PLA coder at 50 MHz. Each PU array in the post-selection architecture can cascade five bimode 6B8B coders in a pipeline for a throughput rate higher than 1.4 Gb/s. Although the number of cascaded coders is fewer, each coder is larger than the original. The low clock rate design is helpful if the original FSM coder is relatively simple. In this case, the clock rate slowdown and the PLA expansion is not serious, and we can take advantage of the PLA's compact-

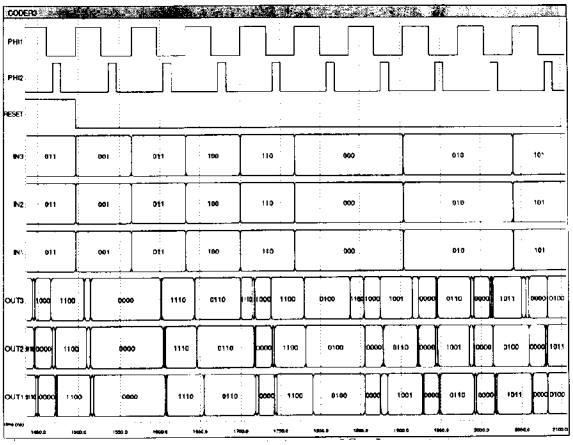
[2]We compare the core size of the coders.



(a)



(b)



(c)

Fig. 11. The simulated waveforms of: (a) the bimode 3B4B coder, (b) the bimode 6B8B coder, and (c) the bimode 9B12B coder displayed in logic analyzer form. Note that the cycle time for PHI1 (precharge phase) increases rapidly as the code complexity or PLA size exceeds a certain threshold.

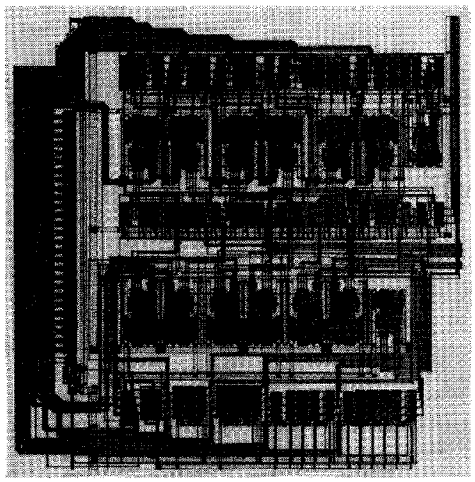


Fig. 12. The layout of the 1.4 Gb/s bimode 3B4B coder. The top PU array (cascaded PLA's and output skew pipeline latches) and the bottom PU array share the input pipeline skew in the center row. The top row and the bottom row are the output skew pipeline latches. The selector is on the left.

ness. However, if the clock rate is too slow, probably a separate clock and interface is necessary for chip I/O.

Summarizing, high speedup implementations exist with clock rates higher, lower, and the same as the original. The choice depends on applications. Generally speaking, if the original FSM is complicated it is advantageous to implement at higher clock rates. The dual argument is also true: a low-clock-rate implementation is suitable for simple FSM's. Otherwise, the post-selection architecture should cascade the original FSM's directly.

Previous discussion also applies to composite feedback systems. If a system contains multiple steps of FSM coding or multiple feedback loops, the design procedure is to apply our method to the throughput-limiting loop recursively. Alternatively, different loops can be combined as a big FSM or cascaded multilevel logics before the post-selection method is applied. This often leads to a better utilization of chip area and I/O.

V. A Solution for Complex Feedback Systems

Most feedback coding systems in optical communications have relatively few feedback states. However, in general, the number of states can be large, which renders the post-selection method ineffective. This suggests the following alternative algorithm.

Consider again the trellis blocks in Fig. 7. Instead of processing multiple path segments first and finding the right segment later, we change our strategy to finding the correct block initial state first before processing the block. That is, similar to the idea in [21], if the block initial states are known, then different blocks can be processed independently from their individual initial states. The algorithm then becomes:

1) Determine dependency between adjacent block initial states.

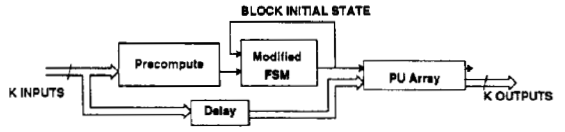


Fig. 13. An alternative architecture for systems with large numbers of states. Steps 1, 2, and 3 of the algorithm map to the precomputation, modified FSM, and the PU array, respectively.

2) Compute recursively the block initial states using the result in step 1.

3) Process each block from its initial state.

Note that both steps 1 and 3 are feedforward computations, meaning that higher throughput is (always) possible. Step 2 requires a feedback of the previous block initial state, and thus is the throughput-limiting step.

The architecture in Fig. 13 embodies the algorithm. Algorithm steps 1, 2, and 3 map to the precomputation, the modified FSM, and the PU array in Fig. 13, respectively. In step 1, dependency between the current and the next block initial state is a function of the K inputs in the current block. As each block initial state has at most N possible next block initial states, the dependency can be described with an N-level indicator (with log (N) bits). The precomputation circuits can be synthesized directly from its logic function, such as using multilevel AND–OR gates [19]. Step 2 itself is a new FSM whose input is the N-level indicator from the precomputation and whose state is the block initial state of the original FSM. Thus, the modified FSM in Fig. 13 has the same N states as the original FSM, but the input alphabet size is extended from A to N. Using the block initial state generated by the modified FSM, the PU array completes the block processing in step 3 using a cascade structure as before.

Because the precomputation and the PU array can be fully pipelined, the throughput of the architecture is $K/\tau_{\max^*}$, where $\tau_{\max^*}$ is the propagation delay of the modified FSM. Since the modified FSM is more complex than the original, the throughput improvement is less than K, or $K\,\tau_{\max}/\tau_{\max^*}$ to be exact, where $\tau_{\max}$ is the propagation delay of the original FSM.

The complexity of the precomputation and the modified FSM depends on applications. A more sophisticated complexity analysis and architecture details can be found in [19], wherein the resemblance and the mathematical relation between the FSM architecture in Fig. 13 and the well-known block-state filter structure is also discussed.

VI. Conclusion

We have shown architectural techniques for small-state feedback circuits that significantly improve the throughput without requiring circuit design efforts or advanced technologies. The method is flexible in terms of achievable implementations and speedups. For higher speedup and more complex feedback systems, our methods outperform the conventional expansion method in terms of speed and die area.

References

[1] O. N. Porokhov and P. Radev, "3B2T-RBS—An efficient transmission method for digital fibre-optic communications," *Electron. Lett.*, vol. 22, no. 21, pp. 1125–1126, Oct. 1986.

[2] E. V. Jones and S. Zhu, "Data sequence coding for low jitter timing recovery," *Electron. Lett.*, vol. 23, no. 7, Mar. 1987.

[3] Y. Takasaki, "Paired block line codes for pure photonic networks," *Electron. Lett.*, vol. 25, no. 4, pp. 256–257, Feb. 1989.

[4] R. Petrovic, "5B6B optical fibre line code bearing auxiliary signals," *Electron. Lett.*, vol. 24, no. 5, pp. 274–275, Mar. 1988.

[5] P. W. Hooijmans, M. T. Tomesen, and A. Van De Grijp, "Penalty free biphase linecoding for pattern independent FSK coherent transmission systems," *J. Lightwave Technol.*, vol. 8, no. 3, pp. 323–328, Mar. 1990.

[6] R. C. Steele and M. Creaber, "565 Mbit/s AMI FSK coherent system using commercial DFB lasers," *Electron. Lett.*, Feb 1989.

[7] M. C. Stefanovic, "Performance of signals coded by Miller code in the presence of noise and interference," *Electron. Lett.*, vol. 21, no. 16, pp. 667–668, Aug. 1985.

[8] R. J. McEliece, E. R. Rodemich, and L. Swanson, "An entropy maximization problem related to optical communication," *IEEE Trans. Inform. Theory*, vol. IT-32, no. 2, pp. 322–326, Mar. 1986.

[9] R. J. McEliece, "Practical codes for photon communications," *IEEE Trans. Inform. Theory*, vol. IT-27, no. 4, pp. 393–398, July 1981.

[10] N. Yoshikai, "Error-correcting code for optical-fibre transmission systems," *Electron. Lett.*, vol. 23, no. 3, pp. 97–98, Jan. 1987; comment/reply, p. 647, June 1987.

[11] M. Herro and L. Hu, "A new look at coding for APD-based direct-detection optical channels," *IEEE Trans. Inform. Theory*, vol. 34, no. 4, pp. 858–866, July 1988.

[12] J. H. Weber, "Bounds and constructions for binary block codes correcting asymmetric or unidirectional errors," Ph.D. thesis, Department of Electrical Engineering, Delft University of Technology, May 1989.

[13] E. E. Bergmann, A. M. Odlyzko, and S. H. Sangani, "Half weight block codes for optical communication," *AT&T Tech. J.*, vol. 65, no. 3, pp. 85–93, May–June 1986.

[14] N. Yoshikai, K-I. Katagiri, and T. Ito, "mB1C code and its performance in an optical communication system," *IEEE Trans. Commun.*, vol. COM-32, no. 2, pp. 163–168, Feb. 1984.

[15] N. Yoshikai, S. Nishi, and J-I. Yamada, "Line code and terminal configuration for very large-capacity optical transmission system," *IEEE J. Select. Areas Commun.*, vol. SAC-4, no. 9, pp. 1432–1437, Dec. 1986.

[16] D. G. Messerschmitt, "Synchronization in digital system design," *IEEE J. Select. Areas Commun.*, vol. 8, no. 8, pp. 1404–1419, Oct. 1990.

[17] D. A. Schwartz and T. P. Barnwell III, "A graph theoretic technique for the generation of systolic implementation of shift invariant flow graphs," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process.*, Mar. 1984.

[18] M. Renfors and Y. Neuvo, "The maximum sampling rate of digital filters under hardware speed constraints," *IEEE Trans. Circ. Syst.*, pp. 196–202, Mar. 1981.

[19] H.-D. Lin and D. G. Messerschmitt, "Finite state machines have unlimited concurrency," *IEEE Trans. Circ. Syst.*, vol. 38, 1991.

[20] LagerIV 1.3 Manual, *Lager Tool Set Vol. 1a-4*, Univ. of Calif., Berkeley, UCLA, Mississippi State Univ., Inst. of Technology Development, March 1989.

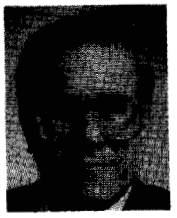
[21] H.-D. Lin and D. G. Messerschmitt, "Algorithms and architectures for concurrent Viterbi decoding," in *Proc. Int. Conf. Commun.*, June 11–14, 1989.

Horng-Dar Lin (SM'78) received the B.S.E.E. degree from the National Taiwan University in 1984, and the M.S. degree from the University of California, Berkeley, in 1988.

Currently, he is working toward the Ph.D. degree at the University of California, Berkeley. Between 1984 and 1986, he served as a Technical Officer in the United Forces in Taiwan working on communication electronics. As a Student Associate, he worked on efficient ASIC pipelining in the IBM Almaden Research Center between 1989

and 1990, and on optical networks in the IBM T. J. Watson Research Center in 1990. His research interests include VLSI design methodology, parallel algorithms, algorithm-to-architecture mapping, memory structures, and applications in high-speed communications and signal processing.

Mr. Lin was elected a member of the Phi Tau Phi scholastic honor society in Taiwan in 1984.



David G. Messerschmitt (S'65–M'68–SM'78–F'83) received the B.S. degree from the University of Colorado, Boulder, in 1967, and the M.S. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1968 and 1971, respectively.

He is currently a Professor of Electrical Engineering and Computer Sciences at the University of California, Berkeley. From 1968 to 1977, he was a Member of Technical Staff and later Supervisor at Bell Laboratories, Holmdel, NJ, where he was involved in systems engineering, development, and research of digital transmission and digital signal processing (particularly relating to speech processing). Since 1977, he has also served as a consultant to a number of companies. His current research interests include applications of digital signal processing, digital communications (on the subscriber loop and fiber optics), architectural approaches to dedicated-hardware digital signal processing (with a current emphasis on video compression applications), and computer aided design of communications and signal processing systems. He has published over 100 papers and two books and has 10 patents.

Dr. Messerschmitt is a member of Eta Kappa Nu, Tau Beta Pi, Sigma Pi, and has several best paper awards. He has served as a Senior Editor of the IEEE COMMUNICATIONS MAGAZINE, as Editor for Transmission of the IEEE TRANSACTIONS ON COMMUNICATIONS, and as a member of the Board of Governors of the Communications Society. He has also organized and participated in a number of short courses and seminars devoted to continuing engineering education.