VLSI Design
1994, Vol. 2, No. 1, pp. 33–50
Reprints available directly from the publisher
Photocopying permitted by license only

High Throughput Error Control Using Parallel CRC

ANDRZEJ SOBSKI and ALEXANDER ALBICKI
Department of Electrical Engineering, University of Rochester, Rochester, NY 14627

(Received September 18, 1992, Revised May 28, 1993)

Redesigning the LFSR (Linear Feedback Shift Register) so that syndrome calculations can be performed in one sweep allows for fast error control in high speed computer networks. The resulting structure forms the basis of the PEDDC (Parallel Encoder, Decoder, Detector, Corrector) which replaces the conventional Serial Encoder, Decoder, Detector, Corrector for generation and utilization of cyclic codes. Since syndromes are calculated in as little as one clock period, information from which the syndrome is calculated can be processed in a parallel stream. In this paper a simple PEDDC is built, its operation is examined in detail, its performance is compared with a serial counterpart, possible variations on the PEDDC structure is given, and further speed enhancement techniques are considered.

Key Words: *CRC schemes; LFSR; Parallelism; VLSI design; Performance Evaluation*

1. INTRODUCTION

Cyclic codes are often used in computer networks for their high error detection qualities and potential error correcting capabilities. The error control scheme for cyclic redundancy checking is illustrated in Figure 1. The code set used in the transfer of information is determined by a generator polynomial, $G(X)$, which is known to both sender and receiver. At the transmitter, the message, $M(X)$, is converted to a unique code word, $C(X)$. Before reaching the receiver, $C(X)$ may change due to noise or atmospheric interference and, therefore, is received as $R(X) = C(X) \oplus E(X)$, where $E(X)$ is an error polynomial. Successful decoding implies that $R(X)$ is converted to $M(X)$. Syndromes are used as parity check information for messages sent between transmitter and receiver. If the extracted syndrome $S(X) \neq 0$, then $E(X) \neq 0$ and an error is detected. In the correcting process, $R(X)$ is converted to the most probable code word, $R'(X)$ [1, 5].

Syndrome generation in the sender and receiver is possible with the LFSR (linear feedback shift register). The LFSR accepts and deposits information in a serial manner with a great amount of latency between bits due to flip-flop delays. If a parallel scheme replaces the LFSR as the basic building block, then the effective latency between bits no longer includes register delays as in the LSFR case. Hence, one would expect an increase in encoding, decoding, detecting, and correcting rate [2].

To demonstrate the advantage of a parallel scheme, we have designed a particular parallel encoder, decoder, detector and corrector (PEDDC) VLSI chip. This paper, in essence, shows the theoretical development, design procedures, and performance curves derived along the way which could serve as a handbook for the development of any PEDDC. In general, we show that a given PEDDC is larger but always faster than its corresponding LFSR; thus, one must weigh the speed degradation of the latter with the area overhead of the former when deciding which is better for a given application. This paper weighs these considerations and through the use of performance curves guides the designer in choosing an optimum error control strategy.

The report is organized in the following way. Section 2 describes LFSR-based error control and shows the compatibility between the PEDDC and the LFSR. Section 3 analyzes the design of a particular PEDDC VLSI chip. Section 4 discusses methods of improving the PEDDC through partitioning. Finally, Section 5 discusses the derivation and use of figure of merit curves in obtaining "the best" PEDDC configurations.

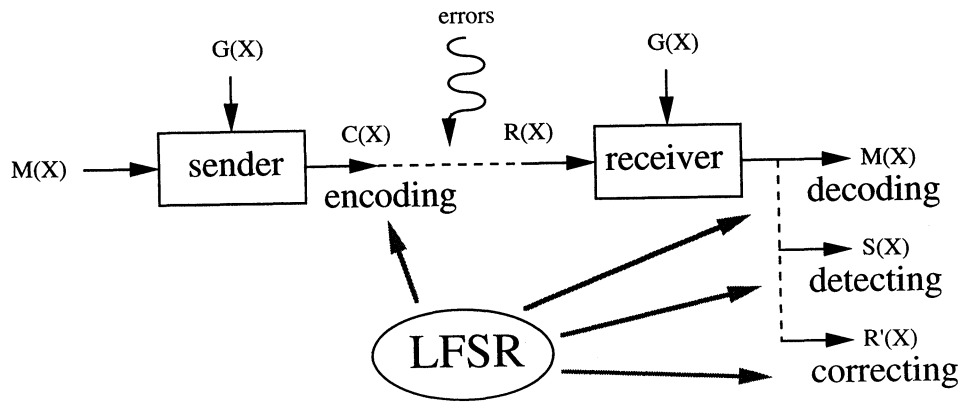


FIGURE 1 A conventional error control scheme with the LFSR serving as the basic building block.

2. THE EQUIVALENCE OF THE SERIAL AND PARALLEL SYNDROME COMPUTATION

In this section we consider just the encoding process. The decoding, detecting and correcting processes are similar to encoding in the sense that the basic operation employed—the polynomial division—is the same [3, 4, 6, 7].

The encoding of messages into code words by the LFSR scheme is described in terms of bit flow in Figure 2. The inputs and outputs to the scheme are paired with a particular time instant to show the order in which data is processed. The k bit long message,

$$M(X) = d_{k-1}X^{k-1} \oplus d_{k-2}X^{k-2} \oplus \ldots \oplus d_0,$$

is multiplied by X^{n-k} in order to zero pad it to n bits. Each bit in the $X^{n-k}M(X)$ shifted message stream causes the formation of a single bit in the n bit long code word,

$$C(X) = d_{k-1}X^{n-1} \oplus d_{k-2}, \ldots \oplus d_0 X^{n-k} \oplus s_{n-k-1}X^{n-k-1} \oplus s_{n-k-2}X^{n-k-2} \oplus \ldots \oplus s_0.$$

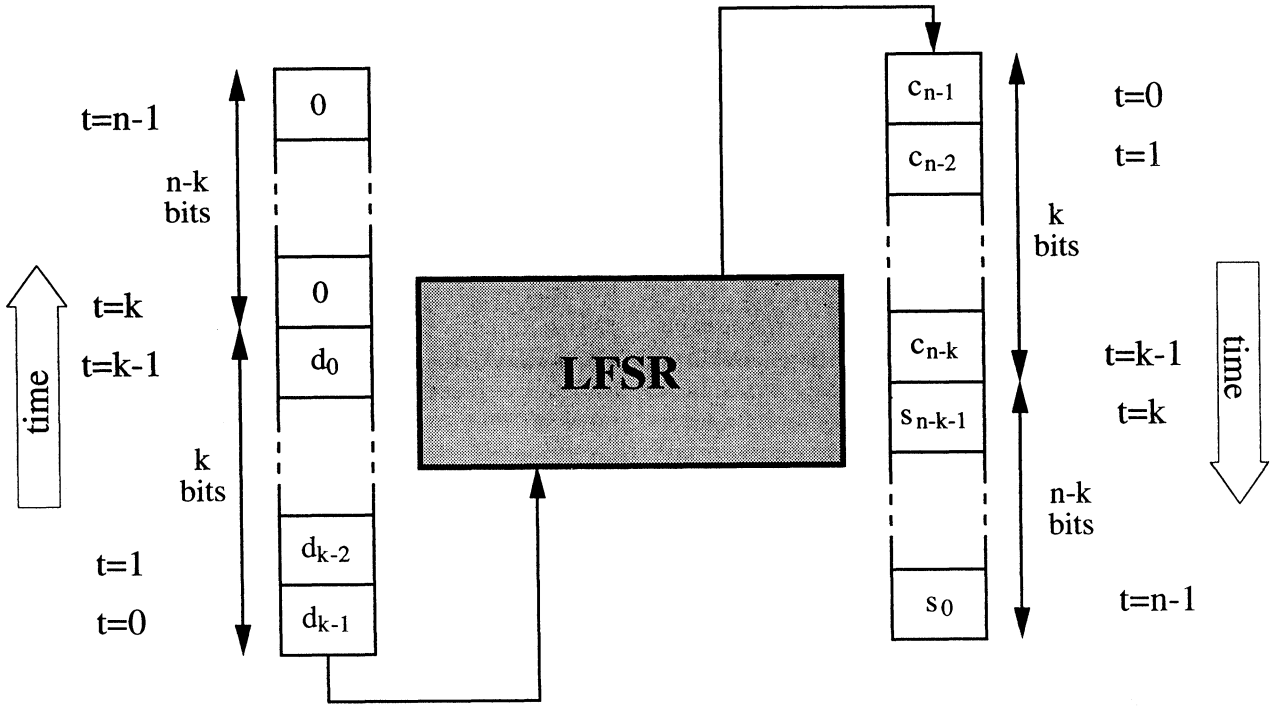


FIGURE 2 The LFSR-based bit flow diagram showing the encoding of $M(X) = (d_{k-1}, d_{k-2}, \ldots d_0)$ to $C(X) = (d_{n-1}, d_{n-2}, \ldots d_{n-k}, s_{n-k-1}, \ldots s_0)$.

$$
\begin{array}{l}
\qquad\qquad \overbrace{d_3}^{L^0}X^3 \oplus \overbrace{(d_2 \oplus g_2 d_3)}^{L^1}X^2 \oplus \overbrace{(d_1 \oplus g_1 d_3 \oplus g_2(d_2 \oplus g_2 d_3))}^{L^2}X \\
\qquad\qquad\qquad \oplus \overbrace{(((d_0 \oplus g_0 d_3) \oplus g_1(d_2 \oplus g_2 d_3)) \oplus g_2(d_1 \oplus g_1 d_3 \oplus g_2(d_2 \oplus g_2 d_3)))}^{L^3} \\
X^3 \oplus g_2X^2 \oplus g_1X \oplus g_0 \overline{)\, d_3X^6 \oplus d_2X^5 \oplus d_1X^4 \oplus d_0X^3} \\
\qquad\qquad L^0X^6 \oplus g_2L^0X^5 \oplus g_1L^0X^4 \oplus g_0L^0X^3 \\
\qquad\qquad \overline{L^1X^5 \oplus (d_1 \oplus g_1L^0)X^4 \oplus (d_0 \oplus g_0L^0)X^3\}} \quad \text{State 1} \\
\qquad\qquad\qquad L^1X^5 \oplus g_2L^1X^4 \oplus g_1L^1X^3 \oplus g_0L^1X^2 \\
\qquad\qquad\qquad \overline{L^2X^4 \oplus ((d_0 \oplus g_0L^0) \oplus g_1L^1)X^3 \oplus g_0L^1X^2\}} \quad \text{State 2} \\
\qquad\qquad\qquad\qquad L^2X^4 \oplus g_2L^2X^3 \oplus g_1L^2X^2 \oplus g_0L^2X \\
\qquad\qquad\qquad\qquad \overline{L^3X^3 \oplus (g_0L^1 \oplus g_1L^2)X^2 \oplus g_0L^2X\}} \quad \text{State 3} \\
\qquad\qquad\qquad\qquad\qquad L^3X^3 \oplus g_2L^3X^2 \oplus g_1L^3X \oplus g_0L^3 \\
\qquad\qquad\qquad\qquad\qquad \overline{(g_0L^1 \oplus g_1L^2 \oplus g_2L^3)X^2 \oplus (g_0L^2 \oplus g_1L^3)X \oplus g_0L^3\}} \quad \text{Syndrome, State } \underset{k}{4}
\end{array}
$$

FIGURE 3 Example of long-hand division of an $n - k$ bit shifted $M(X) = (d_3, d_2, d_1, d_0)$ by $G(X) = (1, g_2, g_1, g_0)$.

The ordering of the bits is such that as the syndrome $(S(X) = s_{n-k-1}X^{n-k-1} \oplus \ldots \oplus s_0)$ is deposited onto the LFSR output, zeroes are entered at the input. This bit flow diagram suggests two ways of measuring the throughput of the CRC scheme: the time between consecutive code bits on the LFSR output or the time between consecutive n-bit message encodings.

We can now examine the division operation in structured detail in order to relate it with the LFSR function. By decomposing the division operation and assigning time dependent variables to each step, it is possible to show its iterative nature and to define a basic building block of this iterative structure.

Figure 3 shows what is required in the long-hand Euclidean mod-2 division of a $k - 1$ $(=3)$ degree shifted message, $M(X)$, by an $n - k$ $(=3)$ degree generator polynomial,

$$G(X) = X^3 \oplus g_2X^2 \oplus g_1X \oplus g_0,$$

TABLE I
Progression of States in an LFSR Designed for a (7, 4) Cyclic Code

Time Instant (i)	LFSR Flip-Flop States ($\mathbf{S}^i$)
0	$S_0^0 = 0$ $S_1^0 = 0$ $S_2^0 = 0$
1	$S_0^1 = g_0 \overbrace{d_3}^{L^0}$ $S_1^1 = g_1L^0$ $S_2^1 = g_2L^0$
2	$S_0^2 = g_0 \overbrace{(d_2 \oplus S_2^1)}^{L^1}$ $S_1^2 = g_1L^1 \oplus S_0^1$ $S_2^2 = g_2L^1 \oplus S_1^1$
3	$S_0^3 = g_0 \overbrace{(d_1 \oplus S_2^2)}^{L^2}$ $S_1^3 = g_1L^2 \oplus S_0^2$ $S_2^3 = g_2L^2 \oplus S_1^2$
4	$S_0^4 = g_0 \overbrace{(d_0 \oplus S_2^3)}^{L^3}$ $S_1^4 = g_1L^3 \oplus S_0^3$ $S_2^4 = g_2L^3 \oplus S_1^3$

TABLE II
Progression of States in an LFSR Designed for a (n, k) Cyclic Code

Time Instant (i)	LFSR Flip-Flop States ($\mathbf{S}^i$)
i for $i = 0, 1, \ldots, k$	$S_0^i = g_0 \overbrace{(d_{k-i-1} \oplus S_{n-k-1}^{i-1})}^{L^{i-1}}$ $S_1^i = g_1L^{i-1} \oplus S_0^{i-1}$ $S_2^i = g_2L^{i-1} \oplus S_1^{i-1}$ $\vdots$ $S_{n-k-1}^i = g_{n-k-1}L^{i-1} \oplus S_{n-k-2}^{i-1}$

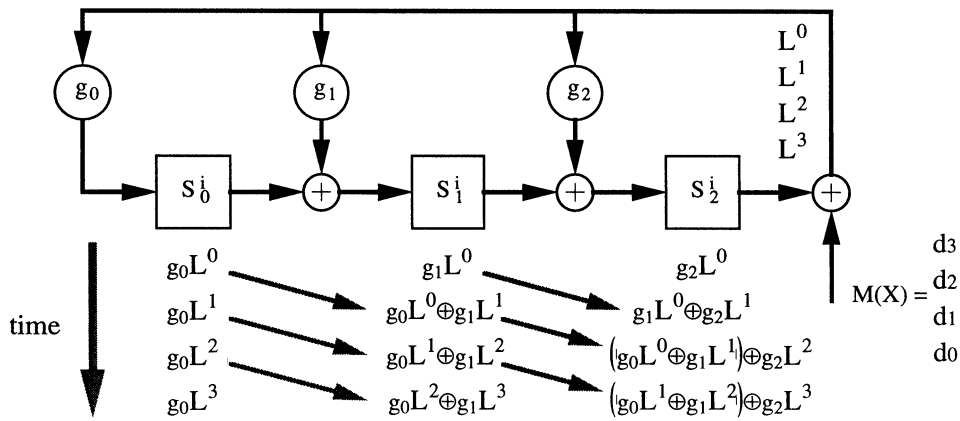


FIGURE 4 The state of the major nodes and flip-flops in an LFSR designed to encode a (7, 4) cyclic code.

to form an $n - 1$ ($=6$) degree code word. The results are specific for a (7, 4) cyclic code but the same pattern applies for any (n, k) cyclic code.

Mapping the progression of data through the LFSR during the k initial shifts in the encoding process, one can find the state values of the LFSR flip-flops in terms of the coefficients, g_i, and the variables, L^i, which are $L^i = d_{k-1} \oplus S^i_{n-k-1}$ where S^i_j is a state of flip-flop j ($0 \le j \le n - k - 1$) at time i ($0 \le i \le k$) inside the LFSR.

The state of each flip-flop in a LFSR which generates a (7, 4) cyclic code is reported in Table I. The state of each flip-flop for a (n, k) code is determined in Table II. As row $i = 0$ of the table shows, all flip-flops are initialized to zero. Before time $i = 1$, the inputs of each flip-flop (for $0 \le j \le n - k - 1$) evaluate to the most significant bit of $M(X)$ ($=d_3$) depending on the value of g_j. At time $i = 1$, the flip-flop inputs during state $\mathbf{S}^0 = (S^0_2, S^0_1, S^0_0)$ latch into the flip-flops and become the current $\mathbf{S}^1 = (S^1_2, S^1_1, S^1_0)$ (the second row in Table I). The table is filled accordingly. The assignment for S^1_0 in the 2nd row of the table equates d_3 with the variable L^0. Due to the recursive structure of the LFSR, this assignment is necessary to reduce the unwieldy relationships that otherwise would be placed under each column. In addition, this L^i assignment reveals the relationship between long hand mod-2 division and the operation of the LFSR, which is:

$$L^i = d_{k-1} \oplus S^i_{n-k-1} \text{ for each } 0 \le i \le k.$$

Comparison of Table I with Figure 3 shows that $\mathbf{S}^4$ of the LFSR is identical to the remainder obtained in Euclidean mod-2 division. However, this similarity does not exist in the states $\mathbf{S}^{4-1}$ through $\mathbf{S}^0$. The LFSR gradually converges to the correct division as represented by Figure 3 as time progresses.

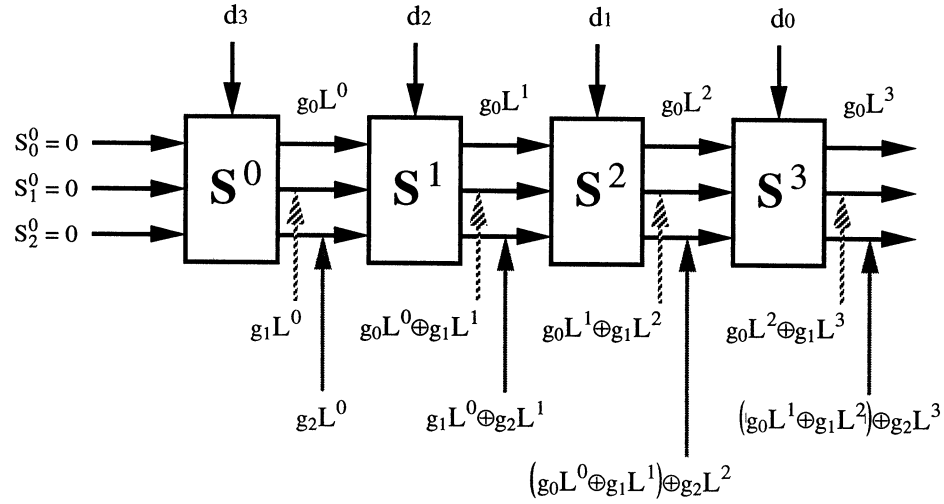


FIGURE 5 The state of the syndrome stages in a PEDDC designed to encode a (7, 4) cyclic code.

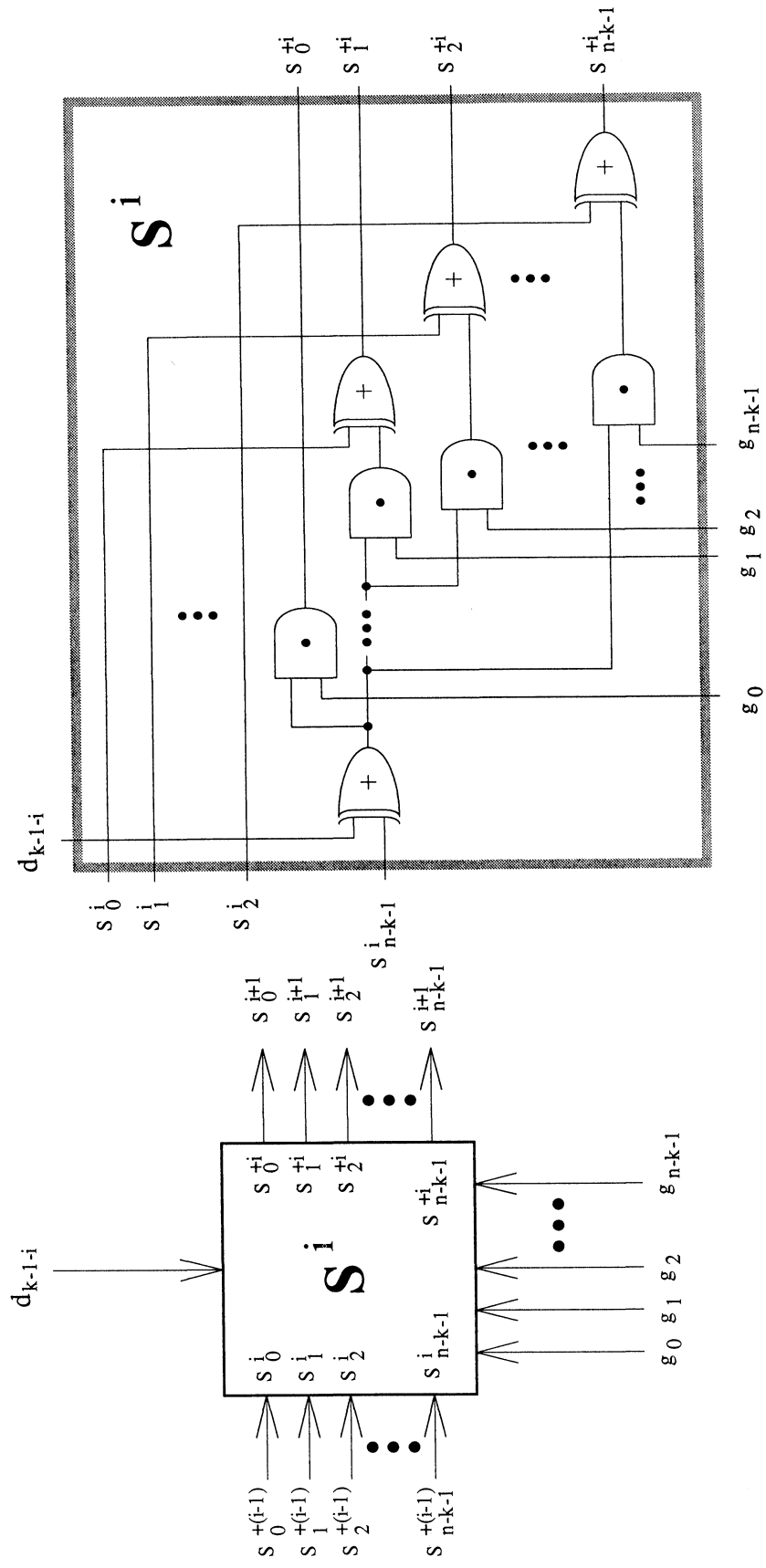


FIGURE 6 The basic building block of a PEDDC: the syndrome stage.

Equivalent to the LFSR, the parallel structure could be defined by Table II. Since it is desired that the PEDDC produce the syndrome in one time instant, the table can not be interpreted with respect to time as with the LFSR. One possibility is to have each row in the table represent the set of syndrome outputs from each stage in an iterative network. This is illustrated in Figures 4 and 5 where the output from each successive iterative stage in the PEDDC is equivalent to the contents of the LFSR for each time instant. Since every stage of the PEDDC calculates $\mathbf{S}^i$, this ensures the equivalence of the PEDDC and the LFSR.

The iterative stage of the PEDDC is shown in Figure 6. The notation s_j^i refers to the value of the j'th bit of the syndrome state preceding the i'th iterative block (the output of the i'th iterative block corresponds to the output from the syndrome registers in the LFSR following the i'th latching). Also, s_j^{+i} refers to the value of the j'th bit of the syndrome at the output of the i'th iterative block. These notation conventions are similar to those established for the LFSR with the exception that i no longer represents time, but rather, space. Note that this stage construction always suffers no less than 2 XOR and 1 AND gate delays independent of the size of the generator polynomial.

To realize a PEDDC encoder for a (n, k) cyclic code one must concatenate k stages of the form of Figure 6 such that the $n - k$ S_j^i outputs of one stage connect with the corresponding S_j^{+i} inputs of the next stage. The individual bits of $M(X)$ are fed into each stage from the top in sequence with the most significant bit placed in stage 0. The last stage produces and deposits the syndrome onto the last set of S_j^i outputs. The order of information flow in the encoder is illustrated in Figure 7.

The error correcting circuitry for a PEDDC is similar to the circuitry used in a comparable LFSR with the exception that it is repeated $n - 1$ times. For details refer to [7].

Figure 8 shows the configuration of a fully functional PEDDC. The syndrome from encoding, $S(X)$, appears at the output of stage $k - 1$ (not pictured); the syndrome from decoding, $S'(X)$, appears at the output of stage $n - 1$. The error correction circuitry becomes necessary in the second step of correcting (as in the LFSR), hence, it is connected to the outputs of stages $n - 1$ through $2n - 2$ (note that the outputs of stage $n - 1$ mark the end of decoding and the beginning of correcting just as time instant $n - 1$ marks the same transition in the LFSR [3, 6, 7]).

During encoding, the message bits, d_{k-1} through d_0, are entered at the PEDDC inputs, r_{n-1} through r_{n-k-1}. During error detection or correction, the bits of the received message,

$$R(X) = r_{n-1}X^{n-1} \oplus r_{n-2}X^{n-2} \oplus \ldots \oplus r_0$$

are loaded at the top of the first n PEDDC stages. During error correction, the error correction stage outputs, Out, are placed at the top of the next $n - 1$ PEDDC stages. At the same time, the bits of $R(X)$ are placed in the n error correction stages where they are stripped of errors.

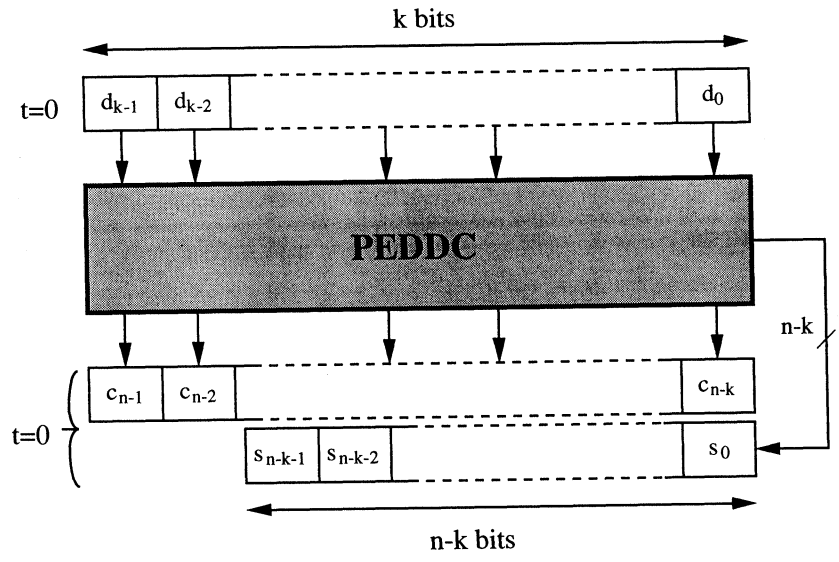


FIGURE 7 The PEDDC-based bit flow diagram showing the encoding of $M(X)$ to $C(X)$.

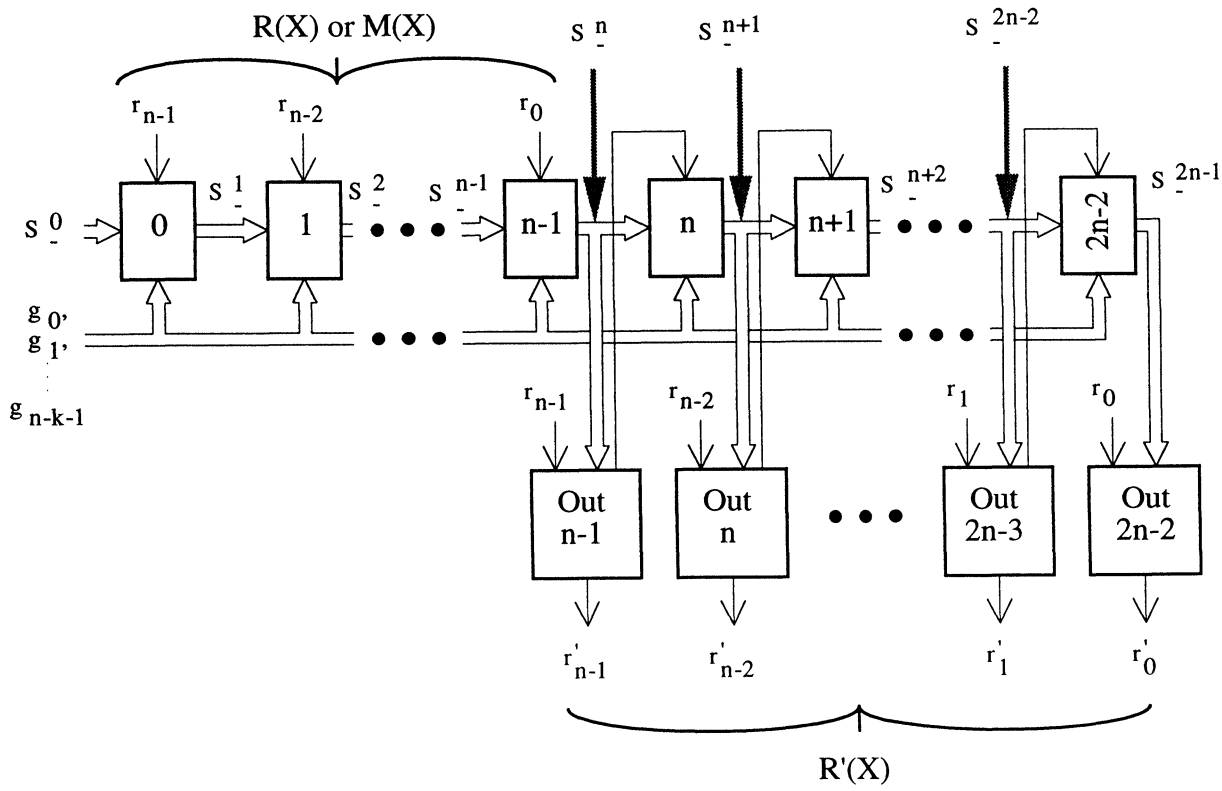


FIGURE 8 The PEDDC organization.

3. VLSI IMPLEMENTATION

Since the PEDDC is a regular structure which requires considerable amounts of logic to realize for practical applications, it seems natural that a PEDDC should be designed as a VLSI chip. In this section, the design and simulation of a simple PEDDC in 2um CMOS technology is discussed. The chip is built to support (7, 4) cyclic codes generated with $G(X) = X^3 \oplus X \oplus 1$. Though the PEDDC may be optimized in terms of size through partitioning, an unpartitioned PEDDC chip is built since its simplicity allows for easy extraction of performance parameters for any PEDDC.

3.1 Function Verification

In this section, logic verification of the PEDDC encoding and correcting functions is illustrated with a digital simulator, *Design Works*. Since decoding and detecting of errors are subfunctions of error correction, they are verified as well.

Figure 9a,b shows extracts of timing diagrams which test the PEDDC operation. In Figure 9a, each possible message, $M(X) = (d_3, d_2, d_1, d_0)$ is encoded to a corresponding $C(X)$ after one clock period. In effect, the message during the previous clock period is sent out on the output lines (c_6, c_5, c_4, c_3) followed by the syndrome, (c_2, c_1, c_0), which is computed for that message. In Figure 9b, a random sample of code words with single bit errors enter the PEDDC inputs, $R(X)$, at time = -40, 100, 180, 260, and 320 ns. After one clock period, the PEDDC corrects the erroneous code vector and deposits the correct version on its outputs, $R'(X)$. At time = 425 ns, a valid code word enters the PEDDC and is appropriately left unchanged at the PEDDC outputs.

3.2 Complexity

In this section, the complexity in terms of chip area of various components of the (7, 4) PEDDC VLSI chip is discussed. This analysis leads to the conclusion that the overhead is mostly due to the interconnect between iterative blocks.

The (7, 4) PEDDC VLSI chip contains a total of 880 transistors which occupy a rectangular area of 1000×4000 μm^2. Approximately 20% of this rectangular area is un-used due to the L-shaped geometry of the composite circuitry which has been

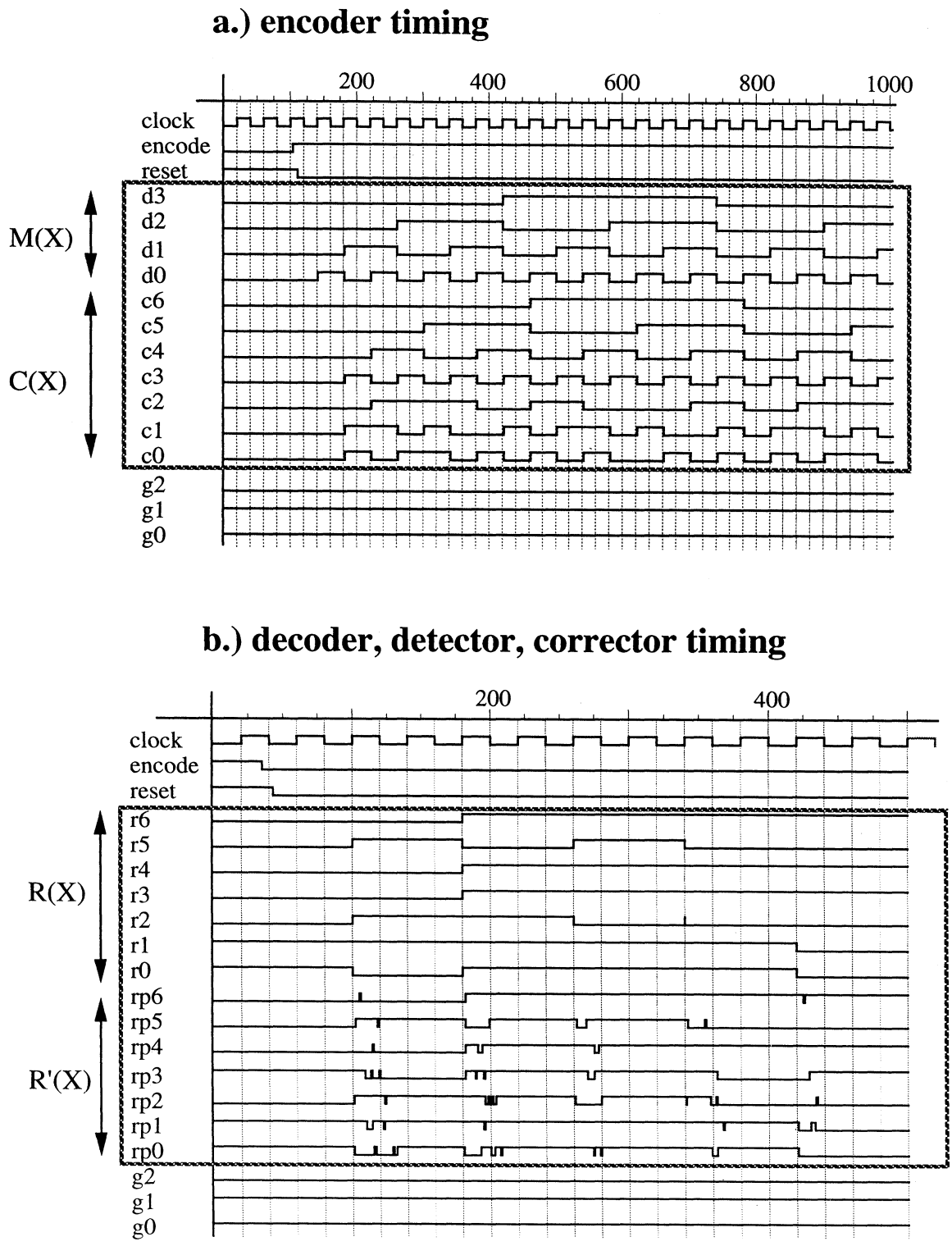


FIGURE 9a,b The timing diagrams for a PEDDC based on the (7, 4) cyclic dode with $G(X) = 1011$.

optimized as much as possible. All subsequent percentages will be calculated with respect to this used layout area (i.e. 3.2 mm^2). The transistors have been arranged so that a PEDDC chip which contains additional $\mathbf{S}^i$ stages for larger cyclic codes results in the same area utilization percentage (i.e. 80%). The buffers, repeaters, and drivers throughout the chip occupy a rectangular area of 8×10^5 μm^2 or 25% of

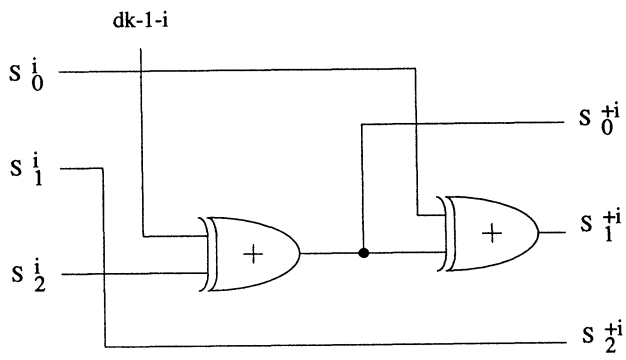


FIGURE 10 The simplified syndrome stage based on the (7, 4) cyclic code with $G(X) = 1011$.

the total circuit area. Due to the conservative circuit structure, this percentage is expected in all PEDDC chips of differing sizes. Each $\mathbf{S}^i$ stage and error correcting stage built for this code occupies an area of 150×300 μm^2 and 150×200 μm^2, respectively. Since there are $2n - 1$ ($=13$) and n ($=7$) replicas of these stages, respectively, the PEDDC combinational logic occupies 8×10^5 μm^2 or 25% of the total circuit area. This implies that in a double-level metal and single polysilicon CMOS process, the interconnect and routing occupy approximately 50% of the circuit area. Overall, this percentage can be expected of PEDDC chips built for different codes.

The circuit was built conservatively in order to allow for regularity in the design and easy expandability to higher order PEDDCs. For instance, buffers were placed at strategic nodes in the (7, 4) PEDDC to take into account the fact that in larger PEDDCs these nodes would experience greater loads. This proved useful for it allowed for accurate extrapolation of the design to higher order PEDDCs in order to yield data on the speed/area performance given different generator polynomial degrees.

3.3 Encoding, Decoding, Detecting, and Correcting Rate

To accurately simulate the performance of the $\mathbf{S}^i$ stage of the PEDDC chip with SPICE 3, it is necessary to load its outputs with the input capacitances of $\mathbf{S}^{i+1}$ and the capacitance of the input of the error correcting stage associated with $\mathbf{S}^i$. The $\mathbf{S}^i$ stage in Figure 7 is simplified by fixing the generator polynomial coefficients to $(g_2, g_1, g_0) = (0, 1, 1)$ (see Figure 10). Since the simulation complexity of the entire PEDDC is beyond the capability of SPICE 3, one replica of the circuit fragment containing $\mathbf{S}^i$ and $\mathbf{S}^{i+1}$ and the error correcting stage following $\mathbf{S}^i$ (Figure 11) could be used to obtain the overall performance of the PEDDC. All interstage buffers that attach to the PEDDC fragment (not shown in Figure

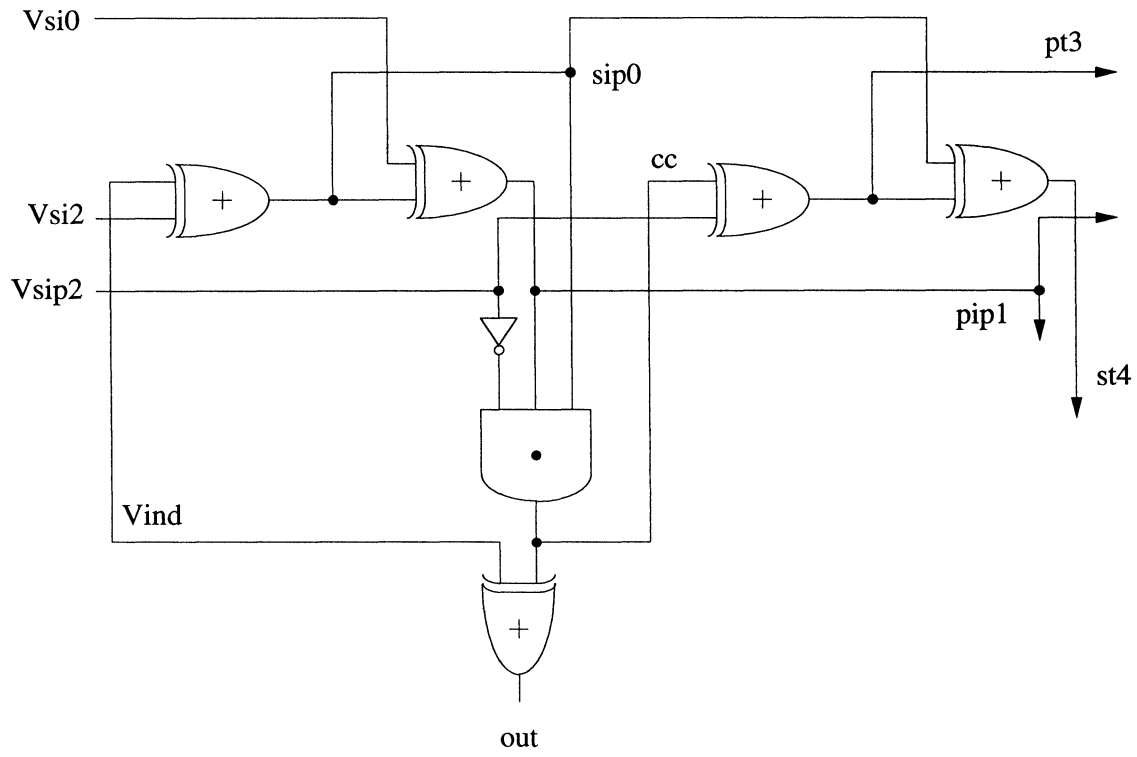


FIGURE 11 The PEDDC fragment which is used for the SPICE transient analysis.

11) are illustrated with triangles in the equivalent transistor schematic of Figure 12. The inputs to $\mathbf{S}^i$ are Vsi0, Vsi2, Vsip2, and Vind, and the outputs are sip0 and pip1. The outputs from the error correcting stage are out and cc; the outputs from $\mathbf{S}^{i+1}$ are pt3 and st4. Both Figures 11 and 12 are repeatable fragments that can be concatenated to form the entire PEDDC. That is, lines pip1 and st4 attach to the error correcting stage of $\mathbf{S}^{i+1}$, nodes pt3 and Vsi0 both attach to the sip0 input of $\mathbf{S}^{i+2}$, etc.

The maximum delay through the PEDDC is determined by the path from Vsi2 to pip1 (abbreviated {Vsi2, pip1}) which is multiplied by n since it is replicated and concatenated n times as it traverses the entire PEDDC. Note, as Figure 11 shows, this path skips over $\mathbf{S}^{i+1}$ and, in general, traverses through every other stage. The first $n/2$ replications of this path occur through the n stages of the PEDDC devoted to encoding and decoding—the final $n/2$ replications occur in the final $n - 1$ stages of the PEDDC devoted for detecting and correcting. These final $n/2$ replications experience the greatest propagation delay due to an additional error correcting stage load. Ignoring the specific delays associated with the parasitics and loads, and instead, lumping these together with the propagation delay through the basic gates, the total critical path delay is equivalent to $2n$ XOR delays $\approx 4n$ gate delays.

From Figure 12, the number of minimum size transistor gate capacitances, N_{Cg}, and drain capacitances, N_{Cd}, that must be charged or discharged in the traversal of the critical path (excluding the capacitances associated with the buffers 3:6 and super-buffers 3:6:12:24) is $N_{Cg} = 18$ and $N_{Cd} = 12$. If one assumes that traversing through 3:6 requires charging and discharging an effective $N_{Cg} = 4$ and $N_{Cd} = 4$ and that traversing through 3:6:12:24 is equivalent to traversing through two 3:6 stages, then $N_{Cg} = 30$ and $N_{cd} = 24$ for {Vsi2, pip1}. The N_{Cg} and N_{Cd} approximation is based on the fact that scaling minimum sized transistors increases the drain capacitance by the same factor as reducing the drain resistance; therefore, the effective delay of a buffer is approximated by assuming it is composed of purely minimum sized transistors.

Figure 13 displays the SPICE 3 transient analysis for the PEDDC fragment of Figure 12. The transient analysis is graphed for all possible input combinations to the PEDDC fragment. Vind is equivalent to a digit of $R(X)$ while out is the corresponding digit of $R'(X)$. $R'(X)$ follows $R(X)$ whenever $R(X)$ is determined to be a code word. Thus, an error in Vind is detected at 400 and 550 ns. Whenever cc evaluates to 5 volts, an error is detected at Vind and corrected at out. Vsi0, Vsi2, Vsip2, and Vind are the control inputs corresponding to s_0^i, s_1^i, s_2^i, and d_{k-1-i} of Figure 9. They are represented by the pulse generators in Figure 12. The outputs, pt3, pip1, st4, and out, are associated with the second syndrome stage, $(\mathbf{S}^{i+1})$, and the error correcting stage. These nodes are used in the determination of the delay through various paths in the PEDDC. The outputs aa, bb, and cc correspond to the inputs/outputs of the buffer/super-buffer pair connected to the largest capacitive node in the PEDDC.

As mentioned earlier, the critical path in Figure 12, {Vsi2, pip1}, is repeated N $(=7)$ times in the entire PEDDC. The critical delay associated with this path is measured from the vertical dotted line, where there is a change in the inputs, to the point of stabilization of pip1. This delay is represented by a shaded region in Figure 13. As can be seen from the size of the shaded regions in the figure, pip1 takes at most $\tau = 25$ ns to stabilize for every two $\mathbf{S}^i$ stages. Since there are 13 $\mathbf{S}^i$ stages in the entire correcting process and 7 bits in $R(X)$, the PEDDC VLSI chip can correct code words transmitting at a frequency of:

$$\frac{\text{bits of } R(X)}{\text{critical delay through PEDDC}} = \frac{n}{(2n - 1) \times \frac{\tau}{2}}$$

$$= \frac{7}{13 \times 12.5 \text{ ns}}$$

$$= 43.1 \text{ MHz.}$$

Similarly, since there are 4 $\mathbf{S}^i$ stages in the encoding process and 7 bits in $C(X)$, the PEDDC chip can encode messages and transmit code words at a rate of:

$$\frac{\text{bits of } M(X)}{\text{critical delay through encoder}} = \frac{n}{k \times \frac{\tau}{2}}$$

$$= \frac{7}{4 \times 12.5 \text{ ns}}$$

$$= 140 \text{ MHz.}$$

Finally, since there are 7 $\mathbf{S}^i$ stages in the decoding/detecting process and 7 bits in $R(X)$, the PEDDC

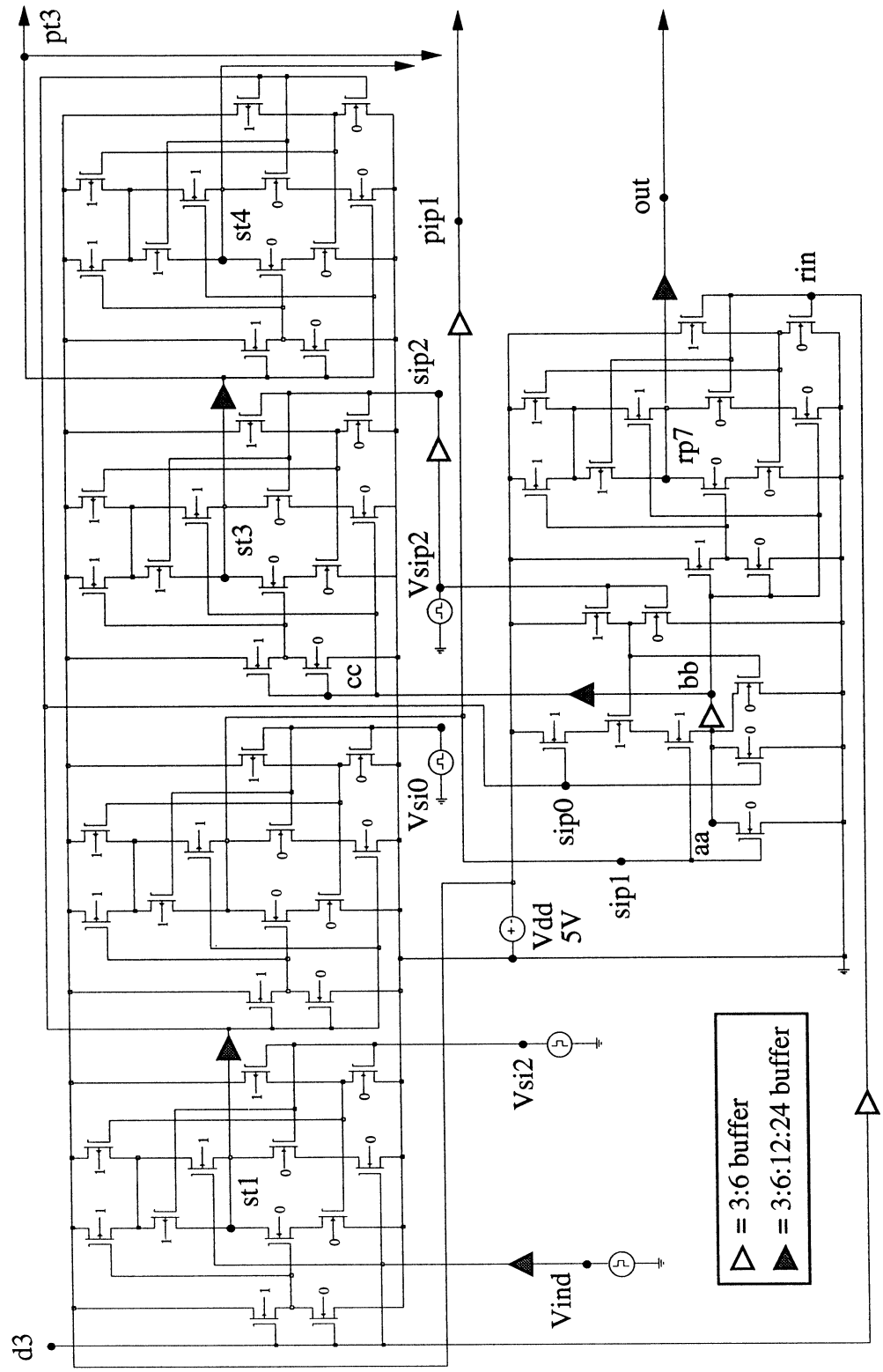


FIGURE 12 The transistor schematic of the PEDDC fragment which is used in the SPICE transient analysis.

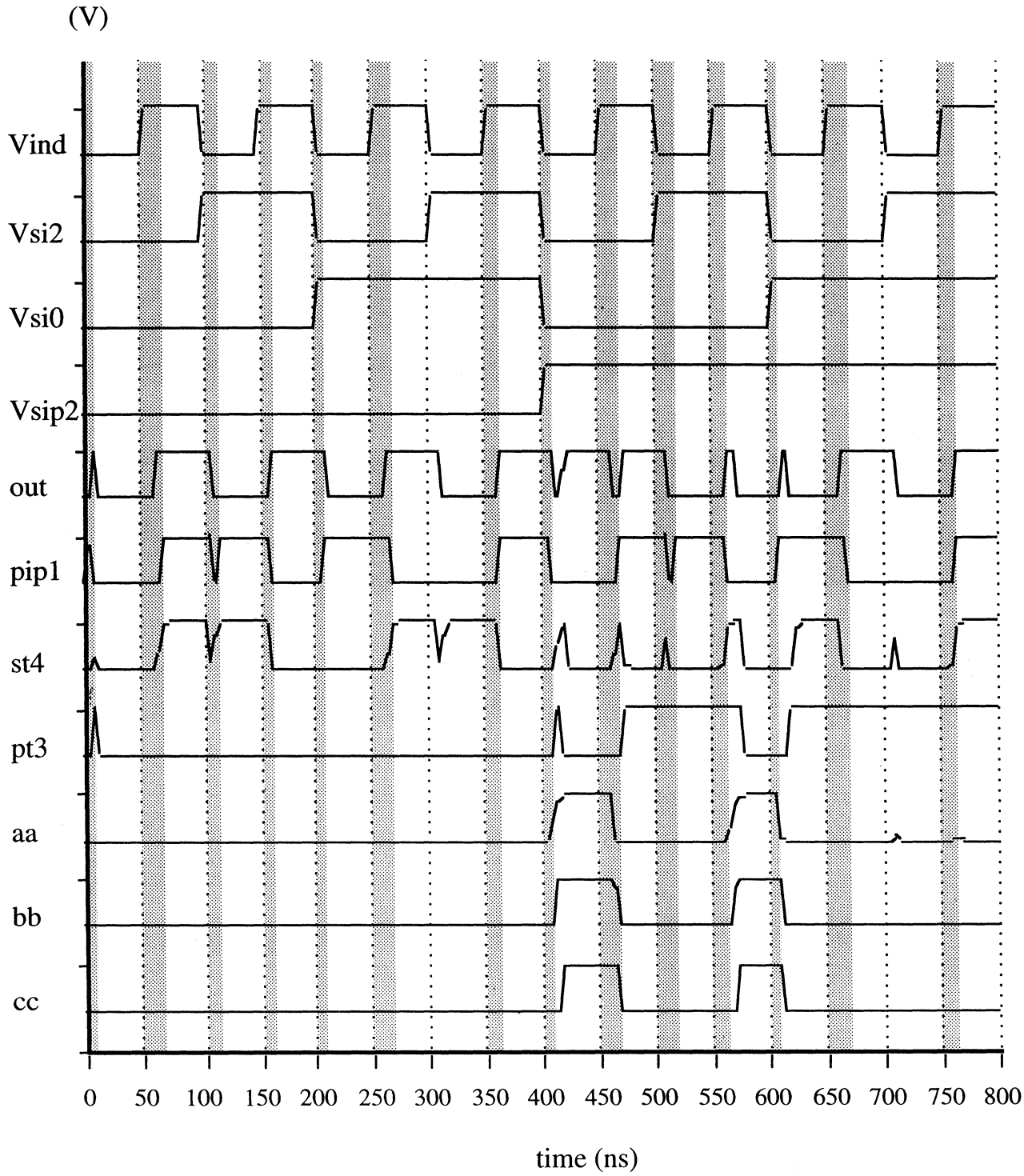


FIGURE 13 Timing plot of the SPICE transient analysis of the PEDDC fragment.

chip can decode and detect errors while receiving code words transmitting at a rate of:

$$\frac{\text{bits of } R(X)}{\text{critical delay through decoder}} = \frac{n}{n \times \frac{\tau}{2}} = \frac{7}{7 \times 12.5 \text{ ns}} = 80 \text{ MHz}.$$

4. pPEDDC CONSTRUCTION

Since the size of the PEDDC grows exponentially with the degree of the generator polynomial, the PEDDC must be modified before it may be used in practical situations. For example, in fiber optic LANs generator polynomials are of degree equal to 32 requiring a PEDDC with 8 million stages [2, 4, 7]. This section describes a method of optimizing the PEDDC so that it becomes tractable for large generators. The performance of this refined PEDDC can easily be extracted from the (7, 4) PEDDC chip.

Since a code word is too long to allow parallel loading at once, the PEDDC can be parallel loaded several times to re-use certain stages in the PEDDC over again. Such a PEDDC is called an r bit partitioned PEDDC (pPEDDC) where r refers to the maximum number of bits that can be loaded at one time. Figure 14 demonstrates the procedure by which a partitioned PEDDC converts a message into a code word. Encoding one message takes approximately as many clock pulses as the message size, k, "integer divided" by the number of parallel bits, r. The length of the partitioned PEDDC is chosen to be the basic symbol size or any convenient size that matches the length of the parallel information that may be made available to it at once.

If the PEDDC is partitioned too much so that only one syndrome stage of the PEDDC is used over again, the pPEDDC will reduce to simply a LFSR. Obviously, the more syndrome stages re-used per clock period the fewer clock periods (by a similar factor) required to encode, decode, detect, and correct. However, it can be shown that the period of the pPEDDC clock grows by a smaller factor, than the number of stages added [7]. Thus, there is more to gain in speed and lose in size for increasing the number of pPEDDC stages. Unfortunately, the complexity in terms of control increases as a result of partitioning in order to synchronize the loading of sections of $M(X)$ and $R(X)$. It is possible to find an optimum number of stages where there are diminishing returns for added stages to increase speed (i.e. where the ratio of speed increase to size increase is less than one) [7].

The pPEDDC is composed of r $\mathbf{S}^i$ stages and r error correcting stages. These stages are reused as many times as is necessary for encoding, detection, and correction. This is possible due to the addition of a feedback register, multiplexing circuitry, and a buffer register. The pPEDDC requires $\lceil k/r \rceil$, $\lceil n/r \rceil$, and $\lceil 2n - 1/r \rceil$ time instants for encoding, detection, and correction, respectively. For all time instants but the last, the outputs of stage $\mathbf{S}^{r-1}$ are fed back to stage $\mathbf{S}^0$ during which the next set of r bits of $M(X)$ or $R(X)$ are loaded into the top of all r $\mathbf{S}^i$ stages. In the last time instant, the remaining k (**mod** r) (for encoding), n (**mod** r) (for detection) or $2n - 1$ (**mod** r) (for correction) bits are loaded into the right-most $\mathbf{S}^i$ stages while stage $\mathbf{S}^{r-1}$ feeds back information to the appropriate $\mathbf{S}^i$ stage. The operation of the error correcting circuitry is similar for a partitioned PEDDC and a regular PEDDC.

5. LFSR, PEDDC, AND pPEDDC SIZE, SPEED, POWER, AND ENERGY COMPARISONS

It has been shown that the PEDDC is functionally equivalent to the LFSR, so deciding which is better would require weighing the costs of speed (the PEDDC and pPEDDC) vs. size (the LFSR). Figures 15a,b show the trade-offs between the LFSR, PEDDC, and pPEDDC when built for some typical codes (i.e. CRC-12, CRC-16, CRC-CCITT, etc.) as well as some limiting condition codes. On the horizontal axis, the curves refer to codes by their generator polynomial degree, m ($=n - k$), rather than their name. Any code formed from a generator polynomial with degree $3 \leq m \leq 32$ with at least one-half of the generator polynomial coefficients equal to zero is reflected in these curves. The figures provide speed and size in terms of transistors and gate delays (extrapolated from the PEDDC chip [7]), rather than microns and seconds, making the relationships in the charts technology independent. However, it is possible to convert these technology independent parameters to μm^2 and nsec with respect to the CMOS $2\mu m$ double-metal process. That is, given a PEDDC built from a third degree generator polynomial, the area utilization for 880 transistors is 3200 μm^2 (from Section 3.2) and the latency between bits for encoding is less than 12.5 nsec (from Section 3.3). Thus, from Figure 15, one gate delay

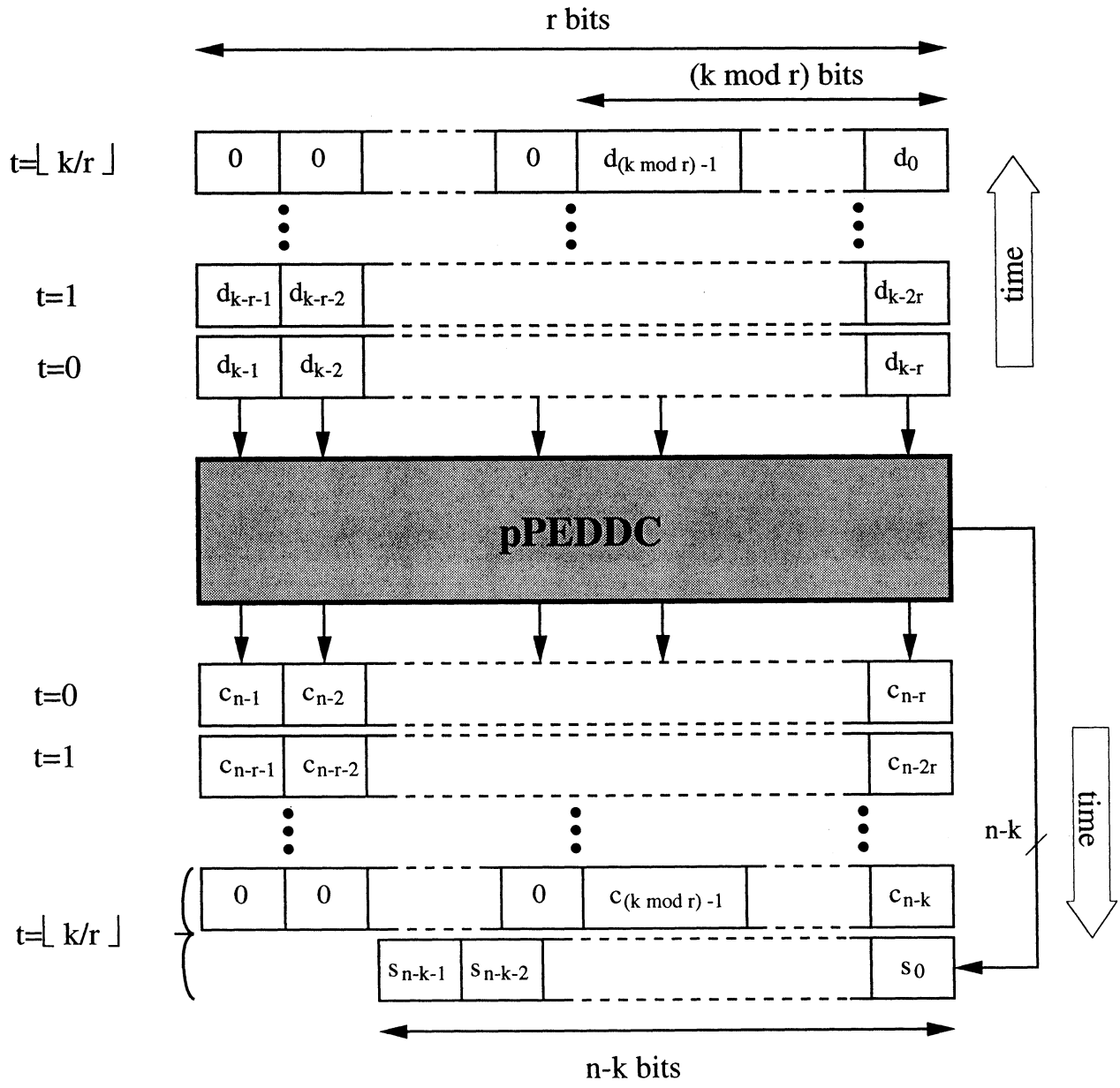


FIGURE 14 The pPEDDC-based bit flow diagram showing the encoding $M(X)$ to $C(X)$.

is 2.5 nsec and 1000 transistors occupy an area of 3.6 mm^2 for a PEDDC and pPEDDC (note: since the structure of the LFSR-based error control circuit is different from the pPEDDC/PEDDC, this transistor area equivalence does not hold for the LFSR).

The pPEDDC curves in these figures are representative of pPEDDCs built with $r = m$. Since there are $2n - 1 = 2^{m+1} - 2$ possible values for r in a pPEDDC, the curves are not reflective of all pPEDDCs. Nevertheless, since the $\mathbf{S}^i$ stages are interconnected between each other with m bit buses, it seems practical to use m bits as the common bus size throughout the pPEDDC.

From Figure 15a one can conclude that a pPEDDC with $r = m$ is on the average 10 times larger than a comparable LFSR. The size of the PEDDC is charted to illustrate that an enormous improvement in size can be achieved with a negligible loss in speed (Figure 15b) by partitioning a PEDDC. It also makes clear the intractability of building a PEDDC for most codes. It should be noted that the pPEDDC curve approaches the LFSR curve in Figure 15a while all curves in Figure 15b remain nearly unchanged when $r < m$, and hence, the size of the pPEDDC may be optimized considerably with an insignificant loss in speed [7]. Figure 15b lists down the right edge in

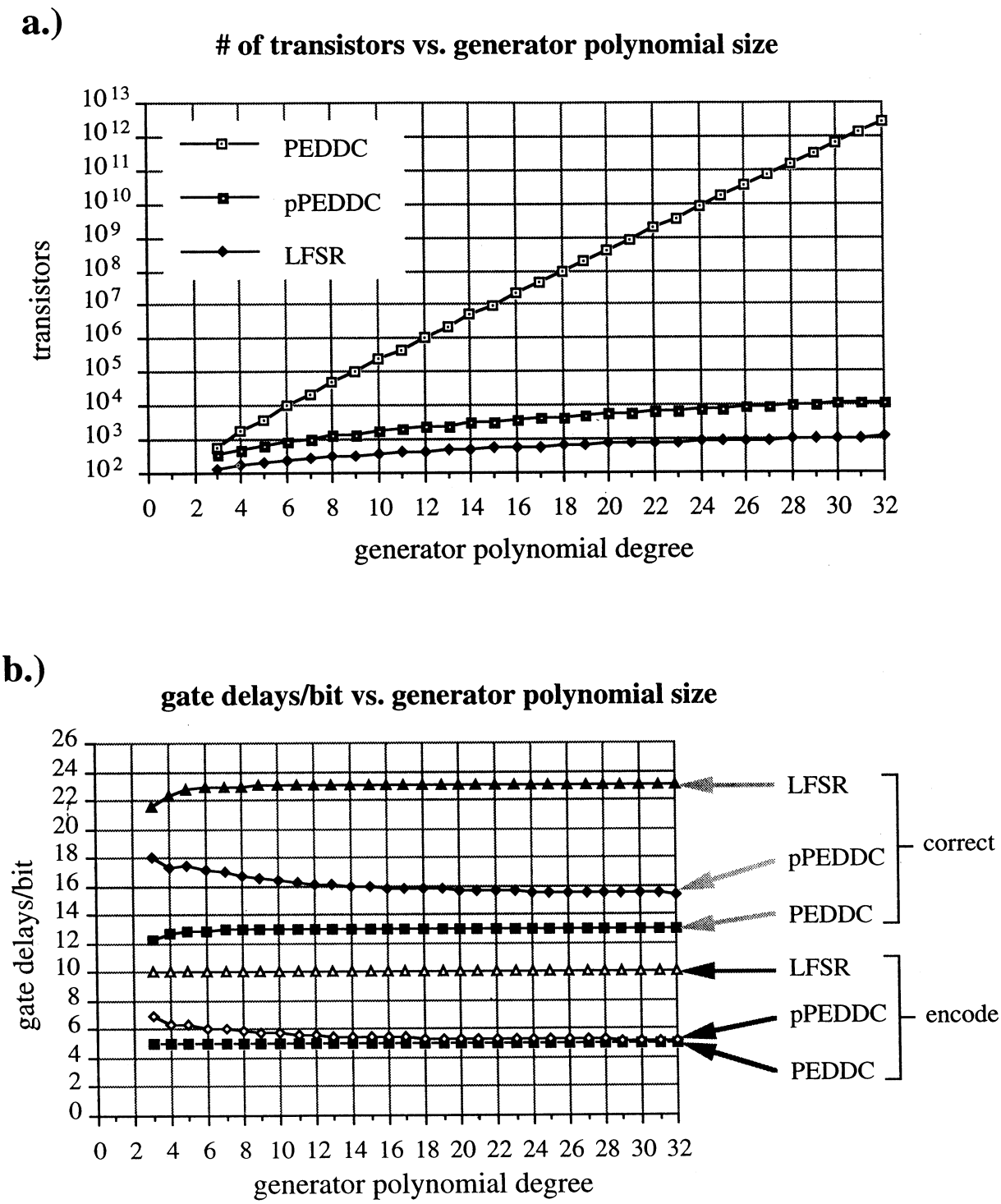


FIGURE 15a,b Size and speed comparisons of the LFSR, PEDDC, and pPEDDC.

relative order: the slowest to fastest circuit and associated mode of operation. Since all the curves asymptotically level off for increasing m, the gate delays/bit associated with $m = 32$ gives the best estimate for the average speed of every LFSR, PEDDC, and pPEDDC. The LFSR spends on average 23 gate delays per bit to correct versus 13 gate delays per bit for the PEDDC. Only 15 gate delays are needed for a pPEDDC to correct. Note: for large m the pPEDDC has nearly comparable speed per-

formance to the idealistic PEDDC. That is, it takes approximately 5 gate delays per bit for both to encode (for $m > 10$) versus 10 gate delays for an LFSR. It takes only 15% more time for the pPEDDC to correct than the PEDDC. Thus a pPEDDC is not very far from a PEDDC in terms of its advantages. For most practical applications of an LFSR, $m \geq 16$, the pPEDDC is a very attractive substitute for the LFSR.

In terms of power dissipation and energy usage, the pPEDDC and PEDDC are better than the LFSR-based error control circuit for almost all types of generators, especially larger degree generators. This is illustrated in Figure 16, where the dynamic power dissipation and energy usage of the pPEDDC/PEDDC is graphed as a fraction of the power dissipation and energy usage of the LFSR. As before, the pPEDDC curves are plotted for $r = m$. These curves assume switching activity occurs at all transistors for each clock cycle. Since the power dissipation is proportional to the number of switching transistors divided by the clock frequency, and since there is r and n ($=2^m - 1$) times more bit calculations per clock cycle in the encoding/decoding process of the pPEDDC and PEDDC compared to the LFSR, respectively, the power dissipation with respect to the LFSR for encoding is calculated as:

$$\text{Power\_Dissipation}_{\text{pPEDDC:LFSR\_encode}} = \frac{\text{transistors}_{\text{pPEDDC}}/(r \times \text{gate\_delays}_{\text{pPEDDC\_encode}})}{\text{transistors}_{\text{LFSR}}/\text{gate\_delays}_{\text{LFSR\_encode}}}$$

$$\text{Power\_Dissipation}_{\text{PEDDC:LSFR\_encode}} = \frac{(\text{transistors}_{\text{PEDDC}}/2)/(n \times \text{gate\_delays}_{\text{PEDDC\_encode}})}{\text{transistors}_{\text{LFSR}}/\text{gate\_delays}_{\text{LFSR\_encode}}}$$

Note that the parameters $\text{transistors}_{\text{pPEDDC}}$, $\text{transistors}_{\text{PEDDC}}$, and $\text{transistors}_{\text{LFSR}}$ are obtained from Figure 15a. Since $\text{transistors}_{\text{PEDDC}}$ is the number of transistors in a $2n - 1$ stage PEDDC and since only n stages are required for encoding/decoding—this number is divided by 2 in the second equation. The parameters $\text{gate\_delays}_{\text{pPEDDC\_encode}}$, $\text{gate\_delays}_{\text{PEDDC\_encode}}$, and $\text{gate\_delays}_{\text{LFSR\_encode}}$ are obtained from the Figure 15b curves which correspond to the encoding process. Since there are r and $2n - 1$ ($=2^{m+1} - 2$) times more bit calculations per clock cycle in the correcting process of the pPEDDC and PEDDC, respectively, compared to the LFSR, the power dissipation with respect to the LFSR for correcting is calculated similarly:

$$\text{Power\_Dissipation}_{\text{pPEDDC:LFSR\_correct}} = \frac{\text{transistors}_{\text{pPEDDC}}/(r \times \text{gate\_delays}_{\text{pPEDDC\_correct}})}{\text{transistors}_{\text{LFSR}}/\text{gate\_delays}_{\text{LFSR\_correct}}}$$

$$\text{Power\_Dissipation}_{\text{PEDDC:LFSR\_correct}} = \frac{\text{transistors}_{\text{PEDDC}}/(2n - 1 \times \text{gate\_delays}_{\text{PEDDC\_correct}})}{\text{transistors}_{\text{LFSR}}/\text{gate\_delays}_{\text{LFSR\_correct}}}$$

The parameters $\text{gate\_delays}_{\text{pPEDDC\_correct}}$, $\text{gate\_delays}_{\text{PEDDC\_correct}}$, and $\text{gate\_delays}_{\text{LFSR\_correct}}$ are obtained from the Figure 15b curves which correspond to the correcting process. Since the energy usage per encoding/decoding and correcting is proportional to the number of switching transistors per clock cycle multiplied by the number of clock cycles and since the LFSR requires r more clock cycles than the pPEDDC for encoding/decoding/correcting and n and $2n - 1$ more clock cycles than the PEDDC for encoding/decoding and correcting, respectively, the energy usage with respect to the LFSR is calculated as:

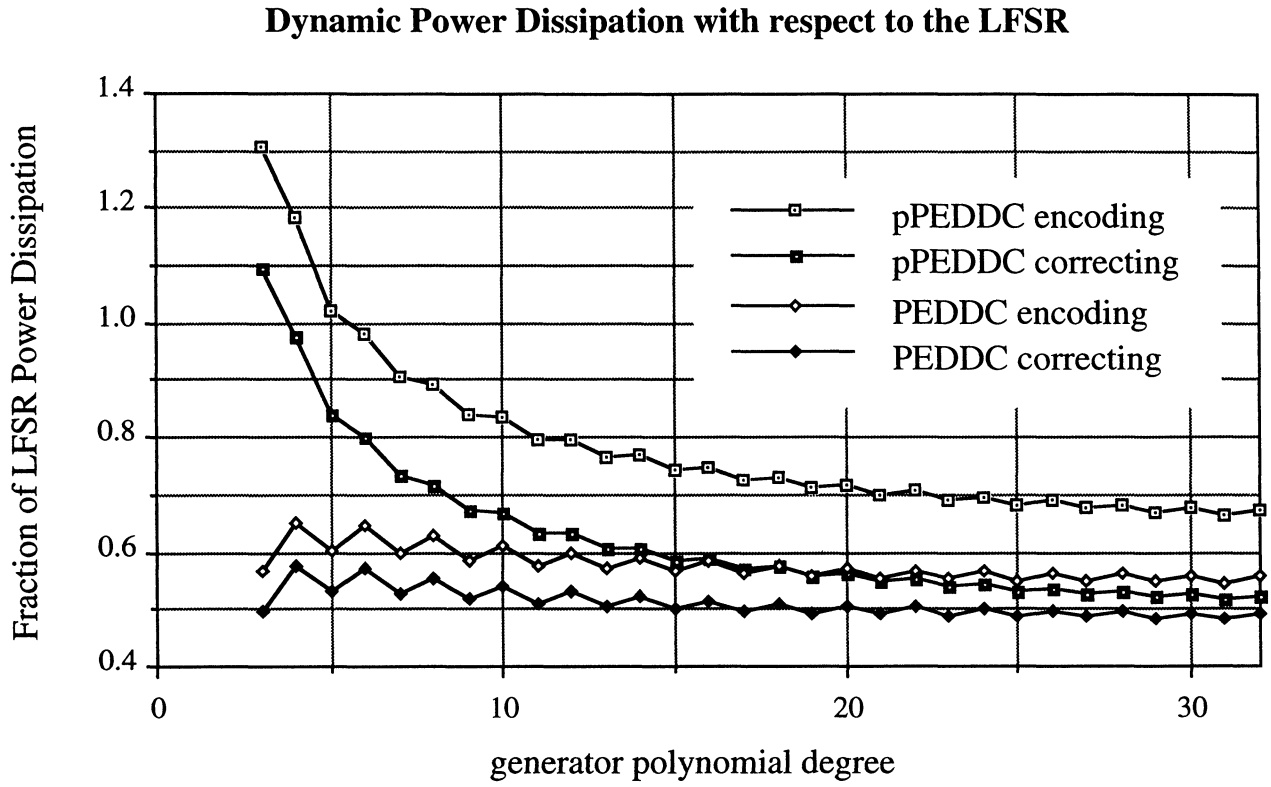
$$\text{Energy}_{\text{pPEDDC:LFSR\_encode}} = \text{Energy}_{\text{pPEDDC:LFSR\_correct}} = \frac{\text{transistors}_{\text{pPEDDC}}}{\text{transistors}_{\text{LFSR}} \times r}$$

$$\text{Energy}_{\text{PEDDC:LSFR\_correct}} = \frac{\text{transistors}_{\text{PEDDC}}/2}{\text{transistors}_{\text{LSFR}} \times n}$$

$$\text{Energy}_{\text{PEDDC:LSFR\_correct}} = \frac{\text{transistors}_{PEDDC}}{\text{transistors}_{LFSR} \times 2n - 1}$$

From the curves it is evident that encoding/decoding dissipates more power than correcting through the energy required to perform those tasks on the same number of bits is equivalent. Most importantly the curves show the advantage of using the pPEDDC/PEDDC over the LFSR; that is, the power dissipation and energy usage of the PEDDC is about one-half of the power dissipation and one-third of the energy usage in the LFSR. As the PEDDC is a special case of the pPEDDC with $r = n$ for encoding/decoding and $r = 2n$ for correcting, one would expect that for the large r the PEDDC

a.)



b.)

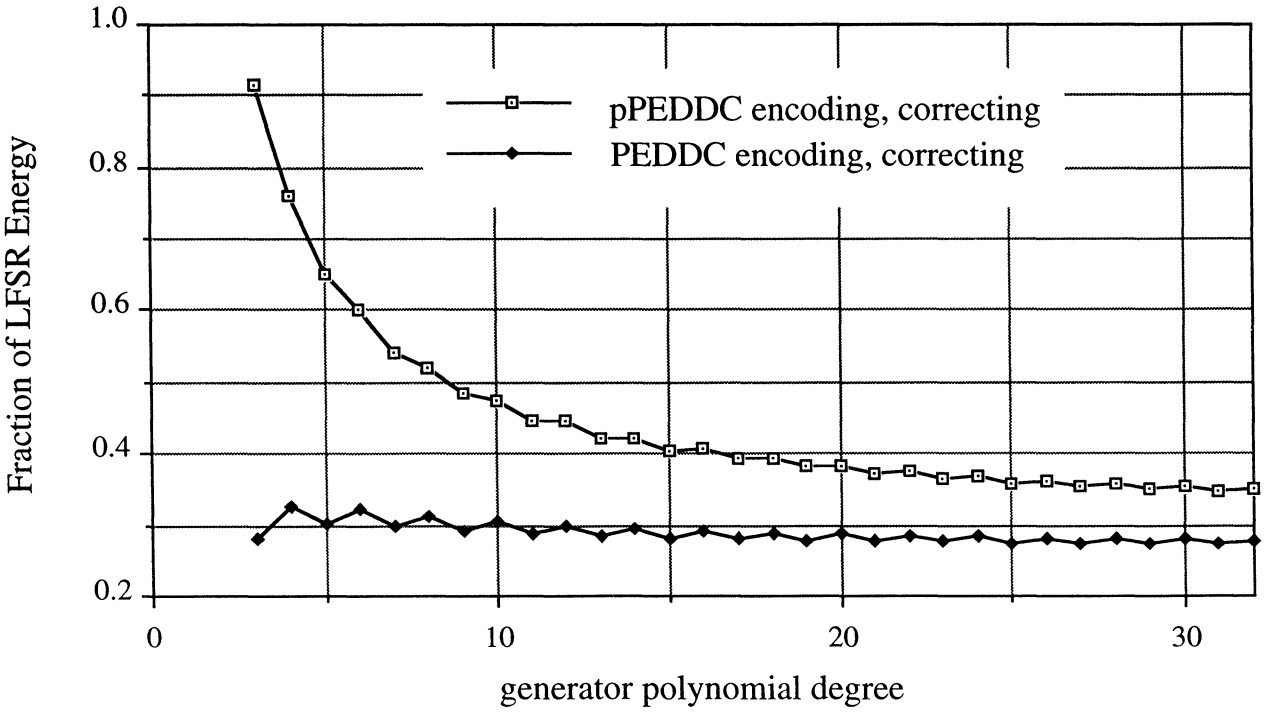


FIGURE 16a,b Ratio of power dissipation and energy usage between the pPEDDC/PEDDC and the LFSR.

and pPEDDC should experience the same level of power dissipation and energy usage. This is evident from Figure 16 where the pPEDDC curves seem to converge to the PEDDC curves with increasing r ($=m$). These curves in conjunction with Figure 15 show that an increase in circuit complexity due to partitioning (i.e. increasing r) is not counterbalanced by a decrease in clock frequency and the number of clock cycles; that is, with increasing r the clock frequency decreases at a faster rate than the complexity increases.

6. CONCLUSION

The objective of this paper is to describe the development of a multi-purpose circuit—the PEDDC—which can be used to encode, decode, detect, or correct a long stream of information sent from a transmitter to a receiver on a communication link. Essentially, the PEDDC is a redesigned LFSR which is capable of loading information in parallel.

Initially, this paper shows the equivalence of the PEDDC with the classical LFSR structure in the long-hand division conceptual framework. After showing this equivalence, the PEDDC structure is implemented and shown to be operationally equivalent to the LFSR. A (7, 4) cyclic code PEDDC VLSI chip is then built for subsequent extraction of performance parameters for any PEDDC.

Next, the PEDDC is redesigned so that it can accept any desired number of bits in parallel. This PEDDC is called a partitioned PEDDC (pPEDDC). The tremendous size advantage and nearly identical operational speed of pPEDDC over the PEDDC makes it a better choice as a substitute for the LFSR. In addition, the pPEDDC approaches the efficiency of the PEDDC for larger generators in terms of power dissipation and energy usage; that is, it dissipates about one half less power and uses one third less energy than the LFSR. By the nature in which data is processed in the pPEDDC—symbol by symbol as opposed to bit by bit—the pPEDDC seems to lead the way to error control of symbolic data on the transport layer in LANs.

The report concludes with the development of figure of merit curves based on data obtained from the VLSI chip. These curves are used to compare the sizes, encoding, decoding, detecting, and correcting rates of PEDDCs, pPEDDCs, and LFSRs for different cyclic codes. These curves show a definite speed, power, and energy advantage of the pPEDDC over the LFSR with a small sacrifice in size.

In general, a pPEDDC and PEDDC can correct 53% and 77% faster than a LFSR. Encoding takes roughly the same amount of time for both the partitioned and regular PEDDC and is 100% faster than the LFSR.

References

[1] Joseph L. Hammond and Peter J.P. O'Reilly, *Performance and Analysis of Computer Networks*, Addison-Wesley, 1986.
[2] Guido Albertengo and Riccardo Sisto, "Parallel CRC Generation," *IEEE Micro*, pp. 63–71, October 1990.
[3] Shu Lin, *An Introduction to Error-Correcting Codes*, Prentice-Hall, 1973.
[4] T.R.N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice-Hall, 1989.
[5] Scott A. Vanstone and Paul C. van Oorschot, *An Introduction to Error Correcting Codes with Applications*, Kluwer Academic Publishers, 1989.
[6] W. Wesley Peterson and E.J. Weldon, *Error-Correcting Codes*, Second Edition, M.I.T. Press, 1972.
[7] A. Sobski and A. Albicki, *Parallel Encoder, Decoder, Detector, Corrector for Cyclic Redundancy Checking*, Technical Report EL-92-01, Department of Electrical Engineering, University of Rochester, July 1992.

Biographies

ALEXANDER ALBICKI received the M.S. degree in electrical engineering from the Technical University of Warsaw, Poland in 1965, and the Ph.D. degree from the Institute of Telecommunications, Warsaw, in 1973. From 1964 to 1980 he was with the Transmission Group of the Technical University of Warsaw where he supervised the development of instrumentation and control equipment for data transmission systems and taught courses in logic design and switching theory. Since 1981, he has been with the University of Rochester where currently he is an Associate Professor in the Department of Electrical Engineering. He is engaged in research and teaching in the areas of VLSI design, metastable operation of flip-flops, design for testability, and computer network protocols. Dr. Albicki is a senior member of the IEEE Computer Society.

ANDRZEJ SOBSKI received a B.S. and M.S. degree in electrical engineering in 1990 and 1992 at the University of Rochester. His primary research interests are in the fields of VLSI design, computer communications, and coding.