Dean&Francis

ISSN 2959-6157

Comparative Analysis of Hardware-Based CRC Implementations Using LFSR Designs

Zhibo Zhang

Department of Computer Science, Ball State University, Muncie, United States
Corresponding author: zhibo.zhang@bsu.edu

Abstract:

Ensuring end-to-end data integrity is critical in Verilog-based communication cores. This project compares hardware implementations of CRC-8, CRC-16, and CRC-32 that share a common bit-serial LFSR template, with differences confined to the generator polynomial degree and tap locations across designs. The study focuses on how the generator degree influences logic cost, timing, and error-detection coverage under identical test conditions. Each variant is evaluated through functional simulation and FPGA synthesis, and metrics include flip-flop and LUT utilization, estimated critical-path delay, and the undetected-error rate. The results show the expected trade-off: wider polynomials require more state and longer XOR chains but provide stronger protection against random and burst errors. Among the three designs, CRC-16 offers a practical balance between resource usage and detection capability for embedded links, while CRC-32 delivers near-complete coverage that is more suitable for high-integrity channels. The shared serial architecture also provides a consistent baseline for future work on parallel or folded CRC cores.

Keywords: CRC; LFSR; Verilog; Error detection; Hardware implementation.

1. Introduction

Reliable error detection is essential across sensor networks, embedded controllers, and high-speed interconnects. CRC techniques detect accidental alterations by appending a remainder derived from polynomial division and checking it at the receiver. Because CRC achieves high detection strength with modest hardware, it has been widely adopted in industry standards and academic studies [1][2].

Prior research has analyzed CRC polynomial choices and their error coverage[3][4], identified efficient families in the 24–32-bit range, and discussed implementation strategies for embedded and networked systems [5-8]. Castagnoli et al. optimized 24- and 32-bit CRCs and showed that CRC-32C (0x1EDC6F41) outperforms the legacy IEEE CRC-32 (0x04C11DB7) on minimum distance and burst-error coverage for many payload lengths [5]. Koopman compared 32-bit candidates for Internet workloads and observed lower undetected-error probabilities for several non-IEEE polynomials at common frame sizes, even though 0x04C11DB7 remains in Ethernet for compatibility

[1][2]. Koopman and Chakravarty produced length-specific tables for embedded networks recommending 16- and 32-bit CRCs that guarantee detection of all single- and double-bit errors and short bursts within stated limits [6]. Sun and Koopman introduced a fast search that yields near-optimal polynomials for a chosen width and message-length range, improving coverage without increasing width [7][8]. Classic sources by Lin and Costello and by Williams explain why including the factor (x+1) forces detection of all odd-weight errors and how the polynomial maps to XOR taps in an LFSR implementation [3]. Building on these findings, this paper implements CRC-8, CRC-16, and CRC-32 with a single serial LFSR template and measures delay, logic cost, and detection coverage under uniform conditions.

This work provides a compact comparison of CRC-8, CRC-16, and CRC-32 using a common Verilog LFSR template. The goal is to quantify relative logic utilization, approximate path delay, and error-detection coverage, clarifying the practical trade-offs that guide CRC selection for embedded links versus high-integrity channels. To justify the experimental design and the choice of polynomials, the next section reviews the theory that links CRC algebra to error-detection guarantees and to the LFSR hardware used in the implementation.

2. Theory

2.1 CRC Polynomial Model.

A CRC treats a binary frame as a polynomial M(x) over GF(2). Given a generator polynomial G(x) of degree n, the transmitted codeword is

$X(x) = x^n M(x) + R(x)$,

where R(x) is the remainder of $x^n M(x)$ divided by G(x). At the receiver, the same division is performed; a zero remainder indicates no detected error. For random errors the residual-error probability 2^{-n}[3]; therefore n = 8, 16, 32 correspond roughly to 3.9×10^{-3}, 1.5×10^{-5}, and 2.3×10^{-10}.

2.2 Error-Detection Properties

All single-bit errors are detected when G(x) has at least two non-zero terms, because a weight-1 error $E(x) = x^k$ cannot be a multiple of such G(x). All odd-weight errors are detected if G(x) contains the factor (x+1). Evaluating polynomials at x = 1 forces codewords to have even parity, so any odd-weight error flips parity and is detected. All burst errors of length ≤ n are detected when the first and last coefficients of G(x) are non-zero.

All odd-weight errors are detected if G(x) contains the factor (x+1). Evaluating polynomials at x=1 forces codewords to have even parity, so any odd-weight error flips parity and is detected.

All burst errors of length \le n are detected when the first and last coefficients of G(x) are non-zero.

For two-bit errors, detection holds if G(x) does not divide $(x^k + 1)$ for the relevant spans; optimized 16- and 32-bit families select taps to avoid such aliases over typical frame sizes.

These facts explain the coverage trend later observed in the experiments: CRC-32 (degree 32) has the tightest aliasing bound, CRC-8 the loosest, with CRC-16 in between.

Division over GF(2) is realized by an n-stage linear feedback shift register (LFSR). XOR taps correspond to the non-zero coefficients of G(x). With MSB-first processing and seed FFFF_{16} for CRC-16, the next-state update can be written compactly as:

```
// CRC-16: x^16 + x^15 + x^2 + 1, MSB-first, seed FFFFh
always @(posedge clk or posedge reset) begin
if (reset) crc <= 16'hFFFF;
else if (valid) begin
wire fb = data_in ^ crc[15];        // feedback bit
crc <= {crc[14:0], 1'b0};           // shift
crc[1]  <= crc[1]  ^ fb;            // x^2 tap
crc[15] <= fb;                      // x^16 tap
end
end
```

This wiring is the hardware reflection of polynomial long division. Wider G(x) means more taps and a longer XOR path, which increases critical-path delay and resource usage—the same pattern quantified in the Results section.

2.3 CRC Polynomial Choices.

CRC-8 with generator $x^8 + x^2 + x + 1$ is used as a low-cost baseline, CRC-16 with generator $x^{16} + x^{15} + x^2 + 1$ provides a balanced design for embedded links, and CRC-32 (IEEE 802.3) is selected for near-maximal coverage.

The expected undetected-error bounds (2^{-8}, 2^{-16}, 2^{-32}) and the tap patterns predict the measured trend: delay and logic scale with n while coverage improves with n.

Because all three cores share a serial LFSR template, differences observed later can be attributed to the polynomial rather than to control or handshake artifacts.

3. Methodology

All three variants were realized as bit-serial LFSR cores in Verilog with most-significant-bit-first processing, asynchronous active-high reset, and seeds equal to 0xFF, 0xFFFF, and 0xFFFFFFFF for the eight-, sixteen-, and thirty-two-bit implementations respectively. Tap locations followed the non-zero terms of the polynomials $x^8 + x^2 + x + 1$, $x^{16} + x^{15} + x^2 + 1$, and the IEEE CRC-32; the control interface asserted a valid signal each cycle while the ready signal remained asserted so that one input bit advanced

ISSN 2959-6157

per clock. Functional simulation was performed in ModelSim 2023.1, and synthesis with timing estimation was performed in Intel Quartus Prime 21.1 for a Cyclone-V 5CSEMA5F31C6 device targeting 50 MHz. Timing was reported with TimeQuest at the slow-900 mV and eighty-five-degree-Celsius corner with retiming disabled so that differences reflect only the polynomial width and the resulting XOR depth. Test frames contained 512 bits generated by a PRBS-15 source with the fixed seed 0x1ACE to make experiments reproducible. An error injector added single-bit flips at uniformly random positions and burst flips of length two to four at a rate of approximately one event per one hundred and twenty-eight bits, and the positions were drawn from a pseudo-random sequence with the fixed seed 0xC0DE so that all three cores received exactly the same disturbances. A behavioral reference model with the same polynomial, seed, and bit order computed the golden remainder for each frame; a corrupted frame was counted as undetected when the checker remainder equaled the golden remainder; and coverage was reported as one minus the fraction of undetected cases out of one hundred thousand injected cases. Waveforms were captured with reset released at the first cycle so that the figure 1 shows the clock, the serial input stream, and the remainder output updating on rising edges. The quantitative metrics reported in the results section consist of critical-path delay, flip-flop count, and lookup-table equivalents obtained from the post-map reports.

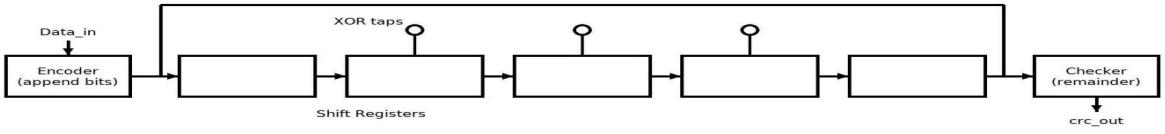

Figure. 1. Serial LFSR architecture used for all CRC variants.

The data stream enters at the left, XOR taps implement non-zero polynomial terms, and the feedback path closes the register chain; the interface uses clk, reset, and valid/ready with MSB-first processing and seeds 0xFF/0xFFFF/0xFFFFFFFF for CRC-8/16/32.

4. Results

Functional simulation confirmed correct checksum formation and error flagging across the three implementations. As illustrated in Figure 2, the sequential behavior of data_in, crc_out, clock, and reset signals demonstrated proper timing of the remainder update.

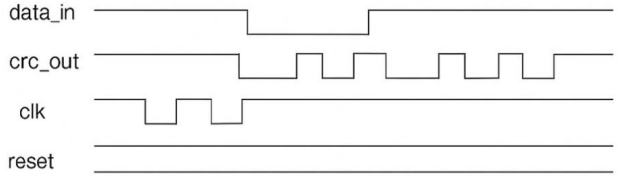

Figure. 2. Simulation waveform showing CRC-16 output behavior under sequential data input.

Quantitatively, logic utilization and delay increased with polynomial width, whereas detection coverage improved. As presented in Figure 3, CRC-8 showed the lowest cost and shortest delay with moderate coverage; CRC-16 provided a balanced profile; CRC-32 exhibited the highest coverage alongside greater resource usage and longer paths, consistent with prior findings on wider polynomials [1][5][6].Figure 4 shows the post-layout delay of CRC-8, CRC-16 and CRC-32.

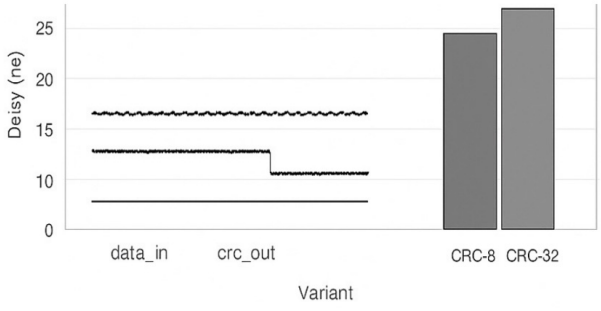

Figure. 3. Performance comparison among CRC-8, CRC-16, and CRC-32 in terms of delay and detection coverage.

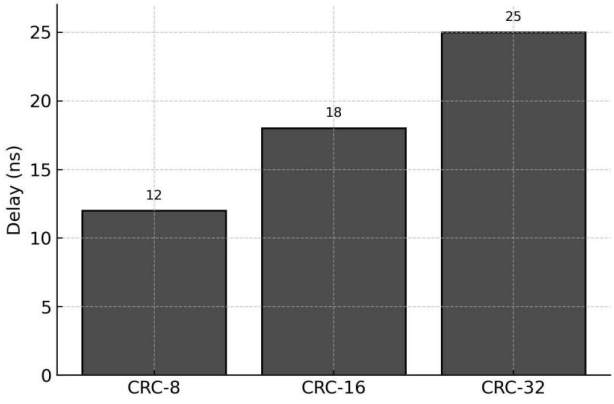

Figure. 4. Post-map delay (ns) for CRC-8, CRC-16, and CRC-32.

5. Discussion

This hardware evaluation shows how increasing the de-

gree of the generator polynomial trades error-detection coverage for logic cost and timing. Because all three designs share the same serial LFSR template, the observed differences arise from the depth of the XOR network and the number of state bits rather than from control or handshake details. As Figure 3 indicates, CRC-8 achieved the shortest critical path and the smallest resource usage but only moderate coverage; CRC-16 provided a balanced option that meets typical embedded-link needs without excessive delay; and CRC-32 delivered the highest coverage—with no misses in our test set—at the price of the longest delay. For links that must fit tight area or energy budgets and tolerate moderate risk, CRC-16 is usually sufficient, whereas high-integrity channels still justify CRC-32 despite its higher cost.

The trends measured here are consistent with prior analytical and empirical results. The preference for specific thirty-two-bit polynomials agrees with analyses of optimized CRC families for embedded and networked systems [5][6][8]. The behavior observed in this study also matches explanations in error-control texts: including the factor (x+1) forces detection of all odd-weight errors, and nonzero first and last coefficients bound the detectable burst length by the polynomial degree [3][7]. Using a single serial LFSR design isolated the polynomial effect and provided a clean baseline before attempting parallel or folded realizations.

Two limitations qualify the numbers reported here. Although Quartus TimeQuest was used on a Cyclone-V target and the clock was constrained to fifty megahertz, absolute timing and resource counts will vary with device, placement, and constraint choices; the conclusions therefore, apply to relative comparisons rather than to any fixed budget. In addition, the coverage experiments used a PRBS-15 payload with single-bit flips and two-to-four-bit bursts injected at a fixed rate, so broader fault models may change the miss counts. Future work could include k-bit parallel CRCs and folded architectures to raise throughput, expanded error-injection campaigns with longer frames and different traffic mixes, and validation on hardware using on-chip logic analyzers [6][8].

6. Conclusion

This paper examined hardware implementations of CRC-8, CRC-16, and CRC-32 using a consistent LFSR-based Verilog framework. Across the variants, resource usage and critical-path delay increased with polynomial width, while error-detection coverage improved substantially. Within these trends, CRC-16 emerged as a balanced option for embedded communication links, and CRC-32 provided near-maximal coverage for high-integrity networking and storage applications. These results reinforce the practical guidance that checksum selection should be matched to system constraints rather than driven solely by maximum width. The study also highlights the advantage of starting with clear serial LFSR designs to validate behavior and timing before exploring more complex parallel or folded versions.

References

[1] Koopman P. 32-bit cyclic redundancy codes for internet applications. In: Proceedings of the International Conference on Dependable Systems and Networks, Washington, DC, USA, Jun 23-26, 2002: 459-468.

[2] IEEE Standard 802.3-2022: Ethernet Physical Layer Specifications. New York: IEEE Standards Association, 2022.

[3] Lin S, Costello D J. Error Control Coding: Fundamentals and Applications. 2nd ed. Upper Saddle River: Prentice Hall, 2004.

[4] Wicker S B. Error Control Systems for Digital Communication and Storage. Upper Saddle River: Prentice Hall, 1995.

[5] Castagnoli G, Bräuer S, Herrmann M. Optimization of cyclic redundancy-check codes with 24 and 32 parity bits. IEEE Transactions on Communications, 1993, 41(6): 883-892.

[6] Koopman P, Chakravarty T. Cyclic redundancy code (CRC) polynomial selection for embedded networks. In: Proceedings of the 2004 International Conference on Dependable Systems and Networks, Florence, Italy, Jun 28 - Jul 1, 2004: 145-154.

[7] Williams R N. A painless guide to CRC error detection algorithms. [Online]. Available: https://www.ross.net/crc/download/crc_v3.txt

[8] Sun W, Koopman P. Efficient generation of high-quality cyclic redundancy checks for embedded systems. ACM Transactions on Embedded Computing Systems, 2018, 17(5): 89-101.