Beijing Boyuan Technology Co. Ltd · BOYUAN TECHNOLOGY

Semiconductor IP Cores · High-Performance Storage & Error Correction Code (ECC)

BCH Codec IP Core
x32bit · GF(2^{13}) Technical Whitepaper

A Configurable ECC Solution for Flash Memory and Compute-in-Memory

Product BCH Codec x32bit GF(2^{13})

Version V1.0

Date October 2026

Audience ASIC / FPGA integration, system architecture and storage reliability evaluation

Abstract This whitepaper presents BOYUAN's next-generation BCH codec IP core. Operating over GF(2^{13}) with a 32-bit input/output data bus, the core features configurable error-correction capability and shortened-code lengths. The decoder adopts a three-stage EDC / KES / CSM pipeline, with configurable Key Equation Solver (KES) parallelism to flexibly trade area against throughput. The document covers BCH fundamentals, shortened-code zero-padding and word-alignment rules, codec architecture, AXI-Stream interface, verification methodology, SMIC 28nm synthesis results, and applications in compute-in-memory (CIM) chips.

© 2026 Beijing Boyuan Technology Co. Ltd | www.bjbytech.com | sale@bjbytech.com | Phone / WeChat: +86 136 0100 5061

Contents

1 New Challenges in Storage Reliability
2 BCH Fundamentals and GF(2^{13})
3 Shortened Codes: Zero-Padding and Alignment
4 Encoder Architecture and AXI-S Input Alignment
5 Decoder: EDC / KES / CSM Pipeline
6 Configurable KES Parallelism
7 AXI-Stream Interface and Result Reporting
8 Test and Verification Methodology
9 Synthesis Results and Frequency Discipline
10 Compute-in-Memory (CIM) Applications
11 Conclusion and Contact

1 New Challenges in Storage Reliability

1.1 Error Sources in Flash Memory

As NAND flash scales down, moves to multi-level cells (SLC → MLC → TLC / QLC) and increases 3D stacking layers, the raw bit error rate (RBER) keeps rising. Major error sources include cell-to-cell interference, oxide wear from program/erase cycling, read disturb, charge leakage during data retention, and process variation. Error correction code (ECC) is the core block that guarantees data integrity in a flash controller; its correction strength directly determines endurance and data-retention time.

1.2 New Demands from Compute-in-Memory

In compute-in-memory (CIM) architectures, memory arrays directly perform multiply-accumulate operations, with weights and activations resident on chip. Process variation, supply noise and device aging can flip weight bits, and erroneous weights propagate directly into the compute array, causing systematic inference errors. CIM chips therefore require **low-latency, online** error correction that can be configured to the array bandwidth and area budget.

> **Positioning of this IP**
> The BOYUAN BCH codec IP serves both conventional flash/SSD ECC and emerging CIM array protection, with a 32-bit bus, configurable correction strength and an area-efficient pipelined architecture.

2 BCH Fundamentals and GF(2^{13})

2.1 The Galois Field GF(2^{13})

BCH codes are built over the finite field GF(2^m). This IP uses m = 13, i.e. GF(2^{13}), with 8192 field elements and a maximum primitive BCH codeword length of $n = 2^{13} - 1 =$ **8191 bits**. Choosing m = 13 lets a single codeword cover a 512-byte (4096-bit) data block while leaving enough parity margin — a common sweet spot for flash and on-chip storage.

2.2 Primitive BCH Codes and Shortened Codes

For a primitive BCH code correcting t random errors, the generator polynomial g(x) is the least common multiple of the minimal polynomials of α, α^2, ..., α^{2t}, giving a designed minimum distance of 2t + 1. Encoding multiplies the message polynomial m(x) by x^{n-k} and takes the remainder modulo g(x) to form the parity bits. Decoding computes syndromes, solves the key equation for the error-locator polynomial, then searches for error locations and corrects them.

Practical data blocks are often shorter than the primitive codeword length. A **shortened code** fixes several high-order message bits to zero (not transmitted, not decoded), preserving correction strength t and parity length while shortening the codeword to the required length. Both the message length and codeword length of this IP are configurable; typical configurations are:

Parameter	16bit correction	36bit correction
Correction capability t	16 bits	36 bits
Message length k	4096 bits (512 bytes)	4136 bits
Parity length	208 bits	468 bits
Codeword length n	4304 bits	4604 bits
Galois field	GF(2^{13})	GF(2^{13})

Parity and codeword lengths follow from the field order, correction strength and shortening amount; other correction levels and message lengths are available on request.

3 Shortened Codes: Zero-Padding and Alignment

With data entering and leaving in 32-bit words (dim = 32), neither the message length nor the parity length is necessarily a multiple of the word length. Without alignment, word boundaries would drift from frame to frame. The alignment rules of this IP are as follows.

3.1 Zero-Padding at the High-Order End

Given a message length `message_length`, the number of words needed to reach a word boundary is `message_dw = ceil(message_length / dim)`. The first `mask_bit` positions of the **first row (high-order end)** of the message matrix are zeroed:

```
mask_bit = message_dw × dim − message_length
Message(1, 1 : mask_bit) = 0 // pad at high-order end
```

Padding at the high-order end — rather than the low-order end — matches the semantics of a shortened code ("high-order message bits of the primitive code are zero"). On decode, the leading zero words are simply discarded, requiring no extra data movement.

3.2 Parity Alignment and Removal of the All-Zero Word

```
bzp = dim - rem(par_bit, dim)
m1(1, 1 : bzp) = 0 // pad parity at high-order end
Code = m2(2 : end, :) // drop the all-zero first row
```

> **Design note**
> The padding produces an all-zero first word that carries no information. Sending it to the decoder would waste bandwidth, so it is removed from the codeword (`Code = m2(2:end,:)`), keeping encoder and decoder word boundaries strictly aligned.

4 Encoder Architecture and AXI-S Input Alignment

4.1 LFSR Division Encoding

The encoder implements polynomial division with a linear-feedback shift register (LFSR) based on the generator polynomial g(x). Message bits are shifted in and XOR-fed back according to g(x); after the last message bit, the register content is the parity. This structure is compact, regular and easy to parameterize for different code lengths and correction strengths.

4.2 Encoder Input Alignment Rule

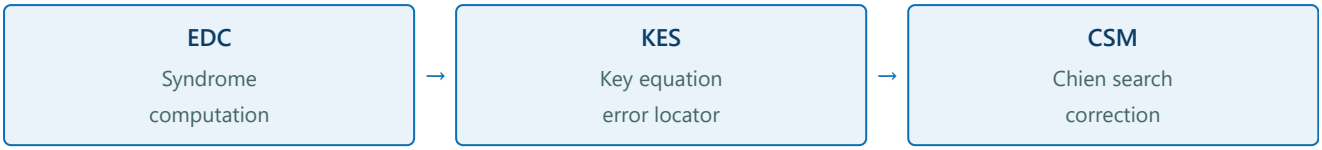
With input data width DW = 32 and total message bits MSG_BITS, the last beat is generally not a full word. Define:

```
LAST_BITS = MSG_BITS mod DW // use DW when remainder = 0
NW = ceil(MSG_BITS / DW) // total beats
LW = NW - 1 // index of last beat
BUF_BITS = DW - LAST_BITS // bits carried from previous beat
```

- **Non-last beat:** {tdata_buf, s_axis_tdata[DW−1 : DW−LAST_BITS]} — take the high LAST_BITS bits of the input word and concatenate with the buffer.
- **Last beat:** {tdata_buf, s_axis_tdata[LAST_BITS−1 : 0]} — take the low LAST_BITS bits and concatenate with the buffer to complete a word.

`s_axis_tlast` marks the last beat and `s_axis_tuser` marks the frame start. Both sides stay word-aligned for any MSG_BITS without extra length negotiation.

5 Decoder: EDC / KES / CSM Pipeline



5.1 EDC: Syndrome Computation

EDC evaluates the received word r(x) at α^1 … α^{2t} to obtain 2t syndromes $S_i = r(\alpha^i)$. All-zero syndromes indicate no error; otherwise the syndromes are passed to KES. EDC processes one 32-bit word per cycle, with latency about `ceil(N/32)` cycles (N = codeword length).

5.2 KES: Key Equation Solving

KES solves the key equation from the syndromes to obtain the error-locator polynomial σ(x). It uses an iterative structure with 2t iterations, processing U error locations per cycle (U is the unroll factor, see Section 6), giving latency about `ceil(2t/U)` cycles. It is the most compute-intensive and area-dominant block, and the core of the IP's configurability.

5.3 CSM: Chien Search and Correction

CSM tests roots of σ(x) position by position to locate errors and flips the corresponding bits. It uses BOYUAN's proprietary technology for area efficiency, processing one 32-bit word per cycle with latency about `ceil(N/32)` cycles. A key implementation detail is the exact derivation of the search iteration bound from codeword length, correction strength and field parameters — an off-by-one error here causes simulation/hardware inconsistency.

5.4 Throughput and Latency Definition

> **Latency definition**
>
> Decoding latency is defined as **the number of cycles from the last input beat with TVALID & TREADY asserted to the first output beat**. In steady state each pipeline stage works on one frame, so throughput is set by the slowest stage. EDC/CSM take about ceil(N/32) cycles and KES about ceil(2t/U); configuring U so the three stages have matched latency yields continuous, stall-free output.

6 Configurable KES Parallelism

The unroll factor U is the number of error locations processed per cycle in KES. A larger U reduces iterations (`ceil(2t/U)`) and KES latency at the cost of more parallel hardware and area. This knob lets one architecture serve very different applications:

Correction level	KES structure	Characteristic
16bit (t = 16)	Half-unrolled	2t far below data-beat count, moderate parallelism balances the three stages, area friendly
36bit (t = 36)	Fully parallel	2t near data-beat count, full unroll avoids KES bottleneck, throughput first

The choice of KES unrolling depends on the relationship between correction strength and the data-beat count, so that the three pipeline stages have matched processing periods:

- **16bit correction (t = 16, 2t = 32):** 2t is far smaller than the codeword data-beat count `ceil(4304/32) ≈ 135`, so the KES workload is relatively light. A **half-unrolled** structure is sufficient to match the pipeline rhythm, balancing the three stages with more economical resources and avoiding the area cost of full parallelism.
- **36bit correction (t = 36, 2t = 72):** 2t is already close to the data-beat count `ceil(4604/32) ≈ 144`, and the KES workload grows substantially. A **fully parallel** structure fully unrolls the iterations, compressing the per-frame KES period to match EDC/CSM and preventing KES from becoming a bottleneck. This is why even nearly doubling the correction strength still converges at a 1 GHz constraint — the value of "36bit fully parallel" over simply raising t.

> **Configurable items**
> Correction strength, message length, codeword length and KES unroll factor U are all configurable. Customers can pick the combination that fits array bandwidth, memory capacity, target process and area budget.

7 AXI-Stream Interface and Result Reporting

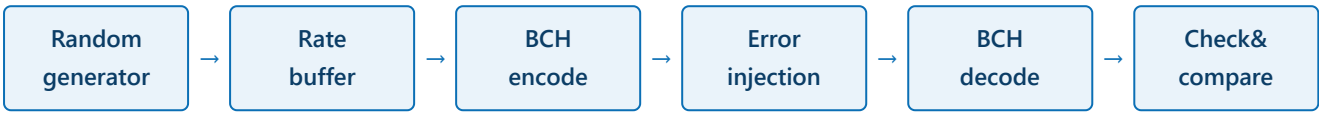
The codec top level uses a standard AMBA AXI-Stream interface with 32-bit input and output. Data transfers on the TVALID / TREADY handshake, and the receiver can apply backpressure by deasserting TREADY — convenient for direct connection to an on-chip bus or CIM macro.

Signal	Dir / Width	Description
s_axis_tdata / tvalid / tready	Input / 32,1,1	Input data bus and handshake
s_axis_tuser / tlast	Input / 1,1	Input frame start / end flags
m_axis_tdata / tvalid / tready	Output / 32,1,1	Output data bus and handshake
m_axis_tuser	Output / 2	Bit[0] output frame start; Bit[1] correction flag
m_axis_tlast	Output / 1	Output frame end flag
dec_result	Output / 8	Bit[6:0] corrected-bit count; Bit[7] decode success
dec_result_en	Output / 1	Result valid, reported in input frame order

The encoder has 12 ports and the decoder 14. `dec_result` and `dec_result_en` let the system learn the corrected-bit count and decode success frame by frame — useful for wear leveling, reliability statistics and online diagnostics.

8 Test and Verification Methodology

A complete self-checking testbench is provided, covering both RTL simulation and FPGA board-level testing. The data path is:



- The **random generator** uses four parallel LFSRs to produce 32-bit pseudo-random data.
- The **rate buffer** handles clock-domain crossing and rate matching for a stable stimulus.
- **Error injection** position, pattern and enable come from configurable registers, covering random, burst and boundary errors across correction levels.
- The **checker** compares original data with decoded output frame by frame, counting pass/fail frames and reporting corrected bits.

In FPGA testing, the testbench is integrated at the top level; VIO configures error patterns and frame enables online while decode results and frame counters are monitored in real time for long runs.

9 Synthesis Results and Frequency Discipline

9.1 Synthesis Conditions

The decoder core was synthesized with Design Compiler in the SMIC 28nm library (scc28nhkcp_vhsc35p140_rvt_tt_v0p9_25c_basic) at the TT typical corner (0.9V / 25°C) under a 1ns (1GHz) clock constraint.

9.2 Results

Item	16bit @ 4304bit	36bit @ 4604bit
Design	bch_4304_4096_dec_core	bch_4604_4136_dec_core
Ports	19,592	43,312
Nets	70,950	171,337
Cells	51,458	128,240
Combinational cells	40,361	108,060
Sequential cells	11,020	19,987
Buf / Inv	5,595	14,486
Total cell area	84,755.16	199,570.40

Area in library-standard units. Buf/Inv area is included in combinational area; with no wire-load model specified, Net Interconnect Area and Total Area are undefined.

9.3 Area Difference Analysis

The 36bit configuration uses about 2.5× the cells of the 16bit one, as expected: the number of syndromes grows with 2t (32 → 72), the fully parallel KES multiplies the parallel datapath, and the CSM search/correction path scales with t. Cell counts should not be compared across processes or constraints; this table only shows the relative cost of two correction levels under the same process and constraint, helping customers choose a level within their area budget.

> **Frequency disclaimer**
> These are pre-layout DC synthesis estimates without post-P&R interconnect delay and **do not represent a final sign-off frequency**. Convergence under a 1ns constraint means timing margin is just satisfied at synthesis stage; margin may be consumed by interconnect delay after place & route, and should be re-evaluated in the target process and layout.

10 Compute-in-Memory (CIM) Applications

In CIM chips, the IP provides online soft-error correction for weight and activation memories: the three-stage pipeline decodes each memory block with low latency, preventing erroneous weights from reaching the multiply-accumulate array. The 32-bit AXI-Stream interface integrates easily with CIM macros and on-chip buses, while correction strength, codeword length and KES parallelism can be tuned to array bandwidth, memory capacity and area budget for both reliability and energy efficiency.

Integration guidance

For area-constrained edge inference arrays, choose a lower correction level with a half-unrolled KES. For dense weight storage or high end-of-life reliability requirements, the 36bit fully parallel configuration delivers higher correction strength and throughput.

11 Conclusion and Contact

The BOYUAN BCH codec IP offers a flexible balance of reliability, area and throughput for flash/SSD and CIM applications, through a 32-bit data bus, GF(2^{13}) domain, configurable correction and codeword length, an area-efficient three-stage pipeline and configurable KES parallelism. A well-defined specification, complete verification platform and SMIC 28nm synthesis data reduce integration risk and shorten time to market.

Contact · Purchase / Evaluation / Technical Consulting

Sales / Evaluation: sale@bjbytech.com **Phone / WeChat:** +86 136 0100 5061

Technical support: tech@bjbytech.com **Phone / WeChat:** +86 136 0100 5061

Web: www.bjbytech.com

© 2026 Beijing Boyuan Technology Co. Ltd. Content is subject to change without notice. Synthesis data are estimates under specific process and constraints and do not constitute a final performance commitment.