

# Throughput-Driven Architecture for BCH Error Correction Codes

Arul K. Subbiah

Design Manager, Arasan Chip Systems Inc.

Email : [aruls@arasan.com](mailto:aruls@arasan.com)

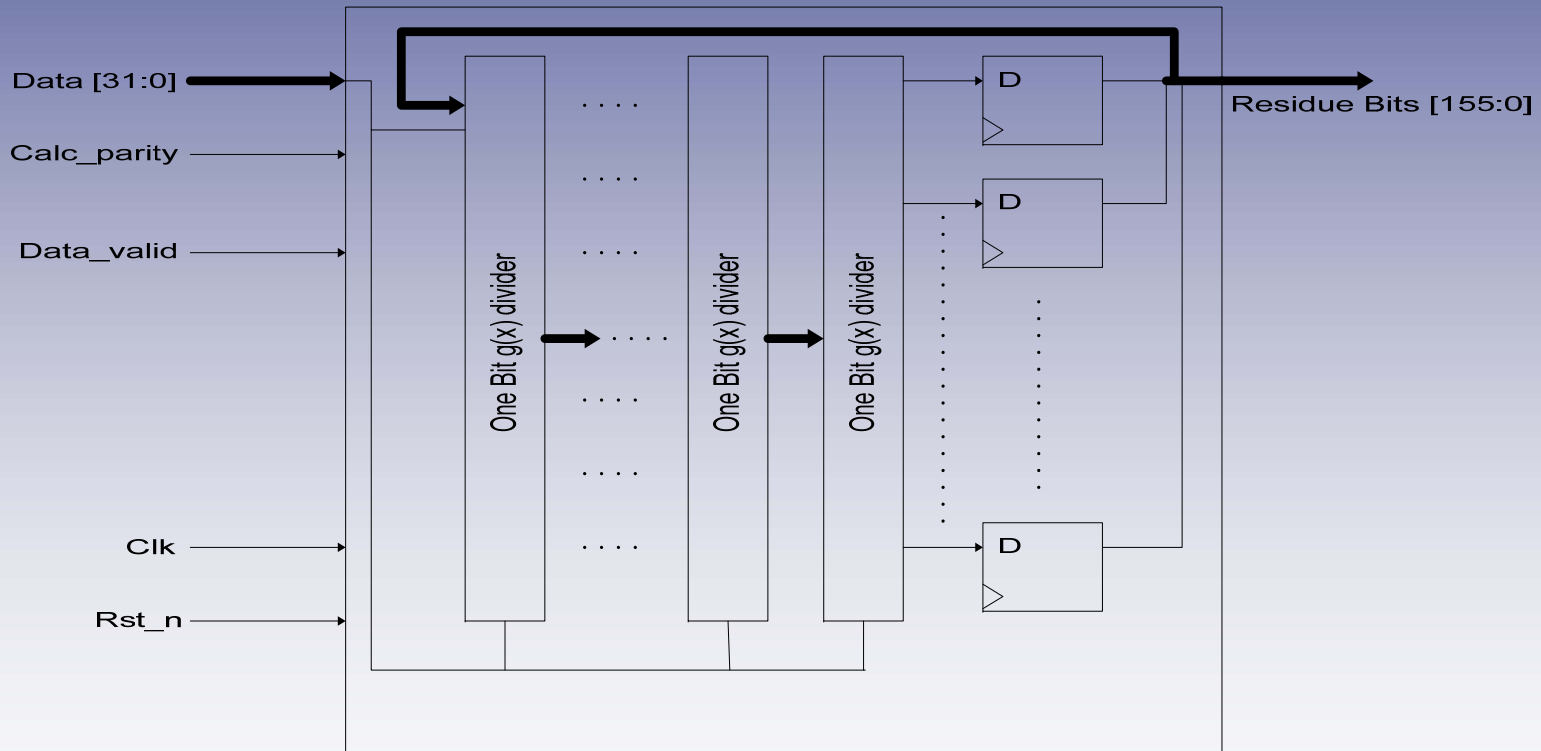
# Agenda

- Overview
- BCH Encoder architecture
- Integrating Encoder module
- BCH Decoder architecture
- Syndrome generator
- Key equation solver
- Error corrector module
- Integrating Decoder module

## Overview

- BCH (Bose – Chaudri – Hocquenghem) is an error correcting code
- Used in **Storage Media, Digital communications**
- Cyclic Code nature makes it easily realizable in hardware
- Two important **parameters** for BCH error correcting codes
  - Number of bytes (K)
  - Number of error to be corrected (T)
- Galois Field extension dictates the complexity of the design, which is given by:
  - M – Galois Field extension = Upper {  $\log_2 (N)$  }
  - N – Number of bits including the parity bits
  - Ex. Block size of 512 bytes with 12 bit error correction has
    - M = Upper {  $\log_2 (512*8+12)$  } = 13
- The number of parity check bits required for correcting T bits is given by:
  - P – Number of Parity bits = M \* T
  - T – Number of error bits to be corrected

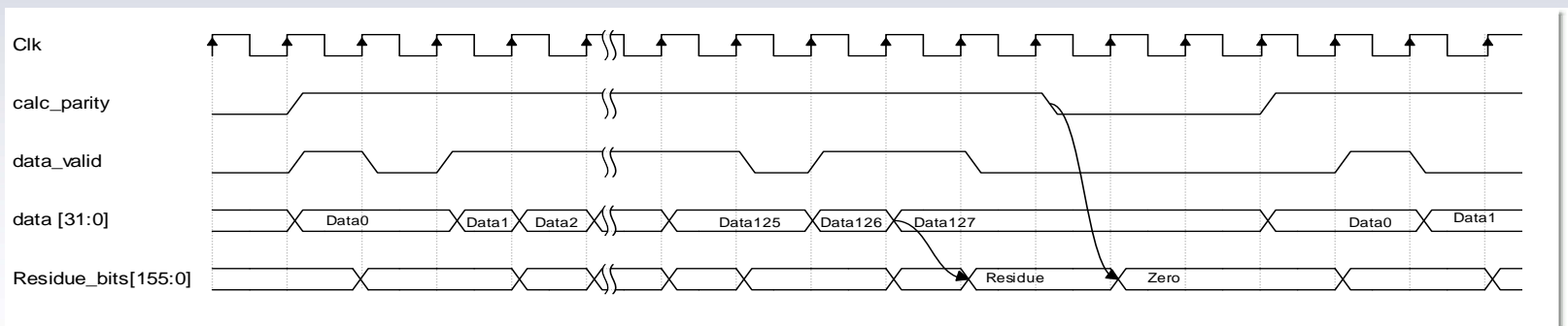
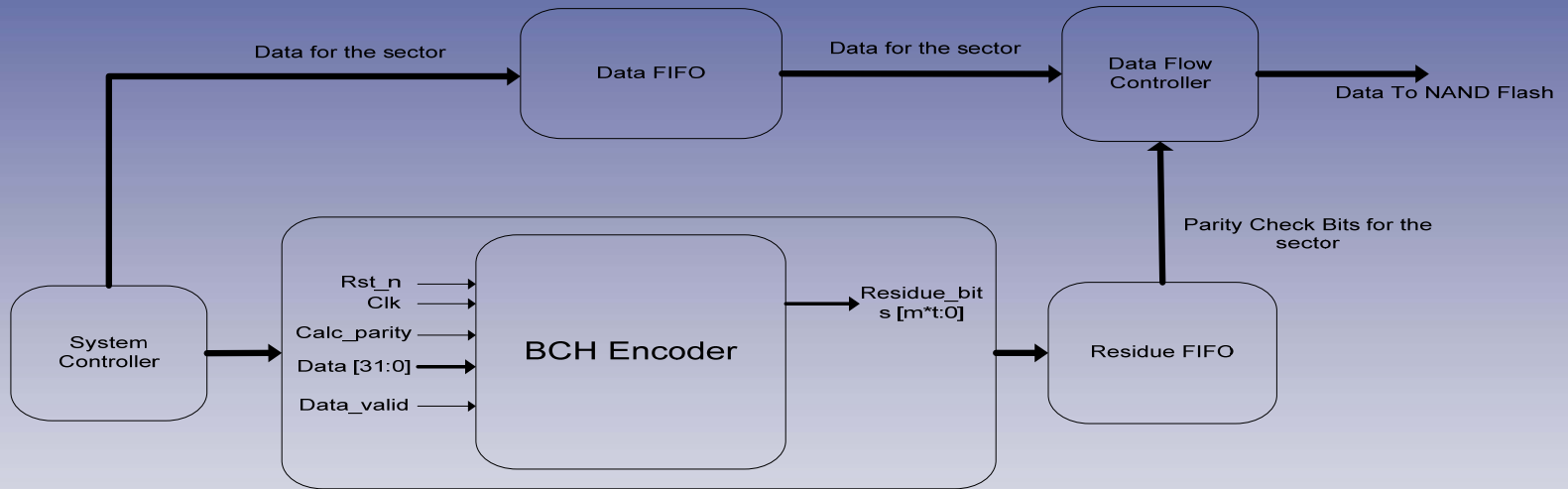
# BCH Encoder Block Diagram



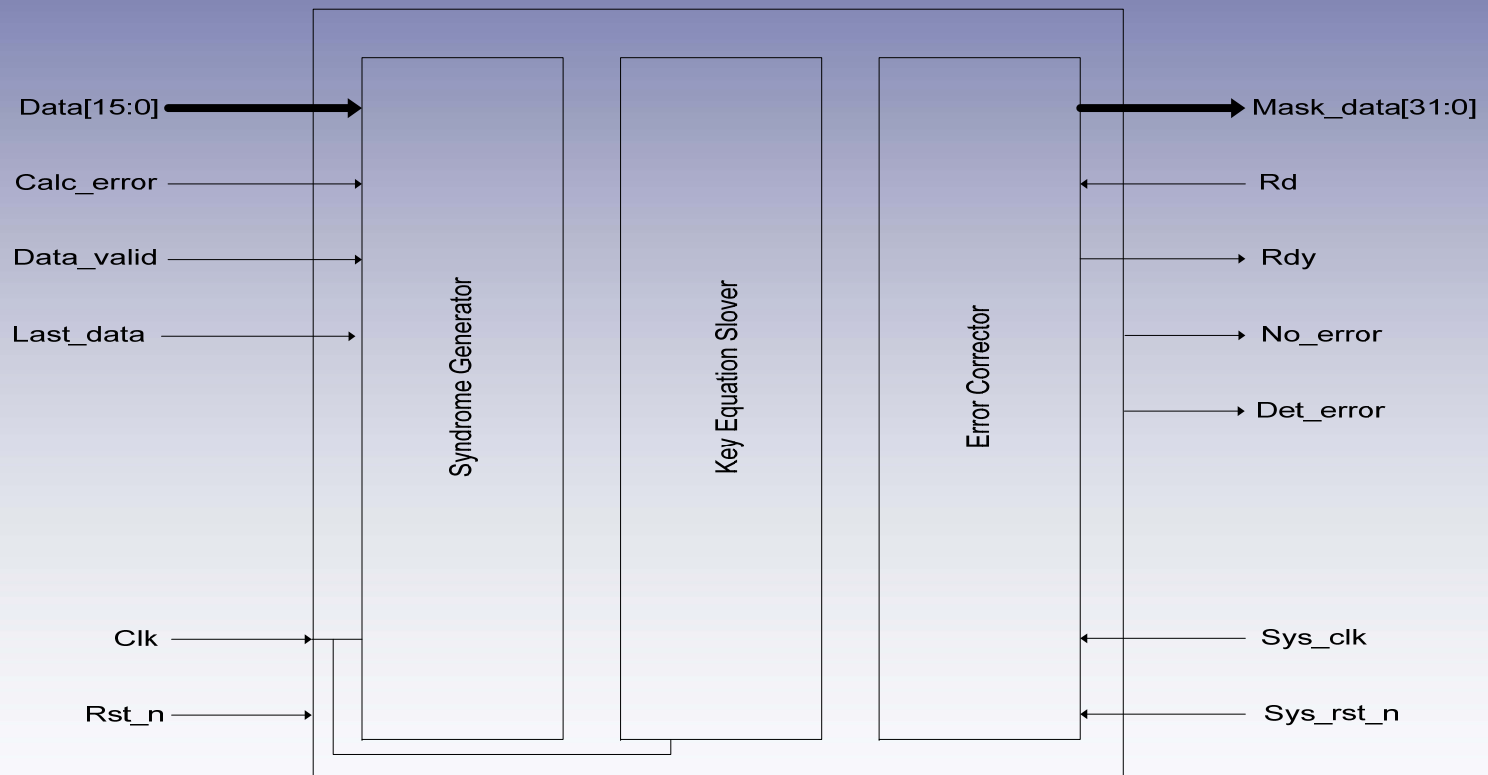
$$g(x) = \text{LCM} \{ \phi_1(x), \phi_2(x), \dots, \phi_{2t}(x) \}$$

$\phi_i(x)$  be the minimal polynomial of  $\alpha^i$

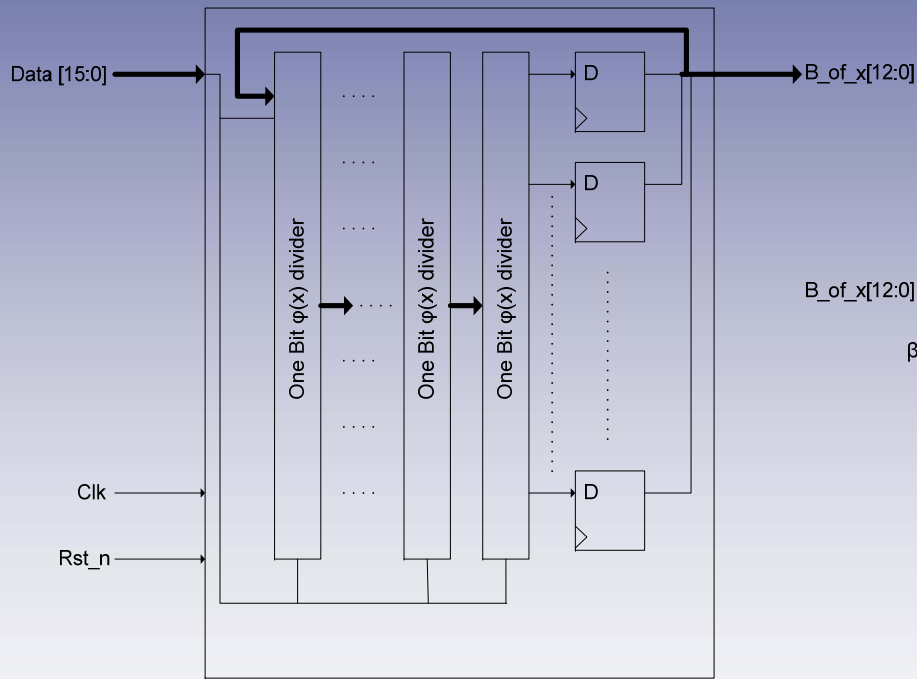
# Integrating Encoder module



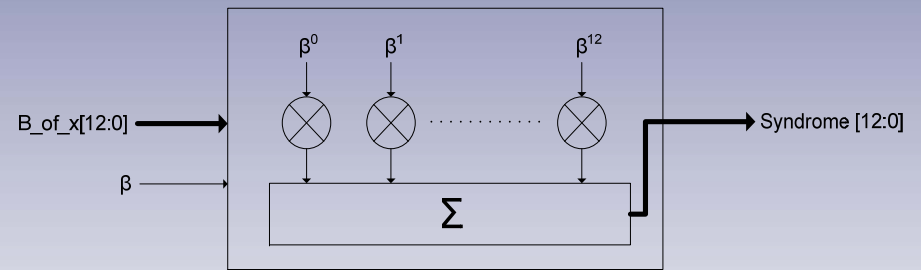
# BCH Decoder block diagram



# Syndrome Generator



(a)  $B(x)$  generator



(b) Syndrome value generator

$$S_i = \sum_{j=0}^{n-1} r_j (\alpha^{i+1}), (0 \leq i \leq 2t)$$

# Key equation solver (inversionless Berlekamp-Massey Algorithm)

The iBMA algorithm can be represented by the following steps.

Step 1:  $d_r = \sum_{i=0}^t \sigma_i^{(r)} \cdot S_{2r-i+1}$ , where  $d_r$  is the discrepancy value

Step 2:  $\sigma^{(r+1)}(x) = d_p \cdot \sigma^{(r)} + d_r \cdot \beta^{(r)}(x)$

Step 3:  $\text{bsel} = 0$  if  $d_r = 0$  OR  $r < l_r$ ;  
 $\text{bsel} = 1$  if  $d_r$  AND  $r \geq l_r$

Step 4:  $\beta^{(r+1)} = x^2 \cdot \beta^{(r)}$  if  $\text{bsel} = 0$ ;  
 $\beta^{(r+1)} = x^2 \cdot \sigma^{(r)}$  if  $\text{bsel} \neq 0$

Step 5:  $l_{r+1} = l_r$  if  $\text{bsel} = 0$ ;  
 $l_{r+1} = 2 \cdot r - l_r + 1$  if  $\text{bsel} \neq 0$

Step 6:  $d_p = d_p$  if  $\text{bsel} = 0$ ;  
 $d_p = d_r$  if  $\text{bsel} \neq 0$

Step 7:  $r = r + 1$

$$d_p = \begin{cases} 1 & \text{if } S_1 = 0 \\ S_1 & \text{if } S_1 \neq 0 \end{cases}$$

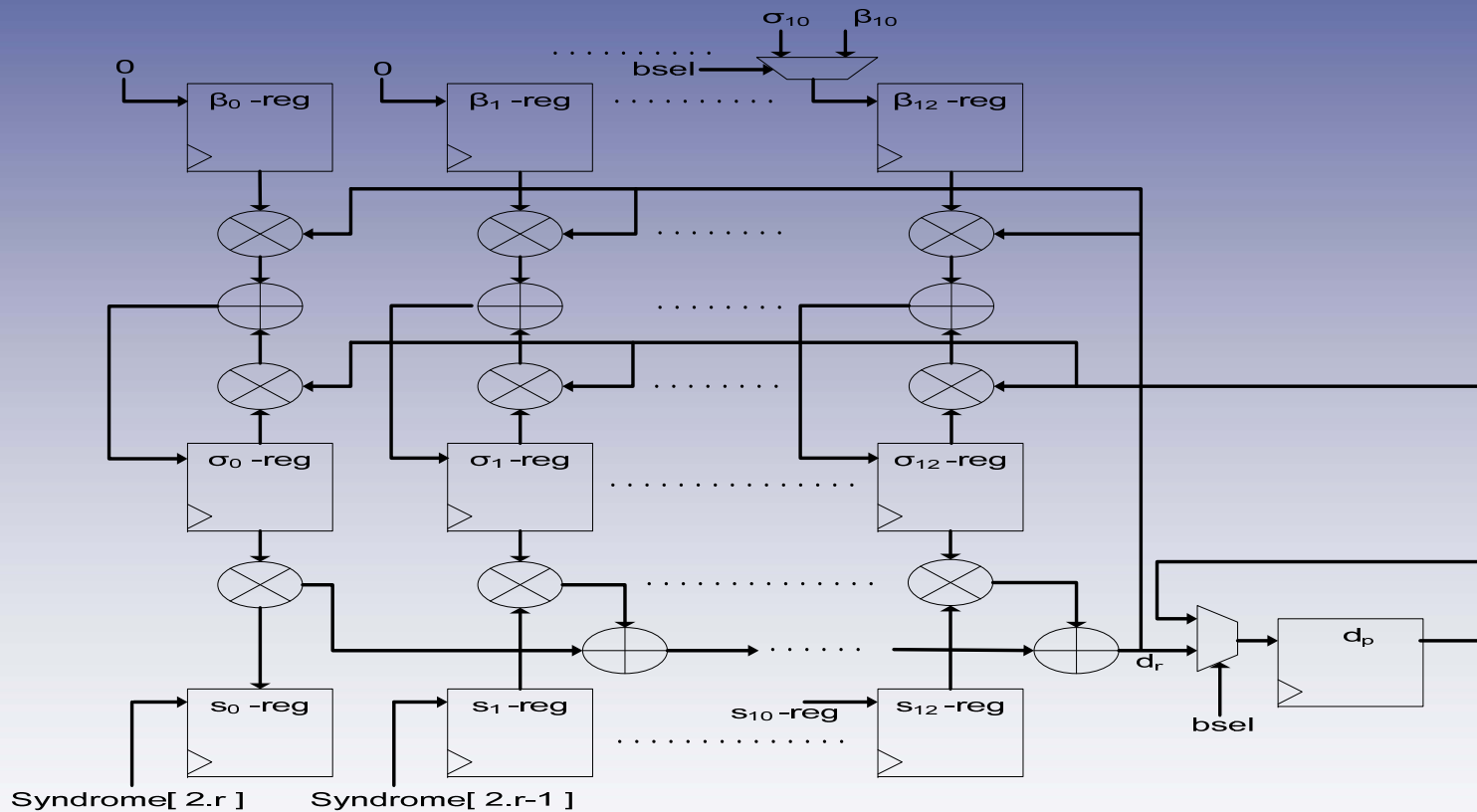
$$\sigma^{(0)}(x) = 1 + S_1 \cdot x$$

$$\beta^{(1)}(x) = \begin{cases} x^3 & \text{if } S_1 = 0 \\ x^2 & \text{if } S_1 \neq 0 \end{cases}$$

$$l_1 = \begin{cases} 0 & \text{if } S_1 = 0 \\ 1 & \text{if } S_1 \neq 0 \end{cases}$$

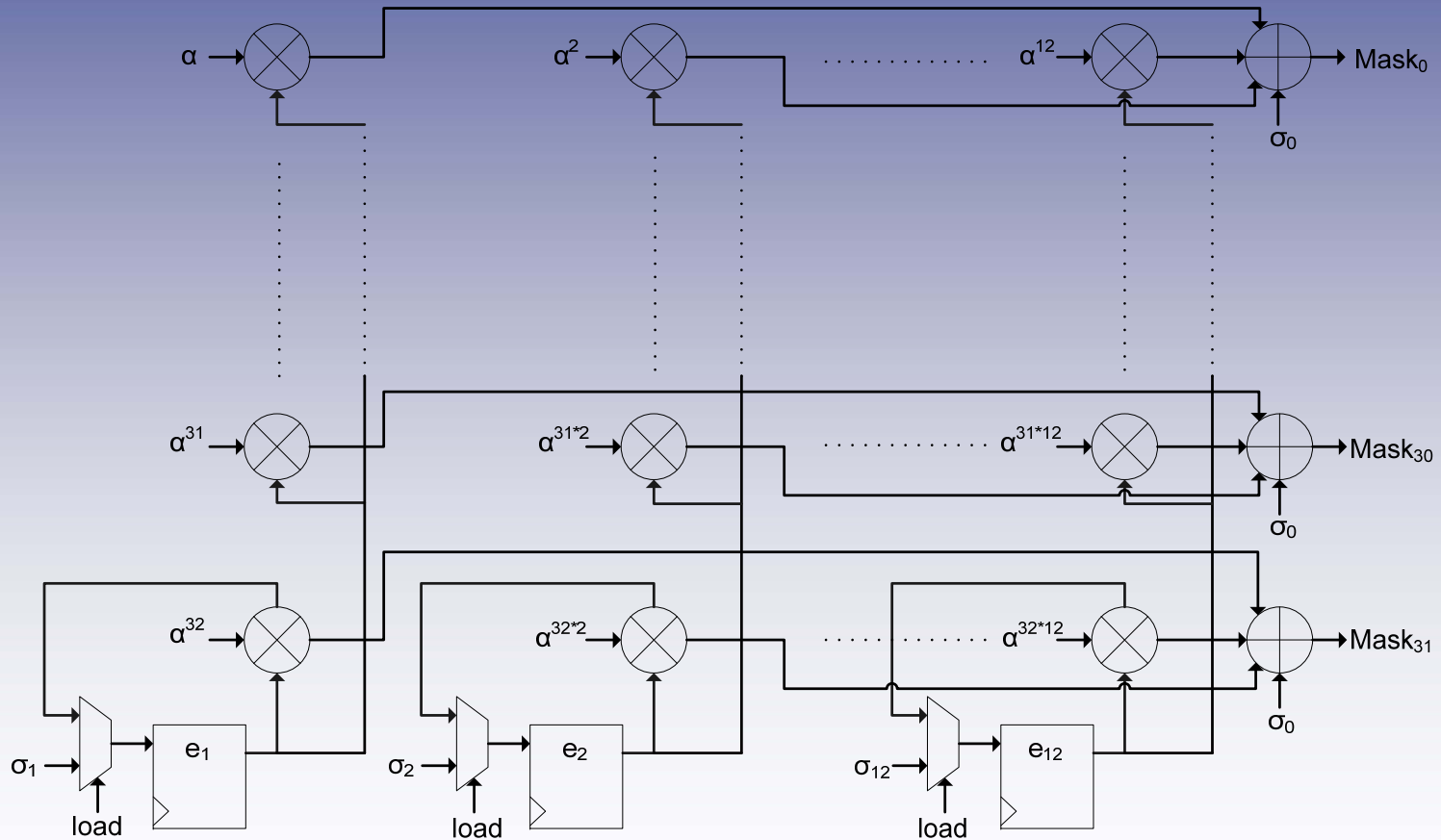
$$r = 1.$$

# Key Equation Solver

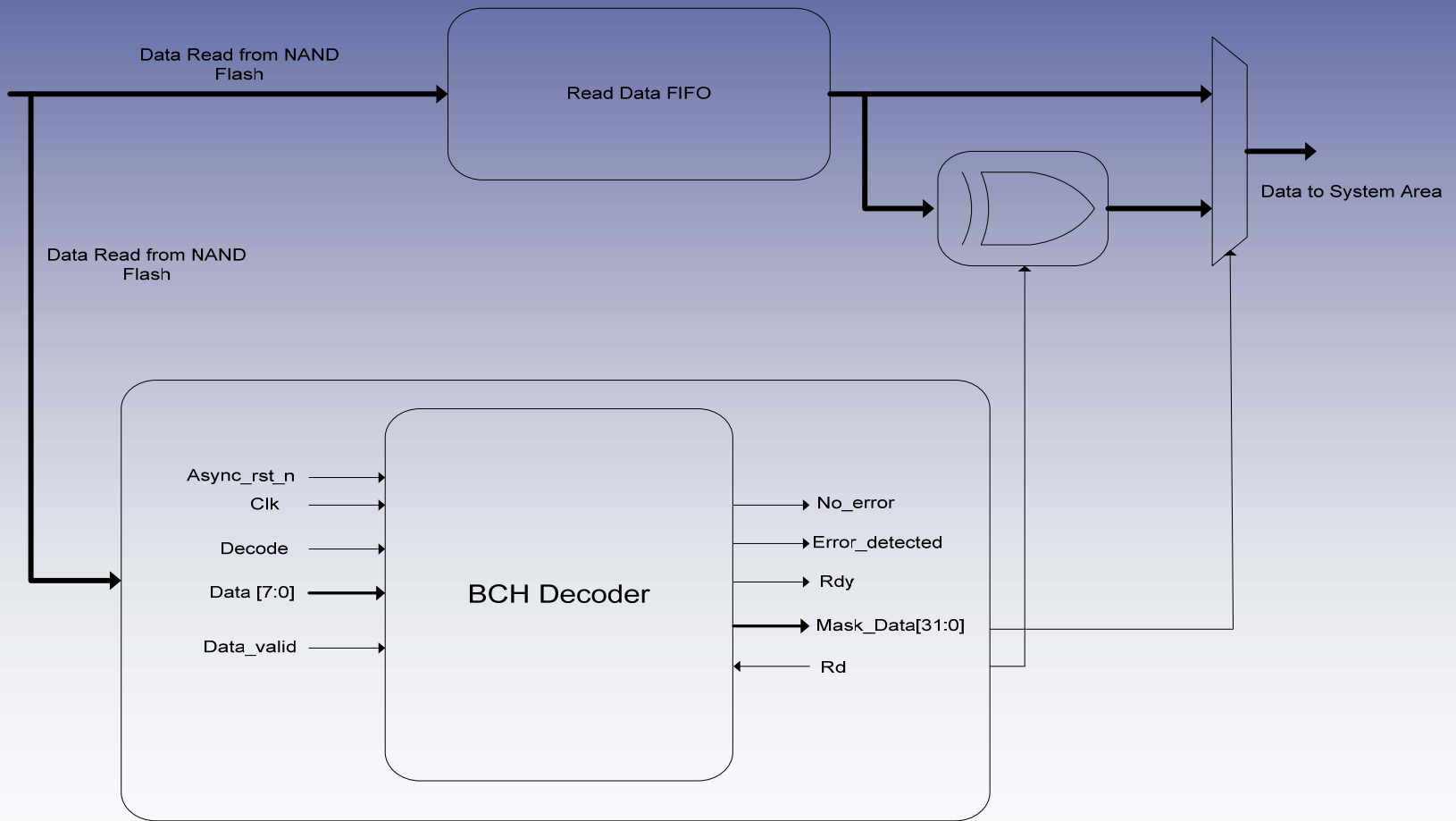


$$\sigma(x) = \sigma_0 + \sigma_1 x + \dots + \sigma_t x^t$$

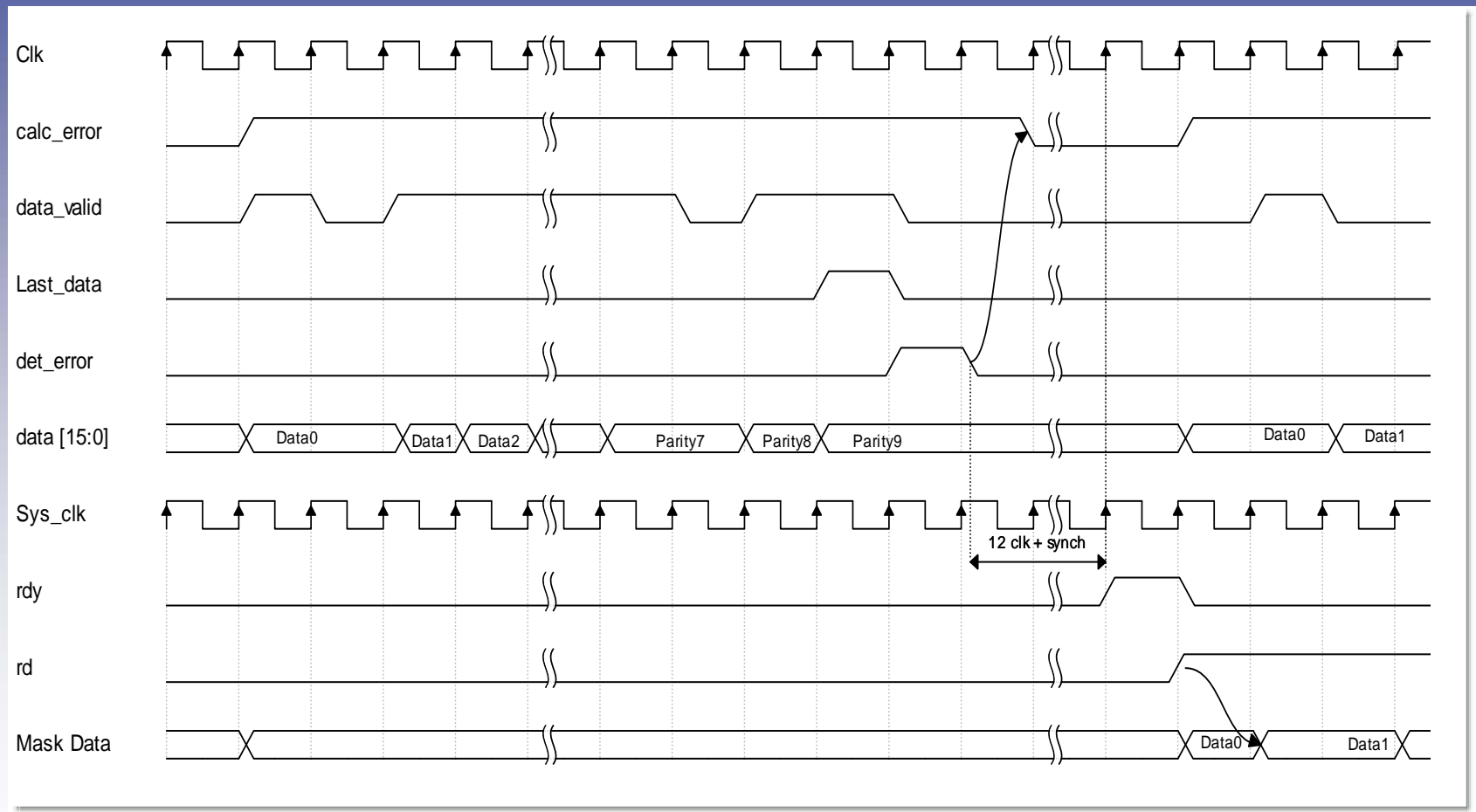
# Error Corrector Module (Parallel Chien Search algorithm)



# Integrating Decoder module



# Timing Diagram



## Throughput comparison

Decoder bit width	4 bit error (clk cycles)	8 bit error (clk cycles)	12 bit error (clk cycles)	16 bit error (clk cycles)
8 (512 clk)	516	520	524	528
16 (256 clk)	260	264	268	252
32 (128 clk)	132	136	140	144

# Summary

- Through put efficiency can be achieved by
  - Parallel bit processing on the BCH encoder
  - Parallel syndrome generation
  - Inversion less Berlekamp-Massey algorithm for Key equation solver
  - Parallel computation for the key equation solver
  - Pipe line stage to have more efficiency
  - Parallel chien search algorithm
  
- Question & Answer