

Fully-parallel linear error block coding and decoding – a Boolean approach

Hermann Meuth, Hochschule Darmstadt
Katrin Tschirpke, Hochschule Aschaffenburg

8th International Workshop on Boolean Problems, 2008

Background and Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic) Block Codes

Circuit Implementation

Code Definition and Optimisation

Example Results and Performance

Conclusions

Background and Basics of Error Coding

- ➔ During transmission and storage of digital data, corruption and errors may – and will – occur.
- ➔ Errors make digital data indecipherable and thus meaningless, unlike errors in analogue data. For digital data there are no insignificant errors in the common sense
- ➔ To allow for error checking, redundancy is required, i.e. additional data without extending information content.

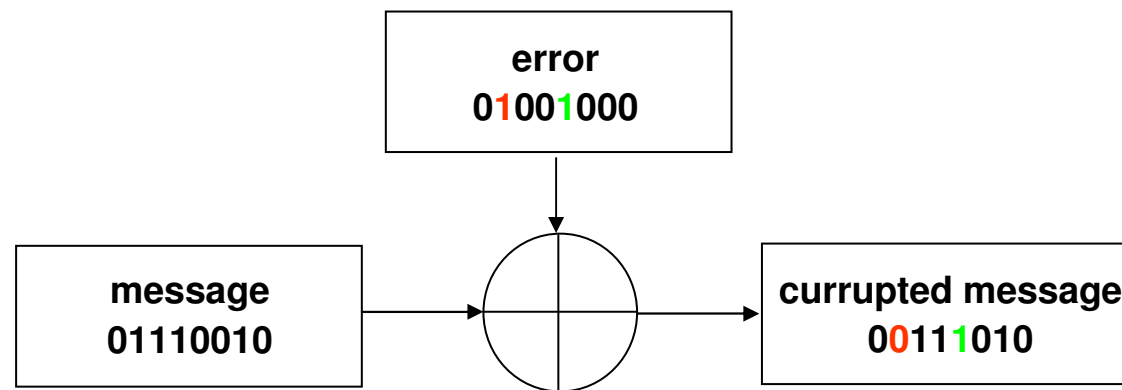
Background and Basics of Error Coding

- ➔ Digital data come in Bits. Redundancy thus adds further Bits without using them for information content

- ➔ Error coding may
 - alter message bits
 - may leave message bits unchanged.
Here, redundancy or parity Bits may be interleaved or appended en bloc.
This then is called a block code.

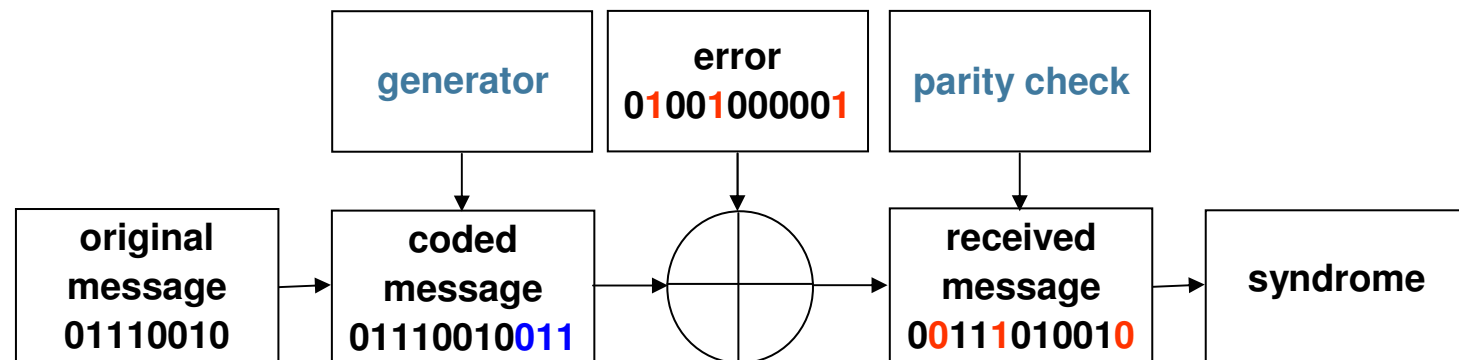
Error Coding: Error Model

- ➔ Errors are superimposed onto the message during storage or transmission
- ➔ Due to the binary property, errors are superimposed bit-by-bit.
- ➔ If an error occurs, this amounts to flipping the respective bit. Otherwise, the bit remains unaltered



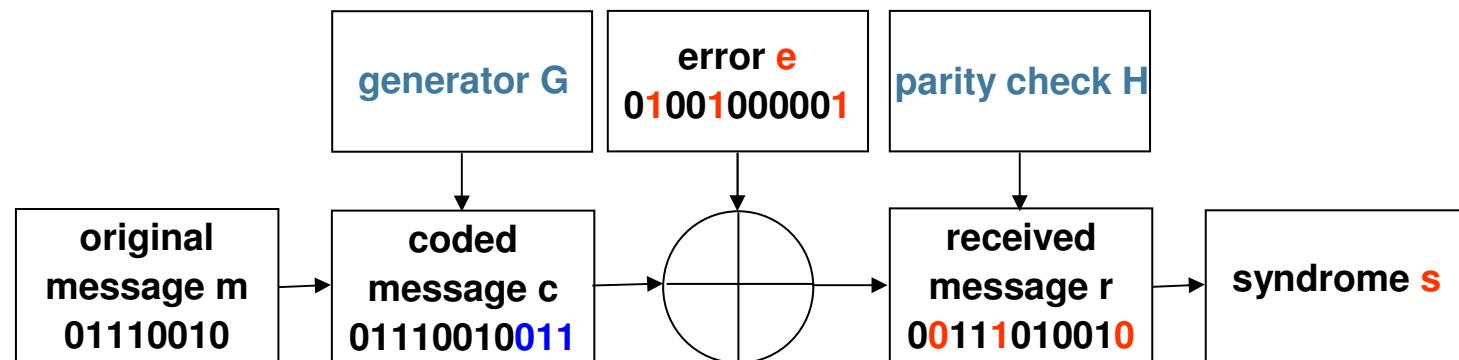
Error Coding and Decoding Model

- ➔ The message is to be transformed by means of a generator, thereby extending the message, to form the coded message
- ➔ The error impacts on the entire coded message
- ➔ The corrupted message is to be decoded by means of a parity check closely related to the generator



Linear Error Coding and Decoding

- ➔ transformations \underline{G} and \underline{H} may be represented by *matrices*
- ➔ messages \mathbf{m} , \mathbf{c} , \mathbf{r} , errors \mathbf{e} and syndromes \mathbf{s} as *vectors*
- ➔ algebra is modulo-2, with basic operations “ \pm ” and “ \cdot ”
- ➔ may be completely represented by Boolean Algebra, with operations “ \oplus ” (XOR) and “ \wedge ” (AND)



Linear Error Block Coding and Decoding

➔ message vector of k Bits

$$\mathbf{m}_k = (m_0, m_1, \dots, m_{k-1})$$

➔ parity-bit vector of $n-k$ Bits

$$\mathbf{b}_{n-k} = (b_0, b_1, \dots, b_{n-k-1}), \quad \mathbf{b}_{n-k} = \mathbf{m}_k \mathbf{P}_{k,n-k}$$

➔ coded message vector of n Bits

$$\begin{aligned} \mathbf{c}_n &= (c_0, c_1, \dots, c_{n-1}) \\ &= (b_0, b_1, \dots, b_{n-k-1}; m_0, m_1, \dots, m_{k-1}) = (\mathbf{b}_{n-k} | \mathbf{m}_k) \\ &= \mathbf{m}_k (\mathbf{P}_{k,n-k} | \mathbf{I}_k) = \mathbf{m}_k \mathbf{G}_{k,n} \end{aligned}$$

➔ generator matrix \mathbf{G} of k rows and n columns

$$\mathbf{G}_{k,n} = (\mathbf{P}_{k,n-k} | \mathbf{I}_k) = \begin{pmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & p_{1,1} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

Linear Cyclic Codes

➔ generator matrix has cyclic property

$$\mathbf{G}'_{k,n} = \begin{pmatrix}
 g_0 & g_1 & g_2 & \dots & g_{n-k-1} & g_{n-k} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
 0 & g_0 & g_1 & \dots & g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & 0 & \dots & 0 & 0 & 0 \\
 0 & 0 & g_0 & \dots & g_{n-k-3} & g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \dots & 0 & 0 & 0 \\
 \cdot & \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \dots & \cdot & \cdot & \cdot \\
 0 & 0 & 0 & \dots & 0 & g_0 & g_1 & g_2 & g_3 & \dots & 0 & 0 & 0 \\
 0 & 0 & 0 & \dots & 0 & 0 & g_0 & g_1 & g_2 & \dots & 0 & 0 & 0 \\
 \cdot & \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \cdot & \cdot & \dots & \cdot & \cdot & \cdot \\
 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & g_{n-k} & 0 & 0 \\
 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & g_{n-k-1} & g_{n-k} & 0 \\
 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & g_{n-k-2} & g_{n-k-1} & g_{n-k}
 \end{pmatrix}$$

➔ generator reduces to vector of $n-k+1$ elements, of which $n-k-1$ may freely be determined for optimum coding

$$\mathbf{g}_{n-k+1} = (1, g_1, g_2, \dots, g_{n-k-2}, g_{n-k-1}, 1)$$

➔ regularly implemented *serially* (for CRC)
 - in software by modulo-2 or polynomial division scheme
 - in hardware by **Linear-Feedback-Shift-Register**

Linear (Non-Cyclic) Block Codes

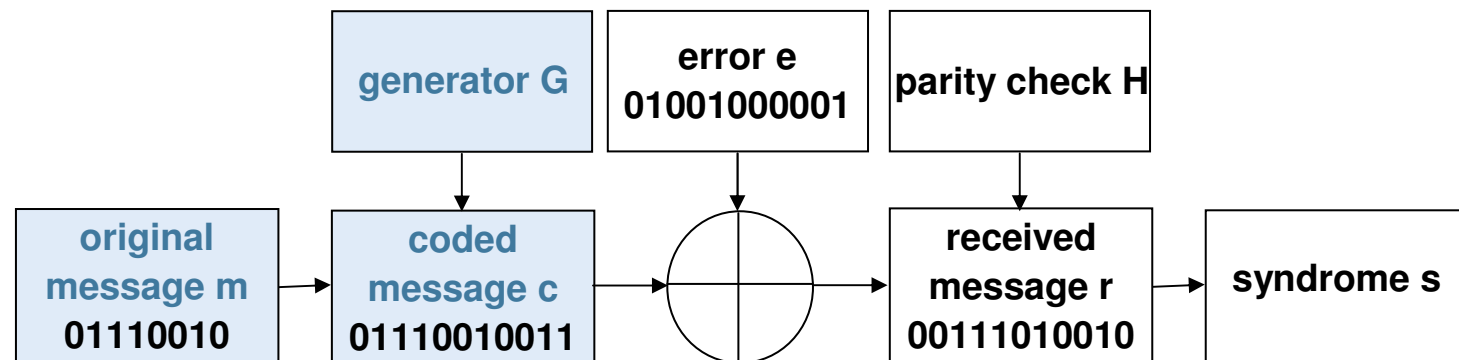
➔ *coding procedure* by means of the generator matrix \mathbf{G}

$$\mathbf{G}_{k,n} = (\mathbf{P}_{k,n-k} | \mathbf{I}_k)$$

➔ $(n-k)$ parities result from the $(n-k)$ Boolean conditions

$$\mathbf{b}_{n-k} = \mathbf{m}_k \mathbf{P}_{k,n-k}$$

$$b_j = m_0 \wedge p_{0,j} \oplus m_1 \wedge p_{1,j} \oplus m_2 \wedge p_{2,j} \oplus m_3 \wedge p_{3,j} \oplus \dots \oplus m_{k-2} \wedge p_{k-2,j} \oplus m_{k-1} \wedge p_{k-1,j}$$



Linear (Non-Cyclic) Block Codes

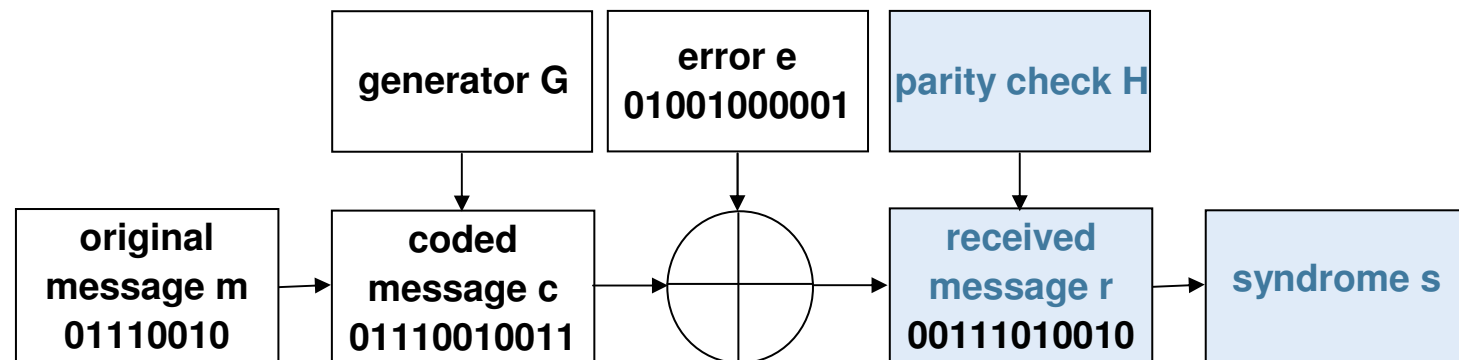
- *decoding procedure* by means of the transposed parity check matrix **H**

$$\mathbf{H}_{n,n-k} = (\mathbf{I}_{n-k} | \mathbf{P}_{n-k,k}^T)^T = \begin{pmatrix} \mathbf{I}_{n-k} \\ \mathbf{P}_{k,n-k} \end{pmatrix}$$

- $(n-k)$ syndromes result from the $(n-k)$ Boolean conditions

$$\mathbf{s}_{n-k} = \mathbf{r}_n \mathbf{H}_{n,n-k}$$

$$s_j = r_j \oplus r_{n-k} \wedge p_{0,j} \oplus r_{n-k+1} \wedge p_{1,j} \oplus r_{n-k+2} \wedge p_{2,j} \oplus \dots \oplus r_{n-2} \wedge p_{k-2,j} \oplus r_{n-1} \wedge p_{k-1,j}$$



Background and Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic) Block Codes

Circuit Implementation

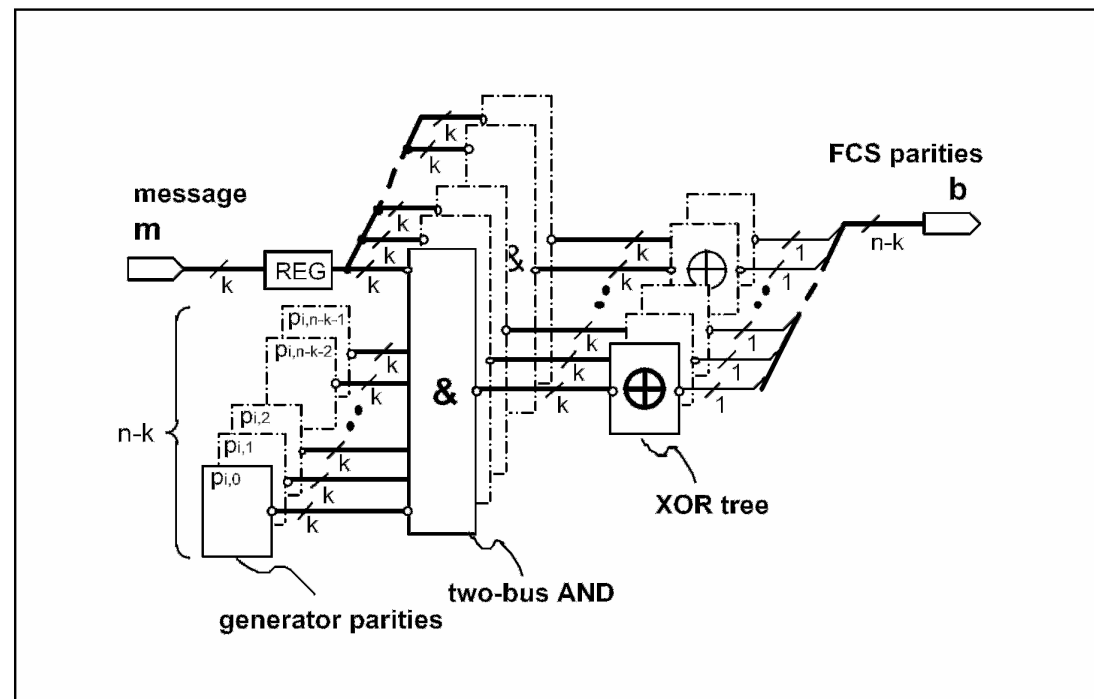
Code Definition and Optimisation

Example Results and Performance

Conclusions

Circuit Implementation of *Encoder*

➔ fully-parallel architecture, containing &- and XOR-trees



➔ readily formulated in HDL as configurable component, and implemented e.g. in FPGA/CPLD based hardware, which allows for real-time zero-clock delay solutions

Background and Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic) Block Codes

Circuit Implementation

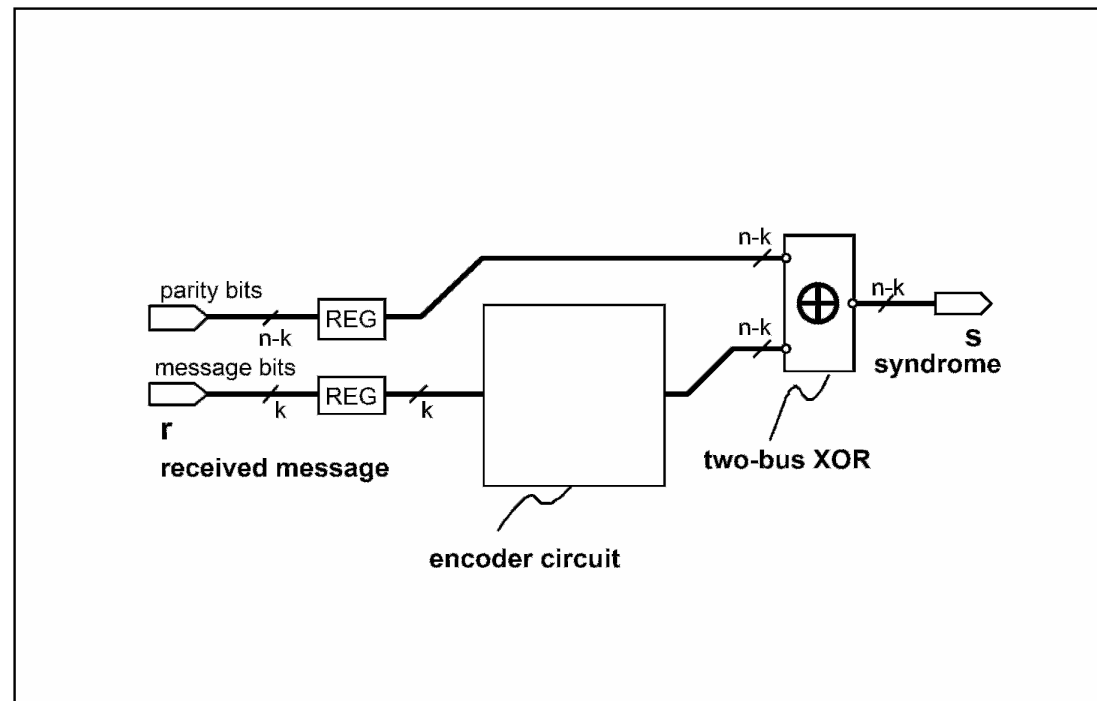
Code Definition and Optimisation

Example Results and Performance

Conclusions

Circuit Implementation of *Decoder*

➔ fully-parallel architecture, containing &- and XOR-trees



➔ readily formulated in HDL as configurable component, and implemented e.g. in FPGA/CPLD based hardware, which allows for real-time zero-clock delay solutions

Code Definition and Optimisation

➔ generally, a generator matrix \mathbf{G} of k rows and n columns contains $(n-k) \cdot k$ binary parameters

$$\mathbf{G}_{k,n} = (\mathbf{P}_{k,n-k} | \mathbf{I}_k) = \begin{pmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & p_{1,1} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

➔ to tailor error coding for optimum or specific feature error detection, these $(n-k) \cdot k$ parameters may be chosen suitably at will.

➔ for comparison, cyclic codes contain only $(n-k-1)$ such free parameters

Code Definition and Optimisation

➔ binary optimization problem

➔ input: all (2^n-1) possible error vectors of length n

$e^1=(0,0,\dots,0,1)$, $e^2=(0,0,\dots,1,0)$, $e^3=(0,0,\dots,1,1),\dots$ etc.

➔ output: the parity bits p_{ij} of generator $\underline{\mathbf{G}}$

$$\mathbf{G}_{k,n} = (\mathbf{P}_{k,n-k} | \mathbf{I}_k) = \begin{pmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & p_{1,1} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

and also the indicators $\delta(e^r)$, where

e^r is detected $\Leftrightarrow \delta(e^r) = 1$

e^r is not detected $\Leftrightarrow \delta(e^r) = 0$

Background and
Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic)
Block Codes

Circuit
Implementation

Code Definition and
Optimisation

Example Results and
Performance

Conclusions

Code Definition and Optimisation

- ➔ define an objective function in terms of the indicators $\delta(e^r)$

$$OF = \text{Max} \sum_{r=1}^{2^n-1} 2^{n+1-\ell(e^r)} \cdot \delta(e^r)$$

where $\ell(e^r)$ is the number of Bit flips = number of errors

- ➔ i.e. *OF* weights 1-Bit errors the highest
- ➔ subject to further constraints to improve on optimisation procedure

Background and
Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic)
Block Codes

Circuit
Implementation

Code Definition and
Optimisation

Example Results and
Performance

Conclusions

Code Definition and Optimisation

➔ constraints

➔ to detect an error e^r , at least one of the $n-k$ syndromes has to be non-zero

$$s_j(e^r) = e_j^r + \sum_{i=0}^{k-1} e_{i+n-k}^r p_{i,j} \text{ mod } 2$$

➔ the indicator $\delta(e^r)$ for a certain error must be bounded by the sum over all syndromes for this error

$$\sum_{j=0}^{n-k-1} s_j(e^r) \geq \delta(e^r)$$

➔ further, an additional technical constraint is introduced, where $M \gg 1$

$$\sum_{j=0}^{n-k-1} s_j(e^r) \leq M \cdot \delta(e^r)$$

➔ to actually detect all m -bit errors

$$M \cdot \delta(e^r) \geq m + 1 - \ell(e^r)$$

Code Definition and Optimisation

➔ binary optimization problem

➔ input: all (2^n-1) possible error vectors of length n

$e^1=(0,0,\dots,0,1)$, $e^2=(0,0,\dots,1,0)$, $e^3=(0,0,\dots,1,1),\dots$ etc.

➔ output: the parity bits p_{ij} of generator G

$$\mathbf{G}_{k,n} = (\mathbf{P}_{k,n-k} | \mathbf{I}_k) = \begin{pmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,n-k-1} & 1 & 0 & \dots & 0 \\ p_{1,0} & p_{1,1} & \dots & p_{1,n-k-1} & 0 & 1 & \dots & 0 \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot & \cdot & \cdot & \dots & \cdot \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & 0 & 0 & \dots & 1 \end{pmatrix}$$

and also the indicators $\delta(e^r)$, where

e^r is detected $\Leftrightarrow \delta(e^r) = 1$

e^r is not detected $\Leftrightarrow \delta(e^r) = 0$

Example Results and Performance

➔ performance measured against Hamming distance

$$d_{\min} = t_D + t_C + 1 = \text{Min} \sum_{q=0}^{n-1} CI_q \oplus CJ_q \forall I, J; I \neq J$$

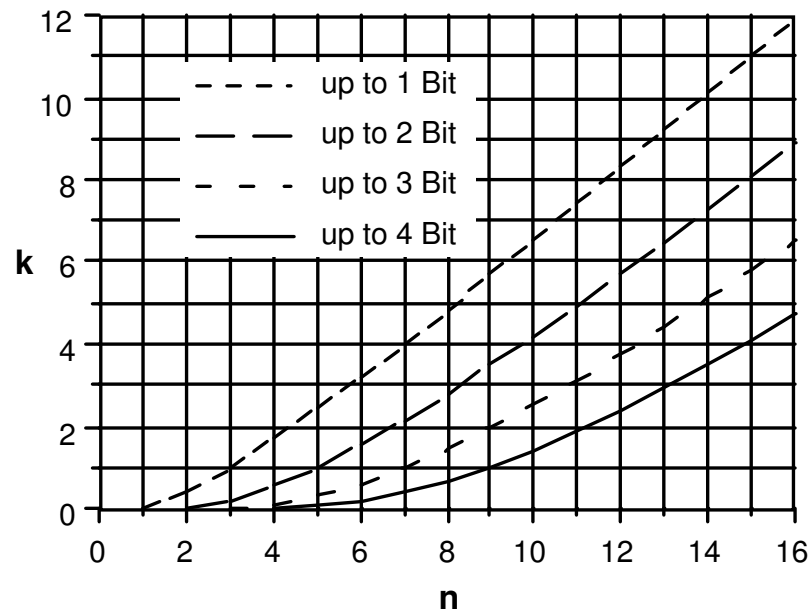
t_D = № of detectable, t_C = № of correctable errors, $t_D \geq t_C$

➔ achieved number of errors detected, t_D , depending on n, k

K\N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
11												1	1	1	2	3	11
10											1	1	1	2	3	3	10
9										1	1	1	2	3	3	3	9
8									1	1	1	2	3	3	3	4	8
7								1	1	1	2	3	3	3	4	5	7
6							1	1	1	2	3	3	3	4	5		6
5						1	1	1	2	3	3	3	4	5	6		5
4					1	1	2	3	3	3	4	5	5	6	7	7	4
3				1	1	2	3	3	3	4	5	5	6	7	7		3
2			1	1	2	3	3	4	5	5	6	7	7	8	9	9	2
1		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
K\N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

Example Results and Performance

- assigning to each syndrome $s_j(e^r)$ one and only one error e^r will constrain the doublets (n, k) according to Bose-Chaudhuri-Hocquenghem (BCH) to arrive at № of correctable errors t_C



- the achieved error detection performance is at the theoretical Hamming limit

Example Results and Performance

➔ sample generator results

➔ standard Hamming $n=7, k=4$
(standard CRC Code for polynomial $g(X)=1+X+X^3$)

$$P_{4,3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

➔ $n=8, k=4$
all errors up to
3 Bit are detected

$$P_{4,4} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

➔ $n=16, k=8$
all errors up to
4 Bit are detected

$$P_{8,8} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Background and
Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic)
Block Codes

Circuit
Implementation

Code Definition and
Optimisation

Example Results
and Performance

Conclusions

Example Results and Performance

- ➔ plain-vanilla FPGA-based simulations on actual hardware implementations show turn-around times of < 10 ns
- ➔ higher gate count is unavoidable in these fully parallel architectures

Background and
Basics

Error Coding

Linear Cyclic Codes

Linear (Non-Cyclic)
Block Codes

Circuit
Implementation

Code Definition and
Optimisation

Example Results and
Performance

Conclusions

Conclusions

- ➔ new error codes with much higher degree of optimisation potential were presented
- ➔ performance is achieved at the theoretical Hamming limit
- ➔ hardware implementations result as scalable, HDL suitable architectures with hitherto impossible turnaround times

Thank you for your attention!