

BOOLEAN FUNCTIONS

Duality, self-duality, neutrality, balancedness,
and orthogonal representations

Definitions • algebraic derivations • counting arguments

Duality swaps operations and constants

02

The dual of a Boolean expression simultaneously interchanges

Original	Dual
AND $\cdot$	OR $+$
OR $+$	AND $\cdot$
0	1
1	0

Variables and complemented variables remain unchanged.

De Morgan's laws produce the dual formula

03

$$f^d(x_1, \dots, x_n) = \neg f(\neg x_1, \dots, \neg x_n)$$

Complement every input, then complement the entire expression.

$$\neg(A + B) = \neg A \cdot \neg B \quad \neg(A \cdot B) = \neg A + \neg B$$

- The outer complement swaps OR and AND.
- It swaps constants because $\neg 0 = 1$ and $\neg 1 = 0$.
- Double complementation restores each variable: $\neg \neg x_i = x_i$.

Self-duality pairs opposite outputs

04

$$f(x) = f^d(x) = \neg f(\neg x)$$

Each input vector x belongs to one complementary pair $\{x, \neg x\}$.

$$x \quad f(x) = 0$$

$$\neg x \quad f(\neg x) = 1$$

The values may be reversed, but they must be different.

Examples of self-dual functions

05

Function	Expression	Reason
Identity	$f(x) = x$	$\neg f(\neg x) = x$
Complement	$f(x) = \neg x$	$\neg f(\neg x) = \neg x$
3-input majority	$M = xy + yz + zx$	Complementing all inputs changes the majority

Non-example: AND has $f(0,1) = f(1,0) = 0$,
although 01 and 10 are complements.

Counting self-dual functions

06

2^n input vectors form 2^{n-1} complementary pairs with 2 choices per pair

Pair	Allowed outputs
$\{x, \neg x\}$	$(0,1)$ or $(1,0)$
Independent choice	2 possibilities

$$\text{Number of self-dual functions} = 2^{(2^{n-1})}$$

Neutral functions preserve the output

07

In these slides, neutral means complement-invariant:

$$f(x) = f(\neg x)$$

- Both members of each complementary pair receive the same output.
- Each of the 2^{n-1} pairs independently receives 0 or 1.
- Examples include XOR, XNOR, and the two constant functions.

$$\text{Number of neutral functions} = 2^{(2^{n-1})}$$

Balanced functions split the truth table equally

08

A balanced function has exactly half 0 outputs and half 1 outputs.

$$\#0 = \#1 = 2^{n-1}$$

Choose which 2^{n-1} of the 2^n truth-table rows receive output 1.

$$\text{Number of balanced functions} = C(2^n, 2^{n-1})$$

Balancedness is a global count; it does not constrain complementary pairs.

Three truth-table properties

09

Property	Complementary pair $\{x, \neg x\}$	Count
Self-dual	Opposite outputs	$2^{(2^{n-1})}$
Neutral	Equal outputs	$2^{(2^{n-1})}$
Balanced	No pairwise rule; half of all rows output 1	$C(2^n, 2^{n-1})$

Every self-dual function is balanced. Neutral functions need not be balanced.

Balanced n neutral: $C(2^{n-1}, 2^{n-2}), n \geq 2$

Orthogonality belongs to terms and representations

10

For a DNF $F = m_1 + m_2 + \dots + m_k$, terms are pairwise orthogonal when

$$m_i \cdot m_j = 0 \quad \text{for every } i \neq j$$

No input assignment satisfies two different product terms.

ORTHOGONAL DNF

$$F = \neg x + xy$$
$$\neg x \cdot xy = 0$$

NONORTHOGONAL DNF

$$F = \neg x + y$$

terms overlap at $x=0, y=1$

Orthogonalization removes overlap

11

$$\mathbf{A + B = A + \neg A \cdot B} \quad \text{because} \quad \mathbf{A \cdot (\neg A \cdot B) = 0}$$

- Repeated decomposition converts overlapping DNF terms into disjoint terms.
- The canonical sum of minterms is always orthogonal because distinct minterms represent distinct rows.
- Every Boolean function therefore admits an orthogonal DNF.

There is no separate truth-table count of “orthogonal functions.”

Dual

$$f^d(x) = \neg f(\neg x)$$

Self-dual

$$f(x) = \neg f(\neg x)$$

Neutral

$$f(x) = f(\neg x)$$

Balanced

$$\#0 = \#1 = 2^{n-1}$$

Orthogonal DNF

$$m_i \cdot m_j = 0 \text{ for } i \neq j$$

Key distinction: the first four classify truth tables; orthogonality classifies a representation.

Error Detection and Correction Codes

Hamming distance, Hamming (7,4), and SECDED

COL215 Digital Logic and System Design

Semester I, 2026-27

Why error-control codes add redundancy

If every n -bit word carries data, a corrupted word may look like another valid word.



The unused n -bit words create separation between valid codewords.

The receiver checks the redundant bits to detect corruption or infer a correction.

A single parity bit detects every odd number of errors

Dataword	<table><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr></table>	1	0	1	1	Three 1s	Add 1 for even parity	
1	0	1	1					
Codeword	<table><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	1	1	1		
1	0	1	1	1				
Received	<table><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr></table>	1	0	0	1	1	Odd parity	ERROR DETECTED
1	0	0	1	1				

Two changed bits can preserve even parity. One parity bit cannot guarantee detection of two errors.

Hamming distance counts differing bit positions

$d(x,y)$ = number of 1s in $(x \text{ XOR } y)$

x	1	0	1	1	0	1	0	0
y	1	0	0	1	1	1	0	0
XOR	0	0	1	0	1	0	0	0

$$d(x,y) = 2$$

Minimum distance determines the guarantees

$d_{\min}$ is the smallest distance between any two distinct valid codewords.

DETECT

**up to $d_{\min} - 1$
errors**

CORRECT

up to $\text{floor}((d_{\min} - 1) / 2)$

EXAMPLES

Parity: $d_{\min} = 2$
Hamming: $d_{\min} = 3$
Extended Hamming:
 $d_{\min} = 4$

Detection and correction are different tasks. Correction must identify which valid codeword was sent.

Hamming (7,4) reserves power-of-two positions for parity

Position	1	2	3	4	5	6	7
Binary	001	010	011	100	101	110	111
Contents	p1	p2	d1	p4	d2	d3	d4

For $k = 4$ and $r = 3$: $2^3 = 8 = k + r + 1$

Eight syndromes encode no error plus one error in each of seven positions.

Each parity bit checks positions selected by one index bit

p1

1, 3, 5, 7

least-significant index bit = 1

p2

2, 3, 6, 7

middle index bit = 1

p4

4, 5, 6, 7

most-significant index bit = 1

Even-parity equations

$$p1 = d1 \text{ XOR } d2 \text{ XOR } d4$$

$$p2 = d1 \text{ XOR } d3 \text{ XOR } d4$$

$$p4 = d2 \text{ XOR } d3 \text{ XOR } d4$$

Example: encode dataword 1011

Place d1 d2 d3 d4 = 1 0 1 1 in positions 3, 5, 6, and 7

Position	1	2	3	4	5	6	7
Before	p1	p2	1	p4	0	1	1

p1 = 0 p2 = 1 p4 = 0

Codeword	0	1	1	0	0	1	1
-----------------	---	---	---	---	---	---	---

The syndrome names the erroneous position

Receiver parity checks

$s_1 = b_1 \oplus b_3 \oplus b_5 \oplus b_7$
 $s_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7$
 $s_4 = b_4 \oplus b_5 \oplus b_6 \oplus b_7$

$$S = (s_4 s_2 s_1)_2$$

$S = 000$ all checks pass

$S \neq 000$ under one-error
assumption, flip position
 S

Each position has a unique binary index, so it participates in a unique set of checks.

Example: syndrome 110 locates an error in position 6

Transmitted

0	1	1	0	0	1	1
---	---	---	---	---	---	---

Received

0	1	1	0	0	0	1
---	---	---	---	---	---	---

$$s1 = 0$$

$$s2 = 1$$

$$s4 = 1$$

$$S = 110_2 = 6$$

Flip position 6, recover 0110011, and extract dataword 1011.

Ordinary Hamming code can miscorrect two errors

With errors at positions i and j

$$S = i \text{ XOR } j$$

The result usually names a third position.

A decoder that flips every nonzero syndrome may change a correct bit and silently produce the wrong dataword.

Hamming (7,4) needs one more parity bit to distinguish single and double errors.

Extended Hamming (8,4) adds one overall parity bit

Hamming codeword

0	1	1	0	0	1	1
---	---	---	---	---	---	---

+

p8

=

8 bits

Choose p8 so that all eight bits have even parity.

0	1	1	0	0	1	1	0
---	---	---	---	---	---	---	---

dmin = 4

SECDED: correct one error, detect two

SECDED combines the Hamming syndrome with overall parity

Syndrome S	Parity P	Interpretation	Action
000	0	No error	Pass unchanged
nonzero	1	One error in positions 1-7	Flip position S
000	1	Error in overall parity bit	Flip position 8
nonzero	0	Two-bit error	Report uncorrectable

Single error example

$S = 110$, $P = 1$
Correct position 6

Double error example

$S = 101$, $P = 0$
Detected, not corrected

Guarantee assumes at most two errors.