Contents

Section outline

1. **Handling numbers**
   - Radix number systems
   - Complementation
   - Conversion of bases
   - Binary to BCD
   - Binary codes
   - Linear codes
   - Error detecting code
   - Error correcting code
   - Mininum bits for 1-bit ECC
   - Mininum bits for 1-bit EDC

Radix number systems

- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
  $0 \le a_i < b$, MSB: a_m, LSB: a_{-p}
- $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

Radix number systems

- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
  $0 \leq a_i < b$, MSB: a_m, LSB: a_{-p}
- $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- Integer part: $a_m b^m + \ldots + a_1 b + a_0$
- Fractional part: $a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Radix number systems

- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
  $0 \leq a_i < b$, MSB: a_m, LSB: a_{-p}
- $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- Integer part: $a_m b^m + \ldots + a_1 b + a_0$
- Fractional part: $a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
- Common bases: 10 – decimal, 2 – binary, 8 – octal, 16 – hexadecimal
- $1101.01 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25$

Radix number systems

- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
  $0 \le a_i < b$, MSB: a_m, LSB: a_{-p}
- $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- Integer part: $a_m b^m + \ldots + a_1 b + a_0$
- Fractional part: $a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
- Common bases: 10 – decimal, 2 – binary, 8 – octal, 16 – hexadecimal
- $1101.01 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25$
- $31.1_4 = ?$

Radix number systems

- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
  $0 \leq a_i < b$, MSB: a_m, LSB: a_{-p}
- $123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- Integer part: $a_m b^m + \ldots + a_1 b + a_0$
- Fractional part: $a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$
- Common bases: 10 – decimal, 2 – binary, 8 – octal, 16 – hexadecimal
- $1101.01 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25$
- $31.1_4 = ?$
- $15.2_8 = ?$

Numbers in some bases

Base					
2	4	8	10	12	16
0000	0	0	0	0	0
0001	1	1	1	1	1
0010	2	2	2	2	2
0011	3	3	3	3	3
0100	10	4	4	4	4
0101	11	5	5	5	5
0110	12	6	6	6	6
0111	13	7	7	7	7

Numbers in some bases

Base					
2	4	8	10	12	16
0000	0	0	0	0	0
0001	1	1	1	1	1
0010	2	2	2	2	2
0011	3	3	3	3	3
0100	10	4	4	4	4
0101	11	5	5	5	5
0110	12	6	6	6	6
0111	13	7	7	7	7
1000	20	10	8	8	8
1001	21	11	9	9	9
1010	22	12	10	α	A
1011	23	13	11	β	B
1100	30	14	12	10	C
1101	31	15	13	11	D
1110	32	16	14	12	E
1111	33	17	15	13	F

Complementation

- Complement of digit a, denoted a', in base b is ${a_b}' = (b-1)_b - a_b$
- Binary: $a'_2 = 1_2 - a_2$, $0' = 1$, $1' = 0$
- Decimal: $a'_{10} = 9_{10} - a_{10}$
  $0' = 9$, $1' = 8$, $2' = 7$, $3' = 6$, $4' = 5$, $5' = 4$, $6' = 3$, $7' = 2$, $8' = 1$, $9' = 0$
- Octal: $a'_8 = 7_8 - a_8$

Complementation

- Complement of digit a, denoted a', in base b is ${a_b}' = (b-1)_b - a_b$
- Binary: $a'_2 = 1_2 - a_2$, $0' = 1$, $1' = 0$
- Decimal: $a'_{10} = 9_{10} - a_{10}$
  $0' = 9$, $1' = 8$, $2' = 7$, $3' = 6$, $4' = 5$, $5' = 4$, $6' = 3$, $7' = 2$, $8' = 1$, $9' = 0$
- Octal: $a'_8 = 7_8 - a_8$
  $0' = 7$, $1' = 6$, $2' = 5$, $3' = 4$, $4' = 3$, $5' = 2$, $6' = 1$, $7' = 0$

Complementation

- Complement of digit a, denoted a', in base b is $a_b{}' = (b-1)_b - a_b$
- Binary: $a_2' = 1_2 - a_2$, $0' = 1$, $1' = 0$
- Decimal: $a_{10}' = 9_{10} - a_{10}$
  $0' = 9$, $1' = 8$, $2' = 7$, $3' = 6$, $4' = 5$, $5' = 4$, $6' = 3$, $7' = 2$, $8' = 1$, $9' = 0$
- Octal: $a_8' = 7_8 - a_8$
  $0' = 7$, $1' = 6$, $2' = 5$, $3' = 4$, $4' = 3$, $5' = 2$, $6' = 1$, $7' = 0$
- For, $N = a_m b^m + \ldots + a_1 b + a_0$, let $M = a_m' b^m + \ldots + a_1' b + a_0'$

$\therefore\ M = (b - 1 - a_m)\, b^m + \ldots + (b - 1 - a_1)\, b + (b - 1 - a_0)$

$$\Rightarrow\ M = \sum_{i=1}^{m+1} b^i - \sum_{i=0}^{m} b^i - N = \left(b^{m+1} - 1\right) - N$$

- Diminished radix complement of N is $\left(b^{m+1} - 1\right) - N = M$
- Radix complement of N is $b^{m+1} - N = M + 1 = N'$

Complementation

- Complement of digit a, denoted a', in base b is ${a_b}' = (b-1)_b - a_b$
- Binary: $a'_2 = 1_2 - a_2$, $0' = 1$, $1' = 0$
- Decimal: $a'_{10} = 9_{10} - a_{10}$
  $0' = 9$, $1' = 8$, $2' = 7$, $3' = 6$, $4' = 5$, $5' = 4$, $6' = 3$, $7' = 2$, $8' = 1$, $9' = 0$
- Octal: $a'_8 = 7_8 - a_8$
  $0' = 7$, $1' = 6$, $2' = 5$, $3' = 4$, $4' = 3$, $5' = 2$, $6' = 1$, $7' = 0$
- For, $N = a_m b^m + \ldots + a_1 b + a_0$, let $M = a'_m b^m + \ldots + a'_1 b + a'_0$

$\therefore$ $M = (b - 1 - a_m)\, b^m + \ldots + (b - 1 - a_1)\, b + (b - 1 - a_0)$

$$\Rightarrow M = \sum_{i=1}^{m+1} b^i - \sum_{i=0}^{m} b^i - N = \left(b^{m+1} - 1\right) - N$$

- Diminished radix complement of N is $\left(b^{m+1} - 1\right) - N = M$
- Radix complement of N is $b^{m+1} - N = M + 1 = N'$
- $P - N = P + N' \mod b^m$ (for m digits)

Complementation (contd.)

Example (Decimal subtraction)

- $321 - 123 = 198$
- Ten's complement of 123:

Complementation (contd.)

Example (Decimal subtraction)

- $321 - 123 = 198$
- Ten's complement of 123: $876 + 1 = 877$
- $321 + 876 = 1198 = 198 \mod 10^3$

Complementation (contd.)

Example (Decimal subtraction)

- $321 - 123 = 198$
- Ten's complement of 123: $876 + 1 = 877$
- $321 + 876 = 1198 = 198 \mod 10^3$

Example (Binary subtraction)

- $1\ 0100\ 0001 - 0\ 0111\ 1011 = 0\ 1100\ 0110$
- 2's complement of 0 0111 1011:

Complementation (contd.)

Example (Decimal subtraction)

- $321 - 123 = 198$
- Ten's complement of 123: $876 + 1 = 877$
- $321 + 876 = 1198 = 198 \mod 10^3$

Example (Binary subtraction)

- $1\ 0100\ 0001 - 0\ 0111\ 1011 = 0\ 1100\ 0110$
- 2's complement of 0 0111 1011: $1\ 1000\ 0100 + 1 = 1\ 1000\ 0101$
- $1\ 0100\ 0001 + 1\ 1000\ 0101 = 10\ 1100\ 0110 = 0\ 1100\ 0110 \mod 2^9$

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011
6	0110	1001	1010

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011
6	0110	1001	1010
7	0111	1000	1001

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011
6	0110	1001	1010
7	0111	1000	1001
8	1000	0111	1000

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011
6	0110	1001	1010
7	0111	1000	1001
8	1000	0111	1000
9	1001	0110	0111

Complementation (contd.)

	Num	twos'	two's
0	0000	1111	0000
1	0001	1110	1111
2	0010	1101	1110
3	0011	1100	1101
4	0100	1011	1100
5	0101	1010	1011
6	0110	1001	1010
7	0111	1000	1001
8	1000	0111	1000
9	1001	0110	0111

	Num	twos'	two's
0	0 0000	1 1111	0 0000
1	0 0001	1 1110	1 1111
2	0 0010	1 1101	1 1110
3	0 0011	1 1100	1 1101
4	0 0100	1 1011	1 1100
5	0 0101	1 1010	1 1011
6	0 0110	1 1001	1 1010
7	0 0111	1 1000	1 1001
8	0 1000	1 0111	1 1000
9	0 1001	1 0110	1 0111
10	0 1010	1 0101	1 0110
11	0 1011	1 0100	1 0101
12	0 1100	1 0011	1 0100
13	0 1101	1 0010	1 0011
14	0 1110	1 0001	1 0010
15	0 1111	1 0000	1 0001

Conversion of bases

- Number N_{b_1} in base b_1 to be converted to base b_2 as $A.B$ where A is the integral part and B is the fractional part
- Thus, $N_{b_1} = \underbrace{a_m {b_2}^m + \ldots + a_1 b_2 + a_0}_{A} + \underbrace{a_{-1} {b_2}^{-1} + \ldots + a_{-p} {b_2}^{-p}}_{B}$
- Now, $A = b_2 \cdot \underbrace{\left(a_m {b_2}^{m-1} + \ldots + a_1 \right)}_{A'} + a_0$
- Least significant digit of A_{b_2} is the remainder of A divided by b_2
- If $A' = 0$, terminate, otherwise, apply procedure recursively to A'

Conversion of bases

- Number N_{b_1} in base b_1 to be converted to base b_2 as $A.B$ where A is the integral part and B is the fractional part
- Thus, $N_{b_1} = \underbrace{a_m {b_2}^m + \ldots + a_1 b_2 + a_0}_{A} + \underbrace{a_{-1} {b_2}^{-1} + \ldots + a_{-p} {b_2}^{-p}}_{B}$
- Now, $A = b_2 \cdot \underbrace{\left(a_m {b_2}^{m-1} + \ldots + a_1\right)}_{A'} + a_0$
- Least significant digit of A_{b_2} is the remainder of A divided by b_2
- If $A' = 0$, terminate, otherwise, apply procedure recursively to A'
- Also, $b_2 \cdot B = a_{-1} + \underbrace{a_{-2} {b_2}^{-1} + \ldots + a_{-p} {b_2}^{1-p}}_{B'}$
- First digit of B_{b_2} is the integral part of B multiplied by b_2
- If $B' = 0$, terminate, otherwise, apply procedure recursively to B'

Conversion of bases

- Number N_{b_1} in base b_1 to be converted to base b_2 as $A.B$ where A is the integral part and B is the fractional part
- Thus, $N_{b_1} = \underbrace{a_m {b_2}^m + \ldots + a_1 b_2 + a_0}_{A} + \underbrace{a_{-1} {b_2}^{-1} + \ldots + a_{-p} {b_2}^{-p}}_{B}$
- Now, $A = b_2 \cdot \underbrace{\left(a_m {b_2}^{m-1} + \ldots + a_1\right)}_{A'} + a_0$
- Least significant digit of A_{b_2} is the remainder of A divided by b_2
- If $A' = 0$, terminate, otherwise, apply procedure recursively to A'
- Also, $b_2 \cdot B = a_{-1} + \underbrace{a_{-2} {b_2}^{-1} + \ldots + a_{-p} {b_2}^{1-p}}_{B'}$
- First digit of B_{b_2} is the integral part of B multiplied by b_2
- If $B' = 0$, terminate, otherwise, apply procedure recursively to B'
- Direct evaluation, if arithmetic in base b_2 is convenient

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Q_i	r_i	
68	4	a_0

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Q_i	r_i	
68	4	a_0
8	4	a_1
1	0	a_2
	1	a_3

$548_{10} = 1044_8$

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Q_i	r_i	
68	4	a_0
8	4	a_1
1	0	a_2
	1	a_3

$548_{10} = 1044_8$

Example (345_{10} to base 6)

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Q_i	r_i	
68	4	a_0
8	4	a_1
1	0	a_2
	1	a_3

$548_{10} = 1044_8$

Example (345_{10} to base 6)

Q_i	r_i	
57	3	a_0

Conversion of bases (contd.

Example (548_{10} to octal (base 8))

Q_i	r_i	
68	4	a_0
8	4	a_1
1	0	a_2
	1	a_3

$548_{10} = 1044_8$

Example (345_{10} to base 6)

Q_i	r_i	
57	3	a_0
9	3	a_1
1	3	a_2
	1	a_3

$245_{10} = 1333_6$

Conversion of bases (contd.

Example (0.3125_{10} to base 8)

Conversion of bases (contd.

Example (0.3125_{10} to base 8)

- $0.3125 \times 8 = 2.5000$
- $0.5000 \times 8 = 4.0000$
- $a_{-1} = 2$, $a_{-2} = 4$
- $0.3125_{10} = 0.24_8$

Conversion of bases (contd.

Example (0.3125_{10} to base 8)

- $0.3125 \times 8 = 2.5000$
- $0.5000 \times 8 = 4.0000$
- $a_{-1} = 2$, $a_{-2} = 4$
- $0.3125_{10} = 0.24_8$

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Example (432.2_8 to decimal)

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Example (432.2_8 to decimal)

$432.2_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = 282.25_{10}$

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Example (432.2_8 to decimal)

$432.2_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = 282.25_{10}$

Example (1101.01_2 to decimal)

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Example (432.2_8 to decimal)

$432.2_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = 282.25_{10}$

Example (1101.01_2 to decimal)

$1101.01_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25_{10}$

Example (345_{10} to base 12)

Conversion of bases (contd.)

- Number in base b_1 to be converted to base b_2
- Direct evaluation may be carried out if arithmetic of b_2, if convenient
- $N = a_m b^m + \ldots + a_1 b + a_0 + a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}$

Example (432.2_8 to decimal)

$432.2_8 = 4 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 2 \times 8^{-1} = 282.25_{10}$

Example (1101.01_2 to decimal)

$1101.01_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 13.25_{10}$

Example (345_{10} to base 12)

$3 \times A^2 + 4 \times A + 5 = 3 \times 84 + 34 + 5 = 210 + 39 = 249_{12}$

Binary to BCD

$d = 0 - 9$: binary and BCD are identical

Binary to BCD

$d = 0 - 9$: binary and BCD are identical

$d = 10 - 15$: 1 goes to the next higher place, $d - 10$ in current place

Binary to BCD

$d = 0 - 9$: binary and BCD are identical

$d = 10 - 15$: 1 goes to the next higher place, $d - 10$ in current place

Alternately $d + 6 \mod 16$ in current place, if $d \geq 10$

$d = 12$: $d + 6 = 18 \mod 16 = 2$, 1 goes to next higher place

Binary to BCD

$d = 0 - 9$**:** binary and BCD are identical

$d = 10 - 15$**:** 1 goes to the next higher place, $d - 10$ in current place

Alternately $d + 6 \mod 16$ in current place, if $d \geq 10$

$d = 12$**:** $d + 6 = 18 \mod 16 = 2$, 1 goes to next higher place

$12_{10} = 1100_2$, 1100 + 0110 = 1 0010

Binary to BCD

$d = 0 - 9$: binary and BCD are identical

$d = 10 - 15$: 1 goes to the next higher place, $d - 10$ in current place

Alternately $d + 6 \mod 16$ in current place, if $d \geq 10$

$d = 12$: $d + 6 = 18 \mod 16 = 2$, 1 goes to next higher place

$12_{10} = 1100_2$, 1100 + 0110 = 1 0010

NB: LSB is unaffected, because LSB of $6_{10} = 0$

If bits are handled sequentially, 3 can be added (instead of 6) and then shifted left

Binary to BCD

$d = 0 - 9$: binary and BCD are identical

$d = 10 - 15$: 1 goes to the next higher place, $d - 10$ in current place

Alternately $d + 6 \mod 16$ in current place, if $d \geq 10$

$d = 12$: $d + 6 = 18 \mod 16 = 2$, 1 goes to next higher place

$12_{10} = 1100_2$, 1100 + 0110 = 1 0010

NB: LSB is unaffected, because LSB of $6_{10} = 0$

If bits are handled sequentially, 3 can be added (instead of 6) and then shifted left

110 + 011 = 1001 $\longrightarrow$ 1 0010

To be repeated until conversion is complete

Name Shift-and-add-3 or double-dabble

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100
L Sft	0000	0000	0000	0010	0011	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100
L Sft	0000	0000	0000	0010	0011	1011111001101100
L Sft	0000	0000	0000	0100	0111	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100
L Sft	0000	0000	0000	0010	0011	1011111001101100
L Sft	0000	0000	0000	0100	0111	1011111001101100
Add 3	0000	0000	0000	0100	1010	1011111001101100
L Sft	0000	0000	0000	1001	0101	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100
L Sft	0000	0000	0000	0010	0011	1011111001101100
L Sft	0000	0000	0000	0100	0111	1011111001101100
Add 3	0000	0000	0000	0100	1010	1011111001101100
L Sft	0000	0000	0000	1001	0101	1011111001101100
Add 3	0000	0000	0000	1100	1000	1011111001101100
L Sft	0000	0000	0001	1001	0000	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
L Sft	0000	0000	0000	0000	0001	1011111001101100
L Sft	0000	0000	0000	0000	0010	1011111001101100
L Sft	0000	0000	0000	0000	0101	1011111001101100
Add 3	0000	0000	0000	0000	1000	1011111001101100
L Sft	0000	0000	0000	0001	0001	1011111001101100
L Sft	0000	0000	0000	0010	0011	1011111001101100
L Sft	0000	0000	0000	0100	0111	1011111001101100
Add 3	0000	0000	0000	0100	1010	1011111001101100
L Sft	0000	0000	0000	1001	0101	1011111001101100
Add 3	0000	0000	0000	1100	1000	1011111001101100
L Sft	0000	0000	0001	1001	0000	1011111001101100
Add 3	0000	0000	0001	1100	0000	1011111001101100
L Sft	0000	0000	0011	1000	0000	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100
Add 3	0000	0001	1000	0010	0011	1011111001101100
L Sft	0000	0011	0000	0100	0110	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100
Add 3	0000	0001	1000	0010	0011	1011111001101100
L Sft	0000	0011	0000	0100	0110	1011111001101100
Add 3	0000	0011	0000	0100	1001	1011111001101100
L Sft	0000	0110	0000	1001	0011	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3 L Sft	0000 0000	0000 0000	0011 0111	1011 0110	0000 0001	1011111001101100 1011111001101100
Add 3 L Sft	0000 0000	0000 0001	1010 0101	1001 0010	0001 0011	1011111001101100 1011111001101100
Add 3 L Sft	0000 0000	0001 0011	1000 0000	0010 0100	0011 0110	1011111001101100 1011111001101100
Add 3 L Sft	0000 0000	0011 0110	0000 0000	0100 1001	1001 0011	1011111001101100 1011111001101100
Add 3 L Sft	0000 0001	1001 0010	0000 0001	1100 1000	0011 0111	1011111001101100 1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100
Add 3	0000	0001	1000	0010	0011	1011111001101100
L Sft	0000	0011	0000	0100	0110	1011111001101100
Add 3	0000	0011	0000	0100	1001	1011111001101100
L Sft	0000	0110	0000	1001	0011	1011111001101100
Add 3	0000	1001	0000	1100	0011	1011111001101100
L Sft	0001	0010	0001	1000	0111	1011111001101100
Add 3	0001	0010	0001	1011	1010	1011111001101100
L Sft	0010	0100	0011	0111	0100	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100
Add 3	0000	0001	1000	0010	0011	1011111001101100
L Sft	0000	0011	0000	0100	0110	1011111001101100
Add 3	0000	0011	0000	0100	1001	1011111001101100
L Sft	0000	0110	0000	1001	0011	1011111001101100
Add 3	0000	1001	0000	1100	0011	1011111001101100
L Sft	0001	0010	0001	1000	0111	1011111001101100
Add 3	0001	0010	0001	1011	1010	1011111001101100
L Sft	0010	0100	0011	0111	0100	1011111001101100
Add 3	0010	0100	0011	1010	0100	1011111001101100
L Sft	0100	1000	0111	0100	1000	1011111001101100

Binary of 48748 to BCD example

Op	B4	B3	B2	B1	B0	48748
Add 3	0000	0000	0011	1011	0000	1011111001101100
L Sft	0000	0000	0111	0110	0001	1011111001101100
Add 3	0000	0000	1010	1001	0001	1011111001101100
L Sft	0000	0001	0101	0010	0011	1011111001101100
Add 3	0000	0001	1000	0010	0011	1011111001101100
L Sft	0000	0011	0000	0100	0110	1011111001101100
Add 3	0000	0011	0000	0100	1001	1011111001101100
L Sft	0000	0110	0000	1001	0011	1011111001101100
Add 3	0000	1001	0000	1100	0011	1011111001101100
L Sft	0001	0010	0001	1000	0111	1011111001101100
Add 3	0001	0010	0001	1011	1010	1011111001101100
L Sft	0010	0100	0011	0111	0100	1011111001101100
Add 3	0010	0100	0011	1010	0100	1011111001101100
L Sft	0100	1000	0111	0100	1000	1011111001101100
End	4	8	7	4	8	

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1} b_{n-2} \ldots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1} ... d_{\jmath} ... d_0$

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\ldots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}...d_\jmath...d_0$, $m \leq 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\dots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}...d_\jmath...d_0$, $m \leq 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the $\jmath^{\text{th}}$ shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0, 1\}$ [$\because d_\imath$ is BCD digit]

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1} b_{n-2} \dots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1} \dots d_\jmath \dots d_0$, $m \le 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0, 1\}$ [$\because$ $d_\imath$ is BCD digit]
- On a left shift, new BCD value $d'_\imath = 2y_\imath + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \ge 1$, otherwise the next input bit

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\ldots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}...d_\jmath...d_0$, $m \leq 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0, 1\}$ [$\because d_\imath$ is BCD digit]
- On a left shift, new BCD value $d'_\imath = 2y_\imath + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \geq 1$, otherwise the next input bit
- For the first three left shifts $D_\jmath = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\dots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}\dots d_\jmath\dots d_0$, $m \le 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0,1\}$ [$\because d_\imath$ is BCD digit]
- On a left shift, new BCD value $d'_\imath = 2y_\imath + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \ge 1$, otherwise the next input bit
- For the first three left shifts $D_\jmath = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$
- If the bits are exhausted, the conversion correctly terminates

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\dots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}...d_{\jmath}...d_0$, $m \leq 4\frac{n}{3}$
- Let $D_{\jmath}$ be the value of the BCD number after the $\jmath^{\text{th}}$ shift; $D_0 \equiv 0$
- Let $d_{\imath} \equiv m_{\imath} : y_{\imath}$, $m_{\imath}$ is MSB, $y_{\imath}$, the next 3-bits; initially, $d_{\imath} \equiv 0$
- If $m_{\imath} = 1$, $y_{\imath} \in \{0, 1\}$ [$\because d_{\imath}$ is BCD digit]
- On a left shift, new BCD value $d'_{\imath} = 2y_{\imath} + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \geq 1$, otherwise the next input bit
- For the first three left shifts $D_{\jmath} = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$
- If the bits are exhausted, the conversion correctly terminates
- Otherwise, any $y'_{\imath} \geq 5$ [$m'_{\imath} = 0$] is updated to $y'_{\imath} = 2y_{\imath} + m_{\imath-1} + 3$

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\ldots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}\ldots d_\jmath\ldots d_0$, $m \leq 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0, 1\}$ [$\because d_\imath$ is BCD digit]
- On a left shift, new BCD value $d'_\imath = 2y_\imath + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \geq 1$, otherwise the next input bit
- For the first three left shifts $D_\jmath = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$
- If the bits are exhausted, the conversion correctly terminates
- Otherwise, any $y'_\imath \geq 5$ $[m'_\imath = 0]$ is updated to $y'_\imath = 2y_\imath + m_{\imath-1} + 3$
- MSB of $d'_\jmath$ is the carry to be shifted into d'_{j+1}

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\ldots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}...d_{\jmath}...d_0$, $m \leq 4\frac{n}{3}$
- Let $D_{\jmath}$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_{\imath} \equiv m_{\imath} : y_{\imath}$, $m_{\imath}$ is MSB, $y_{\imath}$, the next 3-bits; initially, $d_{\imath} \equiv 0$
- If $m_{\imath} = 1$, $y_{\imath} \in \{0, 1\}$ [$\because$ $d_{\imath}$ is BCD digit]
- On a left shift, new BCD value $d'_{\imath} = 2y_{\imath} + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \geq 1$, otherwise the next input bit
- For the first three left shifts $D_{\jmath} = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$
- If the bits are exhausted, the conversion correctly terminates
- Otherwise, any $y'_{\imath} \geq 5$ [$m'_{\imath} = 0$] is updated to $y'_{\imath} = 2y_{\imath} + m_{\imath-1} + 3$
- MSB of $d'_{\jmath}$ is the carry to be shifted into d'_{j+1}
- On the next left shift, $D_{\jmath} = 2D_{j-1} + b_{n-j}$ again holds

Correctness of binary to BCD conversion

- Given binary value is $B = b_{n-1}b_{n-2}\dots b_0$, $n = 15$ for the example
- Let D be the BCD number with digits $d_{m-1}\dots d_\jmath \dots d_0$, $m \leq 4\frac{n}{3}$
- Let $D_\jmath$ be the value of the BCD number after the j^{th} shift; $D_0 \equiv 0$
- Let $d_\imath \equiv m_\imath : y_\imath$, $m_\imath$ is MSB, $y_\imath$, the next 3-bits; initially, $d_\imath \equiv 0$
- If $m_\imath = 1$, $y_\imath \in \{0, 1\}$ [$\because d_\imath$ is BCD digit]
- On a left shift, new BCD value $d'_\imath = 2y_\imath + m_{\imath-1}$ where $m_{\imath-1}$ is the MSB of $d_{\imath-1}$ if $\imath \geq 1$, otherwise the next input bit
- For the first three left shifts $D_\jmath = 2D_{\jmath-1} + b_{n-\jmath}$ holds

  $(D_1 = 2D_0 + b_{15}, D_2 = 2D_1 + b_{14}, D_3 = 2D_2 + b_{13})$
- If the bits are exhausted, the conversion correctly terminates
- Otherwise, any $y'_\imath \geq 5$ [$m'_\imath = 0$] is updated to $y'_\imath = 2y_\imath + m_{\imath-1} + 3$
- MSB of $d'_\jmath$ is the carry to be shifted into d'_{j+1}
- On the next left shift, $D_\jmath = 2D_{j-1} + b_{n-j}$ again holds
- Conversion algorithm is reversible

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft	0010	0100	0011	1010	0100	0000000000000000
Sub 3	0010	0100	0011	0111	0100	0000000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft	0010	0100	0011	1010	0100	0000000000000000
Sub 3	0010	0100	0011	0111	0100	0000000000000000
R Sft	0001	0010	0001	1011	1010	0000000000000000
Sub 3	0001	0010	0001	1000	0111	0000000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft Sub 3	0010 0010	0100 0100	0011 0011	1010 0111	0100 0100	0000000000000000 0000000000000000
R Sft Sub 3	0001 0001	0010 0010	0001 0001	1011 1000	1010 0111	0000000000000000 0000000000000000
R Sft Sub 3	0000 0000	1001 0110	0000 0000	1100 1001	0011 0011	1000000000000000 1000000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft Sub 3	0010 0010	0100 0100	0011 0011	1010 0111	0100 0100	0000000000000000 0000000000000000
R Sft Sub 3	0001 0001	0010 0010	0001 0001	1011 1000	1010 0111	0000000000000000 0000000000000000
R Sft Sub 3	0000 0000	1001 0110	0000 0000	1100 1001	0011 0011	1000000000000000 1000000000000000
R Sft Sub 3	0000 0000	0011 0011	0000 0000	0100 0100	1001 0110	1100000000000000 1100000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft	0010	0100	0011	1010	0100	0000000000000000
Sub 3	0010	0100	0011	0111	0100	0000000000000000
R Sft	0001	0010	0001	1011	1010	0000000000000000
Sub 3	0001	0010	0001	1000	0111	0000000000000000
R Sft	0000	1001	0000	1100	0011	1000000000000000
Sub 3	0000	0110	0000	1001	0011	1000000000000000
R Sft	0000	0011	0000	0100	1001	1100000000000000
Sub 3	0000	0011	0000	0100	0110	1100000000000000
R Sft	0000	0001	1000	0010	0011	0110000000000000
Sub 3	0000	0001	0101	0010	0011	0110000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft	0010	0100	0011	1010	0100	0000000000000000
Sub 3	0010	0100	0011	0111	0100	0000000000000000
R Sft	0001	0010	0001	1011	1010	0000000000000000
Sub 3	0001	0010	0001	1000	0111	0000000000000000
R Sft	0000	1001	0000	1100	0011	1000000000000000
Sub 3	0000	0110	0000	1001	0011	1000000000000000
R Sft	0000	0011	0000	0100	1001	1100000000000000
Sub 3	0000	0011	0000	0100	0110	1100000000000000
R Sft	0000	0001	1000	0010	0011	0110000000000000
Sub 3	0000	0001	0101	0010	0011	0110000000000000
R Sft	0000	0000	1010	1001	0001	1011000000000000
Sub 3	0000	0000	0111	0110	0001	1011000000000000

BCD 48748 to Binary example

Op	B4	B3	B2	B1	B0	
Input	0100	1000	0111	0100	1000	
R Sft	0010	0100	0011	1010	0100	0000000000000000
Sub 3	0010	0100	0011	0111	0100	0000000000000000
R Sft	0001	0010	0001	1011	1010	0000000000000000
Sub 3	0001	0010	0001	1000	0111	0000000000000000
R Sft	0000	1001	0000	1100	0011	1000000000000000
Sub 3	0000	0110	0000	1001	0011	1000000000000000
R Sft	0000	0011	0000	0100	1001	1100000000000000
Sub 3	0000	0011	0000	0100	0110	1100000000000000
R Sft	0000	0001	1000	0010	0011	0110000000000000
Sub 3	0000	0001	0101	0010	0011	0110000000000000
R Sft	0000	0000	1010	1001	0001	1011000000000000
Sub 3	0000	0000	0111	0110	0001	1011000000000000
R Sft	0000	0000	0011	1011	0000	1101100000000000
Sub 3	0000	0000	0011	1000	0000	1101100000000000

BCD 48748 to Binary example

R Sft	0000	0000	0001	1100	0000	0110110000000000
Sub 3	0000	0000	0001	1001	0000	0110110000000000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000

BCD 48748 to Binary example

R Sft	0000	0000	0001	1100	0000	0110110000000000
Sub 3	0000	0000	0001	1001	0000	0110110000000000
R Sft	0000	0000	0000	1100	1000	0011011000000000
Sub 3	0000	0000	0000	1001	0101	0011011000000000
R Sft	0000	0000	0000	0100	1010	1001101100000000
Sub 3	0000	0000	0000	0100	0111	1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000
R Sft	0000	0000	0000	0001	0001	1110011011000000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000
R Sft	0000	0000	0000	0001	0001	1110011011000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0000 0000	1000 0101	1111001101100000 1111001101100000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	01101100000000 01101100000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	00110110000000 00110110000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	10011011000000 10011011000000
R Sft	0000	0000	0000	0010	0011	11001101100000
R Sft	0000	0000	0000	0001	0001	11100110110000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0000 0000	1000 0101	11110011011000 11110011011000
R Sft	0000	0000	0000	0000	0010	11111001101100

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000
R Sft	0000	0000	0000	0001	0001	1110011011000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0000 0000	1000 0101	1111001101100000 1111001101100000
R Sft	0000	0000	0000	0000	0010	1111100110110000
R Sft	0000	0000	0000	0000	0001	0111110011011000

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000
R Sft	0000	0000	0000	0001	0001	1110011011000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0000 0000	1000 0101	1111001101100000 1111001101100000
R Sft	0000	0000	0000	0000	0010	1111100110110000
R Sft	0000	0000	0000	0000	0001	0111110011011000
R Sft	0000	0000	0000	0000	0000	1011111001101100

BCD 48748 to Binary example

R Sft Sub 3	0000 0000	0000 0000	0001 0001	1100 1001	0000 0000	0110110000000000 0110110000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	1100 1001	1000 0101	0011011000000000 0011011000000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0100 0100	1010 0111	1001101100000000 1001101100000000
R Sft	0000	0000	0000	0010	0011	1100110110000000
R Sft	0000	0000	0000	0001	0001	1110011011000000
R Sft Sub 3	0000 0000	0000 0000	0000 0000	0000 0000	1000 0101	1111001101100000 1111001101100000
R Sft	0000	0000	0000	0000	0010	1111100110110000
R Sft	0000	0000	0000	0000	0001	0111110011011000
R Sft	0000	0000	0000	0000	0000	1011111001101100
End						48748

Binary codes

- Binary coding scheme for decimal digits
- Sequence of bits $x_3x_2x_1x_0$ (say) for N is it's code word
- Each position i may have a weight w_i (**weighted code**); $N = \sum w_i x_i$
- For BCD $w_3 = 8, w_2 = 4, w_1 = 2, w_0 = 1$
- Sum of weights is 9 for self-complementing code

		weights													
N		8	4	2	1		2	4	2	1		6	4	2	-3
0		0	0	0	0		0	0	0	0		0	0	0	0
1		0	0	0	1		0	0	0	1		0	1	0	1
2		0	0	1	0		0	0	1	0		0	0	1	0
3		0	0	1	1		0	0	1	1		1	0	0	1
4		0	1	0	0		0	1	0	0		0	1	0	0
5		0	1	0	1		1	0	0	1		1	0	1	1
6		0	1	1	0		1	1	0	0		0	1	1	0
7		0	1	1	1		1	1	0	1		1	1	0	0
8		1	0	0	0		1	1	1	0		1	0	1	0
9		1	0	0	1		1	1	1	1		1	1	1	1

Binary codes

BCD				Excess-3				Cyclic				Gray			
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1

- Excess-3, Cyclic and Gray codes are **unweighted** codes
- Excess-3 code is formed by adding 3 (0011) to the BCD value
- It's is self-complementing

Binary codes

BCD				Excess-3				Cyclic				Gray			
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1

- Excess-3, Cyclic and Gray codes are **unweighted** codes
- Excess-3 code is formed by adding 3 (0011) to the BCD value
- It's is self-complementing (n+3 $+$ (9-n)+3 $=$ 15)

Linear codes

Definition (Linear code)

A linear $[n, k]$ code (of length n and rank k) is a linear subspace C with dimension k of the vector space $\mathbb{F}_q^n$; often $|\mathbb{F}_q| = q = 2$ and $\mathbb{F}_2 = \{0, 1\}$ when $\mathbb{F}_2^n$ is a word of n bits

For such a linear code:

- The zero vector is always a codeword
- The sum or difference of two codewords is another codeword (the modulo 2 sum is the bitwise $\oplus$ operation)
- The number of codewords in an linear $[n, k]$ code C over $\mathbb{F}_q^n$ is q^k

Example (Parity check codes)

- $C = \{p, b_{n-1}, \ldots, b_0 | p \oplus b_n - 1 \oplus \ldots \oplus b_0 = 0\}$
- Now $c_1, c_2 \in C \Rightarrow c_1 + c_2 \in C$
- Thus, C is a $[n+1, n]$ linear code

Cyclic codes

Definition (Cyclic code)

A binary code C is cyclic if it is a linear $[n, k]$ code and if for every codeword $\langle c_{n-1}, \ldots, c_1, c_0\rangle \in C$, $\langle c_{n-2}, \ldots, c_0, c_{n-1}\rangle$ is also a codeword in C

Let $\mathbb{F}_2[x]$ denote all: $a_m x^m + \ldots + a_1 x + a_0$ with $a_m, \ldots, a_1, a_0 \in \mathbb{F}_2$

Definition (Code polynomial associated with a codeword)

Let $a = \langle a_{n-1}, \ldots, a_1, a_0\rangle$ be a codeword, then
$a(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ is the corresponding code polynomial

Example (Parity check codes)

- $c_1' + c_2'$ after the same rotation satisfies parity check
- $c_1' + c_2'$ after different rotations also satisfies parity check
- Thus, parity check codes are also cyclic

Cyclic codes (contd.)

Realising cyclic code via code polynomials

- $x \cdot a(x) \equiv a_{n-2}x^{n-1} + \ldots + a_1 x^2 + a_0 x + a_{n-1} \mod (x^n - 1)$
- $x^2 \cdot a(x) \equiv a_{n-3}x^{n-1} + \ldots + a_0 x^2 + a_{n-1} x + a_{n-2} \mod (x^n - 1)$
- $x^\ell \cdot a(x) \equiv a_{n-\ell-1}x^{n-1} + \ldots + a_{n-\ell+1} x + a_{n-\ell} \mod (x^n - 1)$
- By the cyclic property each of the above is a valid code word
- Thus, any sum of these is also a valid code polynomial by the linearity property
- The sum could be expressed as: $(g_\ell x^\ell + g_1 x + g_0)\, a(x) \mod (x^n - 1)$
- Thus, for $g(x) \in \mathbb{F}[x]$, $g(x)a(x) \mod (x^n - 1)$ is a valid code polynomial

Cyclic codes (contd.)

Theorem

Let $n > 1$ and let $g(x) \in \mathbb{F}_2[x]$ divide the polynomial $x^n - 1$. Assume $\deg(g(x)) = n - k$ for some $0 \le k \le n$. The polynomials $\mathcal{P}_g = \{f(x) = g(x) \cdot \alpha(x) \mod (x^n - 1) | \alpha(x) \in \mathbb{F}_2[x] \wedge \deg(\alpha(x)) < k\}$ give rise to a cyclic [n, k] code.

Proof.

- Let C be the code corresponding to all $f_\imath(x) = g(x) \cdot \alpha_\imath(x) \in \mathcal{P}_g$
- Rank of the space is k
- Now, $f_1(x), f_2(x) \in \mathcal{P}_g \Rightarrow f_1(x) + f_2(x) = g(x)(\alpha_1(x) + \alpha_2(x)) \in \mathcal{P}_g$
- Let $f(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \in \mathcal{P}_g$,
- $x \cdot f(x) = \underbrace{a_{n-2}x^{n-1} + \ldots + a_0x + a_{n-1}}_{h(x)} + a_{n-1}(x^n - 1)$
- $g(x)$ divides $f(x)$ and $x^n - 1$, so it also divides $h(x)$
- Hence it's a cyclic $[n, k]$ code

□

Cyclic codes (contd.)

Theorem

Let C be a cyclic code. Then there exists a uniquely determined code (generator) polynomial $g(x)$ of minimal degree in C which has the following properties: $g(x)$ is unique, $g(x)$ divides $x^n - 1$ and the code C can be constructed using $g(x)$

Proof.

- Let $g_1(x), g_2(x)$ be distinct code polys of minimal degree, then $g(x) = g_1(x) - g_2(x)$ is a code poly of lesser degree
- If $g(x)$ doesn't divide $x^n - 1$, then $x^n - 1 = \underbrace{g(x)\beta(x)}_{\text{code poly}} + r(x) \equiv 0$

  So, $r(x)$ is also code poly by $\deg(r(x)) < \deg(g(x))$
- With such a $g(x)$ available, the code can be generated as $\mathcal{P}_g = \{g(x) \cdot \alpha(x) \mod (x^n - 1) | \alpha(x) \in \mathbb{F}_2[x] \wedge \deg(\alpha(x)) < k\}$ □

Binary codes (contd.)

BCD				Excess-3				Cyclic				Gray			
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1

- Gray code is cyclic (in the range 0..15, 0 and 15 also a reflected code – not cyclic in 0..9

Binary codes (contd.)

BCD				Excess-3				Cyclic				Gray			
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1

- Gray code is cyclic (in the range 0..15, 0 and 15 also a reflected code – not cyclic in 0..9
- $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$; $b_i = ?$

Binary codes (contd.)

BCD				Excess-3				Cyclic				Gray			
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1
0	0	1	0	0	1	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	1	1	0	1	1	0	0	1	1	0
0	1	0	1	1	0	0	0	1	1	1	0	0	1	1	1
0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
1	0	0	1	1	1	0	0	0	1	0	0	1	1	0	1

- Gray code is cyclic (in the range 0..15, 0 and 15 also a reflected code – not cyclic in 0..9
- $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$; $b_i = ?$
- $g_i \oplus b_{i+1} = b_i \oplus b_{i+1} \oplus b_{i+1} = b_i \oplus 0 = b_i$
- Are any of these codes linear or cyclic?

Binary codes (contd.)

N	Binary				Gray			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

- $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$
- n and it's bitwise complement $\tilde{n}$ are placed symmetrically about the middle of the table

Binary codes (contd.)

N	Binary				Gray			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

- $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$
- n and it's bitwise complement $\tilde{n}$ are placed symmetrically about the middle of the table
- Their Gray codes should differ only in the MSB
- Let $n \equiv b_{n-1}b_{n-2}\ldots b_0$ and it's Gray code be $g_{n-1}g_{n-2}\ldots g_0$
- By the rule the gray code of $\tilde{n}$ is

Binary codes (contd.)

N	Binary				Gray			
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

- $g_{n-1} = b_{n-1}$, $g_i = b_i \oplus b_{i+1}$
- n and it's bitwise complement $\tilde{n}$ are placed symmetrically about the middle of the table
- Their Gray codes should differ only in the MSB
- Let $n \equiv b_{n-1} b_{n-2} \ldots b_0$ and it's Gray code be $g_{n-1} g_{n-2} \ldots g_0$
- By the rule the gray code of $\tilde{n}$ is

$\overline{b_{n-1}}$	$\overline{b_{n-2}}$	$\ldots$	$\overline{b_0}$
0	$\overline{b_{n-1}}$	$\ldots$	$\overline{b_1}$
$\overline{b_{n-1}}$	$b_{n-2} \oplus b_{n-1}$	$\ldots$	$b_0 \oplus b_1$
$\overline{g_{n-1}}$	g_{n-2}	$\ldots$	g_0

- Thus the Gray codes of n and $\tilde{n}$ differ only in the MSB

Binary codes (contd.)

Is the Gray code weighted?

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only
- $\forall i \exists j | (j+1) - j = \sum_i w_i \left(x_{i,j+1} - x_{i,j} \right) = \pm w_i = 1$ (why?)

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only
- $\forall i \exists j | (j+1) - j = \sum_i w_i \left(x_{i,j+1} - x_{i,j} \right) = \pm w_i = 1$ (why?)
- This precludes representation of 2^n values for a n-bit Gray code

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only
- $\forall i \exists j | (j+1) - j = \sum_i w_i \left(x_{i,j+1} - x_{i,j}\right) = \pm w_i = 1$ (why?)
- This precludes representation of 2^n values for a n-bit Gray code

Is the Excess-3 code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only
- $\forall i \exists j | (j+1) - j = \sum_i w_i \left(x_{i,j+1} - x_{i,j} \right) = \pm w_i = 1$ (why?)
- This precludes representation of 2^n values for a n-bit Gray code

Is the Excess-3 code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- $w_2 = 1$ $[1 \mapsto 4\ (0100)]$
- $w_3 = 5$ $[5 \mapsto 8\ (1000)]$
- $w_1 + w_0 = 0$ $[0 \mapsto 3\ (0011)]$

Binary codes (contd.)

Is the Gray code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- Suppose it's weighted
- Utilise the property that adjacent codes differ in one place only
- $\forall i \exists j | (j+1) - j = \sum_i w_i \left(x_{i,j+1} - x_{i,j} \right) = \pm w_i = 1$ (why?)
- This precludes representation of 2^n values for a n-bit Gray code

Is the Excess-3 code weighted?

- Can we find weights such that $\sum_i w_i x_{i,j} = j$?
- $w_2 = 1$ $[1 \mapsto 4\ (0100)]$
- $w_3 = 5$ $[5 \mapsto 8\ (1000)]$
- $w_1 + w_0 = 0$ $[0 \mapsto 3\ (0011)]$
- But, $w_2 + w_1 + w_0 = 5 \neq 4$ $[4 \mapsto 7\ (0111)]$ – inconsistent

Excess-3 arithmetic

Example (Excess-3 addition)

- $825 + 528 = 1353$
- Excess-3

	0	0	1	1	1	0	1	1	0	1	0	1	1	0	0	0
+	0	0	1	1	1	0	0	0	0	1	0	1	1	0	1	1
	0	1	1	1	10	0	1	1	1	0	1	1	10	0	1	1
	0	1	0	0	0	1	1	0	1	0	0	0	0	1	1	0

Example (Excess-3 subtraction)

- $825 - 528 = 297 \rightarrow 825 + 471 + 1 = 1297 = 297 \mod 1000$
- Excess-3

	0	0	1	1	1	0	1	1	0	1	0	1	1	0	0	0
+	0	0	1	1	0	1	1	1	1	0	1	0	0	1	0	10
	0	1	1	1	10	0	1	0	1	1	1	1	1	1	0	1
	0	1	0	0	0	1	0	1	1	1	0	0	1	0	1	0

Error detecting code

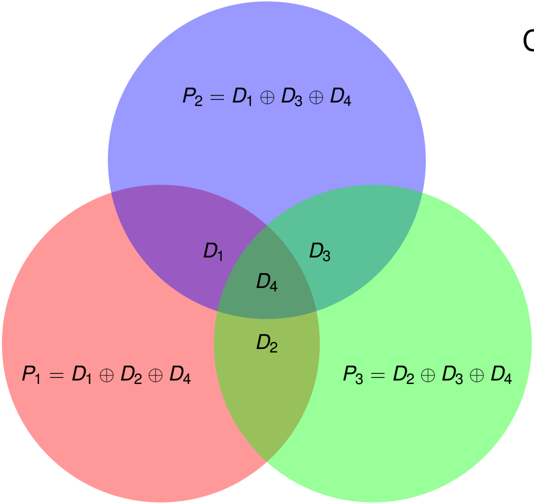
N		Even Parity BCD						2-out-of-5, $\binom{5}{2} = 10$						63210 BCD				
		8	4	2	1	p		0	1	2	4	7		6	3	2	1	0
0		0	0	0	0	0		0	0	0	1	1		0	0	1	1	0
1		0	0	0	1	1		1	1	0	0	0		0	0	0	1	1
2		0	0	1	0	1		1	0	1	0	0		0	0	1	0	1
3		0	0	1	1	0		0	1	1	0	0		0	1	0	0	1
4		0	1	0	0	1		1	0	0	1	0		0	1	0	1	0
5		0	1	0	1	0		0	1	0	1	0		0	1	1	0	0
6		0	1	1	0	0		0	0	1	1	0		1	0	0	0	1
7		0	1	1	1	1		1	0	0	0	1		1	0	0	1	0
8		1	0	0	0	1		0	1	0	0	1		1	0	1	0	0
9		1	0	0	1	0		0	0	1	0	1		1	1	0	0	0

Error detecting code

N		Even Parity BCD						2-out-of-5, $\binom{5}{2}=10$						63210 BCD				
		8	4	2	1	p		0	1	2	4	7		6	3	2	1	0
0		0	0	0	0	0		0	0	0	1	1		0	0	1	1	0
1		0	0	0	1	1		1	1	0	0	0		0	0	0	1	1
2		0	0	1	0	1		1	0	1	0	0		0	0	1	0	1
3		0	0	1	1	0		0	1	1	0	0		0	1	0	0	1
4		0	1	0	0	1		1	0	0	1	0		0	1	0	1	0
5		0	1	0	1	0		0	1	0	1	0		0	1	1	0	0
6		0	1	1	0	0		0	0	1	1	0		1	0	0	0	1
7		0	1	1	1	1		1	0	0	0	1		1	0	0	1	0
8		1	0	0	0	1		0	1	0	0	1		1	0	1	0	0
9		1	0	0	1	0		0	0	1	0	1		1	1	0	0	0

- Hamming distance: number of bits differing between two codes
- If minimum Hamming distance between any two code words is d then $d-1$ single bit errors can be detected

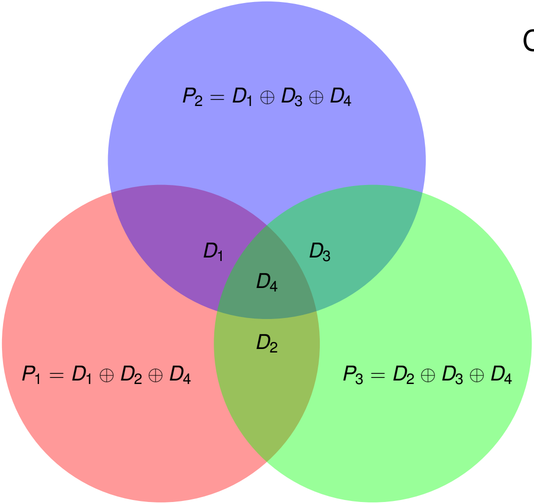
Error correcting code



Correction for single bit error

D_1 P_1 and P_2 affected, P_3 unaffected

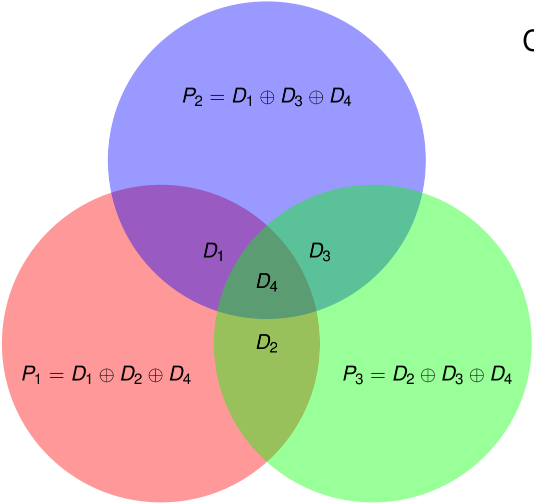
Error correcting code



Correction for single bit error

- D_1 P_1 and P_2 affected, P_3 unaffected
- D_2 P_1 and P_3 affected, P_2 unaffected

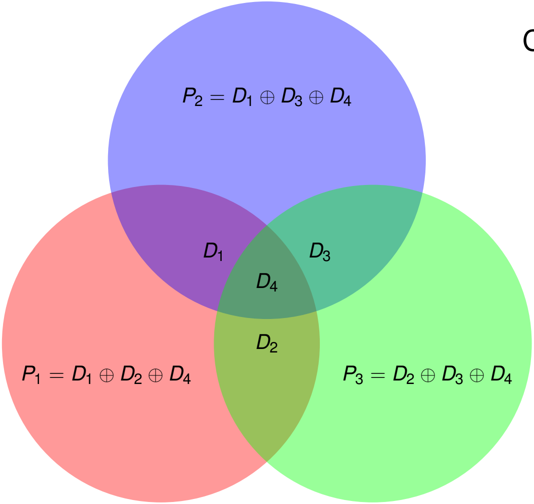
Error correcting code



Correction for single bit error

- D_1 P_1 and P_2 affected, P_3 unaffected
- D_2 P_1 and P_3 affected, P_2 unaffected
- D_3 P_2 and P_3 affected, P_1 unaffected

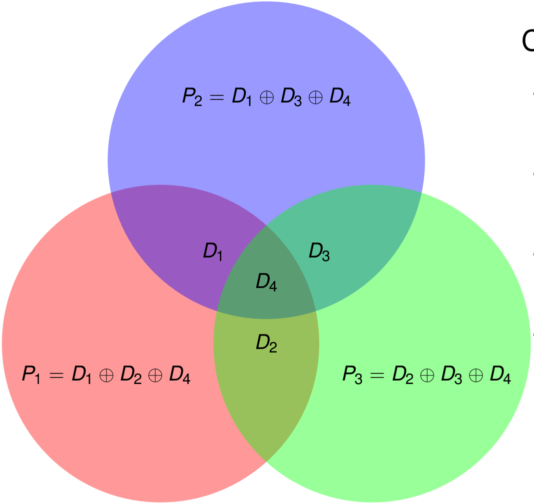
Error correcting code



Correction for single bit error

- D_1 P_1 and P_2 affected, P_3 unaffected
- D_2 P_1 and P_3 affected, P_2 unaffected
- D_3 P_2 and P_3 affected, P_1 unaffected
- D_4 P_1, P_2 and P_2 affected

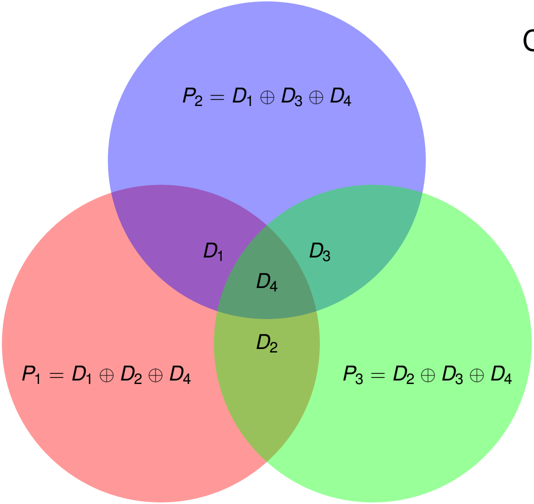
Error correcting code



Correction for single bit error

- D_1 — P_1 and P_2 affected, P_3 unaffected
- D_2 — P_1 and P_3 affected, P_2 unaffected
- D_3 — P_2 and P_3 affected, P_1 unaffected
- D_4 — P_1, P_2 and P_2 affected
- P_1 — Nothing else affected

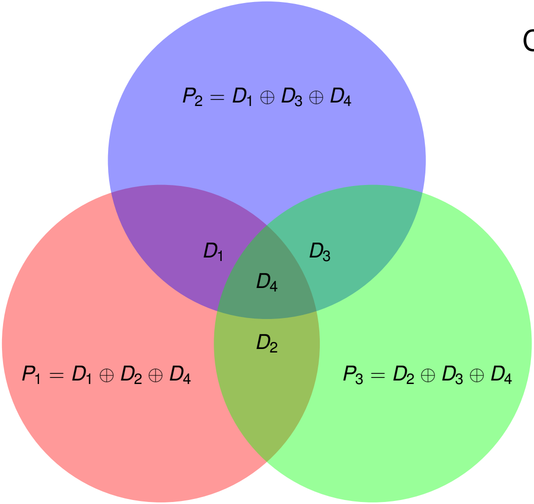
Error correcting code



Correction for single bit error

- D_1 P_1 and P_2 affected, P_3 unaffected
- D_2 P_1 and P_3 affected, P_2 unaffected
- D_3 P_2 and P_3 affected, P_1 unaffected
- D_4 P_1, P_2 and P_2 affected
- P_1 Nothing else affected
- P_2 Nothing else affected

Error correcting code



Correction for single bit error

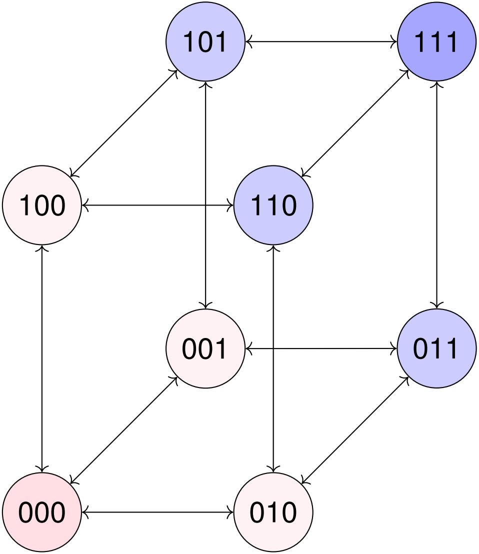
- D_1 P_1 and P_2 affected, P_3 unaffected
- D_2 P_1 and P_3 affected, P_2 unaffected
- D_3 P_2 and P_3 affected, P_1 unaffected
- D_4 P_1, P_2 and P_2 affected
- P_1 Nothing else affected
- P_2 Nothing else affected
- P_3 Nothing else affected

Relating data and parity bits

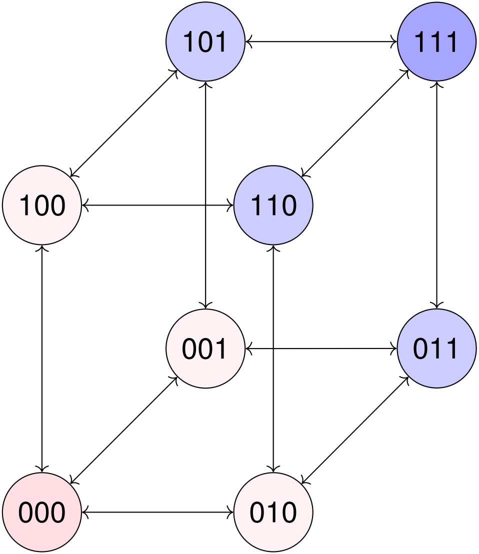
- Association of parity bits to the data bits may be done according to the table below

Bits indices	7	6	5	4	3	2	1
Binary	111	110	101	100	011	010	001
Data/parity	d_4	d_3	d_2	p_3	d_1	p_2	p_1
Association	p_3, p_2, p_1	p_3, p_2	p_3, p_1	p_3	p_2, p_1	p_2	p_1

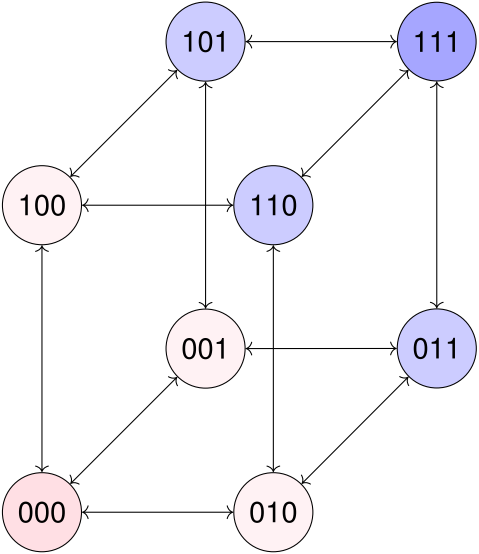
- Bit at 2^i positions (1, 2, 4) are for parity, others for data
- p_1 covers data bit positions having 1 in LSB (1:p_1, 3:d_1, 5:d_2, 7:d_4)
- p_2 covers data bit positions having 1 in next higher bit position (2:p_2, 3:d_1, 6:d_3, 7:d_4)
- p_3 covers data bit positions having 1 in next higher bit position (4:p_3, 5:d_2, 6:d_3, 7:d_4)
- This scheme may be generalised



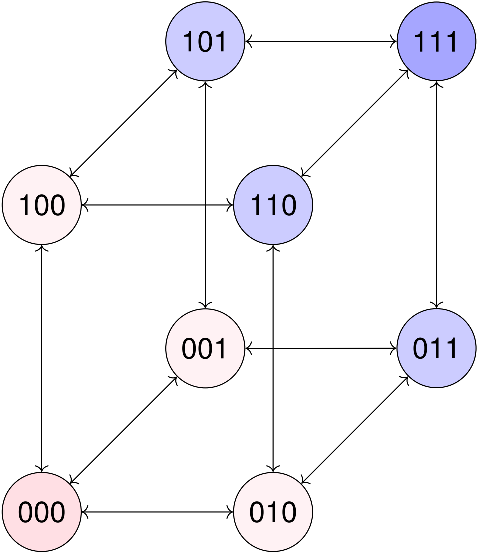
- Consider codes 000 and 111 and all possible single bit errors



- Consider codes 000 and 111 and all possible single bit errors
- Any single bit error code can be traced backed to 000 or 111



- Consider codes 000 and 111 and all possible single bit errors
- Any single bit error code can be traced backed to 000 or 111
- Achive by maintaining Hamming distance of 3 between the code words



- Consider codes 000 and 111 and all possible single bit errors
- Any single bit error code can be traced backed to 000 or 111
- Achive by maintaining Hamming distance of 3 between the code words
- If d is the minimum Hamming distance between code words, up to $\lfloor \frac{d-1}{2} \rfloor$-bit errors can be corrected

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code
- $(n + 1)2^m \leq 2^n$

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code
- $(n + 1)2^m \leq 2^n$
- $n + 1 \leq 2^{n-m} = 2^p$

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code
- $(n + 1)2^m \leq 2^n$
- $n + 1 \leq 2^{n-m} = 2^p$
- $m + p + 1 \leq 2^p$

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code
- $(n + 1)2^m \leq 2^n$
- $n + 1 \leq 2^{n-m} = 2^p$
- $m + p + 1 \leq 2^p$
- For $m = 4$ $p =?$

Mininum bits for 1-bit ECC

- Let there be m information bits in total of n bits; $m + p = n$
- n patterns for 1-bit error in a code word; 1 valid pattern
- Reserve $n + 1$ patterns for each code
- $(n + 1)2^m \leq 2^n$
- $n + 1 \leq 2^{n-m} = 2^p$
- $m + p + 1 \leq 2^p$
- For $m = 4$ $p =?$
- Say $p = 3$ then $2^p = 2^3 = 8 \geq 4 + 3 + 1 = 8$

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error
- Since there is no recovery errorneous codes can be "shared" between valid codes

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error
- Since there is no recovery errorneous codes can be "shared" between valid codes
- Adjacent codes must have separate colours (valid:✓, error:✗)

	000	001	011	010	110	111	101	100
00	✓	✗	✓	✗	✓	✗	✓	✗
01	✗	✓	✗	✓	✗	✓	✗	✓
11	✓	✗	✓	✗	✓	✗	✓	✗
10	✗	✓	✗	✓	✗	✓	✗	✓

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error
- Since there is no recovery errorneous codes can be "shared" between valid codes
- Adjacent codes must have separate colours (valid:✓, error:✗)

	000	001	011	010	110	111	101	100
00	✓	✗	✓	✗	✓	✗	✓	✗
01	✗	✓	✗	✓	✗	✓	✗	✓
11	✓	✗	✓	✗	✓	✗	✓	✗
10	✗	✓	✗	✓	✗	✓	✗	✓

- For single bit error, at most half the codes are usable

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error
- Since there is no recovery errorneous codes can be "shared" between valid codes
- Adjacent codes must have separate colours (valid:✓, error:✗)

	000	001	011	010	110	111	101	100
00	✓	✗	✓	✗	✓	✗	✓	✗
01	✗	✓	✗	✓	✗	✓	✗	✓
11	✓	✗	✓	✗	✓	✗	✓	✗
10	✗	✓	✗	✓	✗	✓	✗	✓

- For single bit error, at most half the codes are usable
- For m bits of data, $n = m + 1$ bits are needed for EDC

Mininum bits for 1-bit EDC

- For single bit error, all codes at Hamming distance of 1 from a valid code are in error
- Since there is no recovery errorneous codes can be “shared” between valid codes
- Adjacent codes must have separate colours (valid:✓, error:✗)

	000	001	011	010	110	111	101	100
00	✓	✗	✓	✗	✓	✗	✓	✗
01	✗	✓	✗	✓	✗	✓	✗	✓
11	✓	✗	✓	✗	✓	✗	✓	✗
10	✗	✓	✗	✓	✗	✓	✗	✓

- For single bit error, at most half the codes are usable
- For m bits of data, $n = m + 1$ bits are needed for EDC
- BCD with error detection cannot be accommodated in 4-bits