Digital systems: from bits to microcontrollers

- Introduction to digital systems fundamentals
- Combinatorial digital systems
- Sequential digital systems
- Introduction to microcontrollers
- Introduction to advanced implementation platforms

Number systems

- positional notation,
- decimal representation,
- binary representation,
- octal representation, and
- hexadecimal representation;
- conversions between bases

Positional notation

$$A = (\underbrace{a_{n-1}a_{n-2}\ldots a_1 a_0}_{\text{integer part}} \,.\, \underbrace{a_{-1}a_{-2}\ldots a_{-m}}_{\text{fractional part}})_r$$

radix

most significant digit

least significant digit

Polynomial notation

$$A = (a_{n-1}a_{n-2}\ldots a_1 a_0 \,.\, a_{-1}a_{-2}\ldots a_{-m})_r$$

$$A = a_{n-1}.r^{n-1} + a_{n-2}.r^{n-2} + \ldots + a_1.r^1 + a_0.r^0 + a_{-1}.r^{-1} + a_{-2}.r^{-2} + \ldots + a_{-m}.r^{-m}$$

$$A = \sum_{i=-m}^{i=n-1} a_i.r^i$$

DECIMAL
- Radix: 10 Symbols: 0,1,2,3,4,5,6,7,8,9
- 1357 $\Rightarrow$ $1x10^3 + 3x10^2 + 5x10^1 + 7x10^0$

OCTAL
- Radix: 8 Symbols: 0,1,2,3,4,5,6,7
- 2615 $\Rightarrow$ $2x8^3 + 6x8^2 + 1x8^1 + 5x8^0$

HEXADECIMAL
- Radix: 16 Symbols: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- A05F $\Rightarrow$ $10x16^3 + 0x16^2 + 5x16^1 + 15x16^0$

BINARY
- Radix: 2 Symbols: 0,1
- 1011 $\Rightarrow$ $1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

Conversion from decimal to binary: integer part

```
23 |_2_
 1  11 |_2_
     1  5 |_2_
        1  2 |_2_
           0  1
```

LSB (least significant bit)

MSB (most significant bit)

$$23_{10} = 10111_2$$

Conversion from decimal to binary: fractional part

$$\begin{array}{r} 0.375 \\ 2 \\ \hline 0.750 \\ 2 \\ \hline 1.50 \\ 2 \\ \hline 1.00 \end{array}$$

$$0.375_{10} = 0.011_2$$

Conversion from decimal to binary

integer part

$$23_{10} = 10111_2$$

$$0.375_{10} = 0.011_2$$

fractional part

$$23.375_{10} = 10111.011_2$$

Conversion from binary to decimal: integer part

$10111_2 = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0 =$

$= 1x16 + 0x8 + 1x4 + 1x2 + 1x1 =$

$= 16 + 4 + 2 + 1 =$

$= 23$

$10111_2 = 23_{10}$

Conversion from binary to decimal: integer part

$10111_2 = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0 =$

$= 16 + 4 + 2 + 1 =$

$= 23$

$10111_2 = 23_{10}$

Conversion from binary to decimal: fractional part

$0.101_2 = 1x2^{-1} + 0x2^{-2} + 1x2^{-3} =$

$= 1x0.5 + 0x0.25 + 1x0.125 =$

$= 0.5 + 0.125 =$

$= 0.625$

$0.101_2 = 0.625_{10}$

Conversion from binary to decimal: fractional part

$0.101_2 = 1x2^{-1} + 0x2^{-2} + 1x2^{-3} =$

$= 0.5 + 0.125 =$

$= 0.625$

$0.101_2 = 0.625_{10}$

Conversion from binary to decimal

$$10111.101_2 = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} =$$

$$= 16 + 4 + 2 + 1 + 0.5 + 0.125 =$$

$$= 23.625$$

$$10111.101_2 = 23.625_{10}$$

Converting binary to/from octal

Power of two radix can be easily converted to/from binary!

Octal → 1 octal digit = 3 binary digit

$$\overbrace{10}^{2}\overbrace{111}^{7}.\overbrace{101}^{5}{}_2 = 27.5_8$$

Binary	Octal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Converting binary to/from hexadecimal

Hexadecimal → 1 hexadecimal digit = 4 binary digit

$$\overbrace{1}^{1}\overbrace{0111}^{7}.\overbrace{101}^{A}{}_2 = 17.A_{16}$$

Binary	Hex	Binary	Hex
0 0 0 0	0	1 0 0 0	8
0 0 0 1	1	1 0 0 1	9
0 0 1 0	2	1 0 1 0	A
0 0 1 1	3	1 0 1 1	B
0 1 0 0	4	1 1 0 0	C
0 1 0 1	5	1 1 0 1	D
0 1 1 0	6	1 1 1 0	E
0 1 1 1	7	1 1 1 1	F

Converting numbers

$$10111.101_2 = 27.5_8 = 23.625_{10} = 17.A_{16}$$

Higher radix → Lower number of digits (compact representation)

Counting in different number systems

Decimal	Binary	Octal	Hex
0	0	0	0
1	1	1	1
2	1 0	2	2
3	1 1	3	3
4	1 0 0	4	4
5	1 0 1	5	5
6	1 1 0	6	6
7	1 1 1	7	7

Decimal	Binary	Octal	Hex
8	1 0 0 0	10	8
9	1 0 0 1	11	9
10	1 0 1 0	12	A
11	1 0 1 1	13	B
12	1 1 0 0	14	C
13	1 1 0 1	15	D
14	1 1 1 0	16	E
15	1 1 1 1	17	F
16	1 0 0 0 0	20	10

Major goal for next steps

Illustrate how modular composition/decomposition is important in system design to suport top-down and bottom-up approaches.

Different types of combinatorial modules will be used to illustrate the concept.

Potential impact on several aspects will be analysed (ranging from costs and power consumption to performance)

Data representation

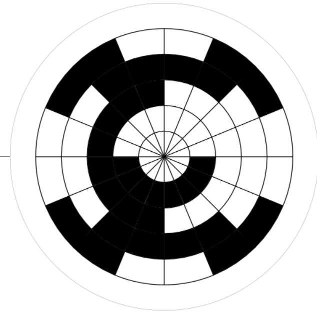
Numerical data
Alphanumeric data (example: ASCII)
Different types of codes and coding strategies

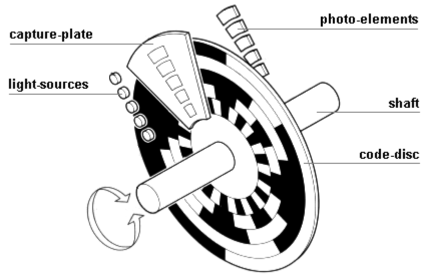
Natural Binary Code (using 4 bits)

Decimal Value	Natural Binary	Decimal Value	Natural Binary
0	0 0 0 0	8	1 0 0 0
1	0 0 0 1	9	1 0 0 1
2	0 0 1 0	10	1 0 1 0
3	0 0 1 1	11	1 0 1 1
4	0 1 0 0	12	1 1 0 0
5	0 1 0 1	13	1 1 0 1
6	0 1 1 0	14	1 1 1 0
7	0 1 1 1	15	1 1 1 1

Gray Code (using 3 bits)

Value	Natural Binary	Gray	Decimal Value
0	0 0 0	0 0 0	0
1	0 0 1	0 0 1	1
2	0 1 0	0 1 1	3
3	0 1 1	0 1 0	2
4	1 0 0	1 1 0	6
5	1 0 1	1 1 1	7
6	1 1 0	1 0 1	5
7	1 1 1	1 0 0	4





Binary decimal codes

Maybe the most well known binary decimal codes are:

- BCD8421
- 2 out-of 5

2-out-of-5 1001001100100010010 1

2015

BCD8421 0010000000010101

BCD 8421		2 out-of 5 P ‘7’ ’4’’2’’1’
0 0 0 0	0	0 1 1 0 0
0 0 0 1	1	1 0 0 0 1
0 0 1 0	2	1 0 0 1 0
0 0 1 1	3	0 0 0 1 1
0 1 0 0	4	1 0 1 0 0
0 1 0 1	5	0 0 1 0 1
0 1 1 0	6	0 0 1 1 0
0 1 1 1	7	1 1 0 0 0
1 0 0 0	8	0 1 0 0 1
1 0 0 1	9	0 1 0 1 0

Decoders

An n-to-2^n decoder is a combinatorial circuit with n input lines and 2^n output signals, which one decoding one minterm of the input lines.

Example of a 2-to-4 decoder:

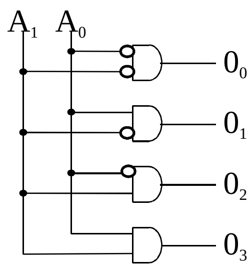
A_1	A_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$O_0 = f_0(A_1,A_0) = m_0 = /A_1./A_0$

$O_1 = f_1(A_1,A_0) = m_1 = /A_1.A_0$

$O_2 = f_2(A_1,A_0) = m_2 = A_1./A_0$

$O_3 = f_3(A_1,A_0) = m_3 = A_1.A_0$



Decoders

Example of a 3-to-8 decoder:

A_2	A_1	A_0	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$O_0 = f_0(A_2,A_1,A_0) = m_0 = /A_2./A_1./A_0$

$O_1 = f_1(A_2,A_1,A_0) = m_1 = /A_2./A_1.A_0$

$O_2 = f_2(A_2,A_1,A_0) = m_2 = /A_2.A_1./A_0$

$O_3 = f_3(A_2,A_1,A_0) = m_3 = /A_2.A_1.A_0$

$O_4 = f_4(A_2,A_1,A_0) = m_4 = A_2./A_1./A_0$

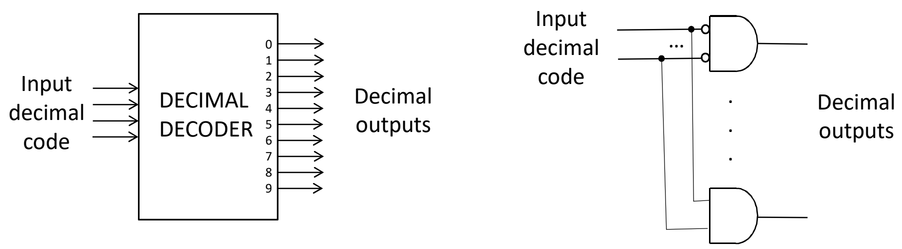
$O_5 = f_5(A_2,A_1,A_0) = m_5 = A_2./A_1.A_0$

$O_6 = f_6(A_2,A_1,A_0) = m_6 = A_2.A_1./A_0$

$O_7 = f_7(A_2,A_1,A_0) = m_7 = A_2.A_1.A_0$

Decimal decoders

A decimal decoder is a combinatorial circuit receiving a decimal binary code and producing 10 output signals, each one decoding one decimal code presented in the input.

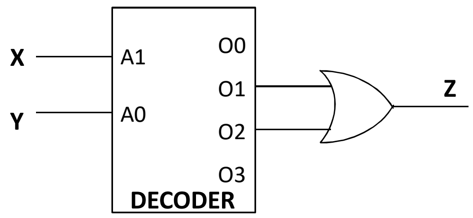


Implementation of combinatorial functions using decoders

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$$Z(X,Y) = \Sigma(1,2) \equiv O1 + O2$$

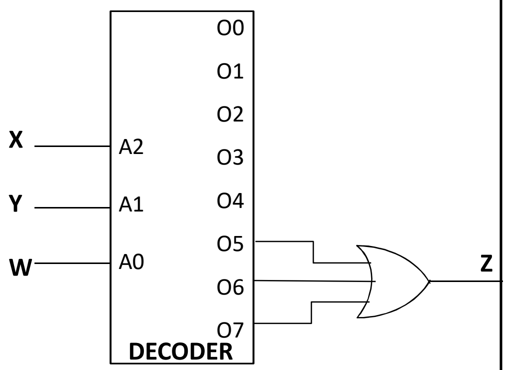
A_1	A_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



Implementation of combinatorial functions using decoders

$Z(X,Y,W) = \Sigma(5,6,7) \equiv O5 + O6 + O7$

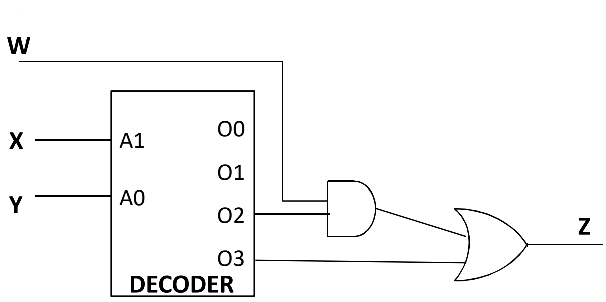
	X Y W	Z
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	0
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1



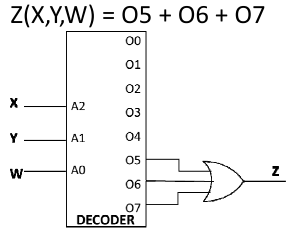
Implementation of combinatorial functions using decoders

$Z(X,Y,W) = \Sigma(5,6,7)$

	X Y W	Z
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	0
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1



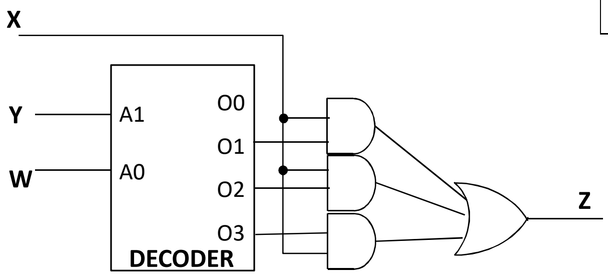
$Z(X,Y,W) = O5 + O6 + O7$

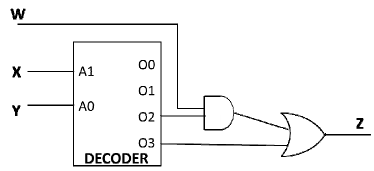


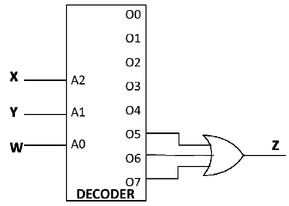
Implementation of combinatorial functions using decoders

$Z(X,Y,W) = \Sigma(5,6,7)$

	X Y W	Z
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	0
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1



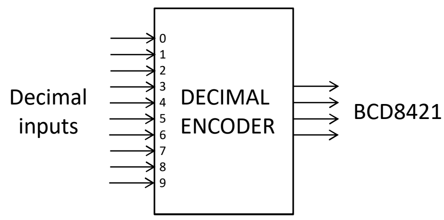




Encoder

An encoder is a combinatorial circuit generating a unique code on its outputs associated with each input signal.

Example: BCD encoder

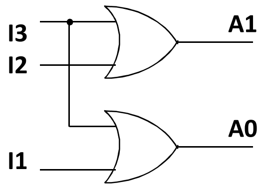


Encoding into 2-bit natural binary

I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

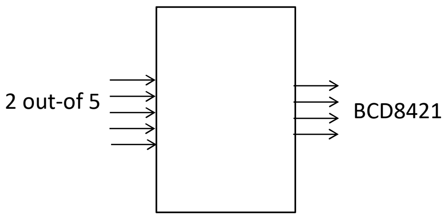
A1 = I3 + I2

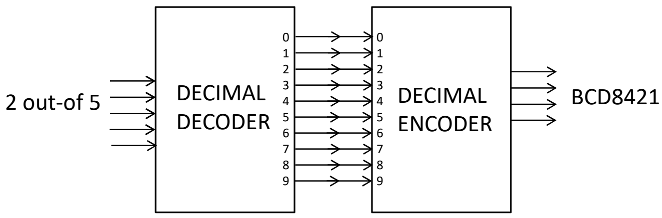
A0 = I3 + I1



Considering that at least one input is active and only one input is active at a time

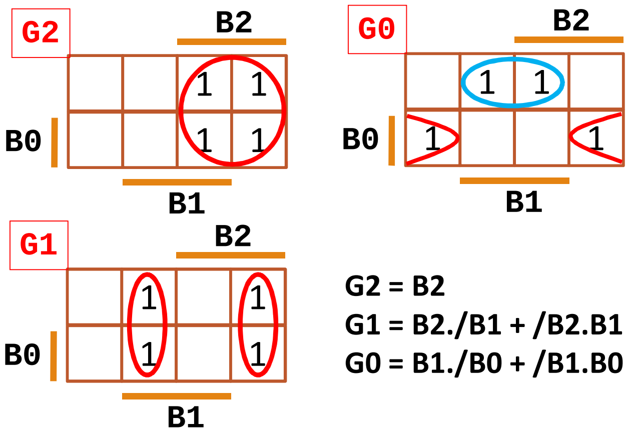
Converting codes: from 2 out-of 5 to BCD





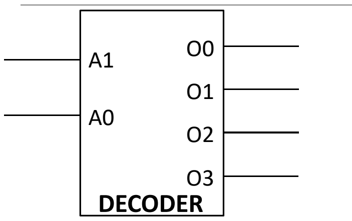
Converting codes: from natural binary to Gray (using 3 bits)

Natural Binary	$B_2B_1B_0$	$G_2G_1G_0$ Gray
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

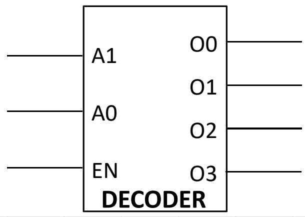


G2 = B2
G1 = B2./B1 + /B2.B1 = B2 ⊕ B1
G0 = B1./B0 + /B1.B0 = B1 ⊕ B0

Introducing an enable input

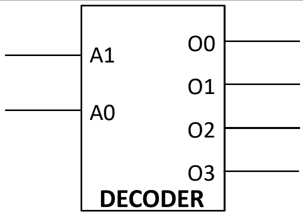


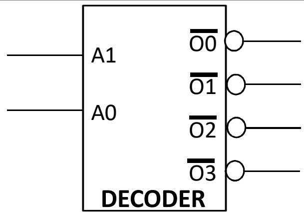
A_1	A_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



EN	A_1	A_0	O_0	O_1	O_2	O_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Active at 1 or active at 0 ?

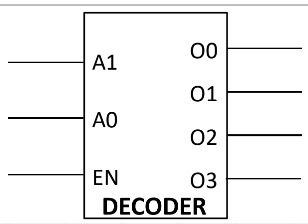


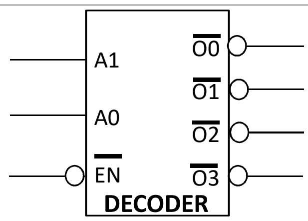


A_1	A_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

A_1	A_0	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Active at 1 or active at 0 ?





EN	A_1	A_0	O_0	O_1	O_2	O_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$\overline{EN}$	A_1	A_0	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0