EEEN 222 Digital Systems Design

Classification of electrical signals

- An electrical signal is a function of an independent variable, usually, time defined by

$$f(t) : A \to B$$

where A and B are some sets

- The electrical signals are categorized in accordance with the type of the sets A and B

A	B	f(t)
$\mathbb{R}$	$\mathbb{R}$	Continuous-time signal
$\mathbb{Z}$	$\mathbb{R}$ or $\mathbb{Z}$	Discrete-time signal
$\mathbb{R}$ or $\mathbb{Z}$	$\mathbb{R}$	Analog signal
$\mathbb{R}$ or $\mathbb{Z}$	$\mathbb{Z}$	Quantized signal

digital signal: If a quantized electrical signal is also encoded then it is referred to as digital signal

Encoding

- Assigning a distinct number in a given base for each object or character or a signal is known as encoding

  - this number is called as a "code word" and the number of digits of this code word is called as "legth of code"

- let the number of objects be n, then the length of code must be at least

$$s = \lceil \log_a n \rceil$$

where a refers to the base and $\lceil \cdot \rceil$ is the ceiling function

e.g. let $a = 2$ and $n = 5$ then

$$s = \lceil \log_2 5 \rceil$$

$$= \lceil 2.3219 \rceil$$

$$= 3$$

In particular, we have

$$a^{s-1} < n \leq a^s$$

digital system : A system which processes only digital signals

analog system : A system which processes only analog signals

hybrid system : A system which is able to process both digital and analog signals

Standard codes

- The most often used codes are

a. EBCDIC (Extended Binary Coded Decimal Interchange Code)

b. ASCII (American Standard Code for Information Interchange)

e.g. Let us encode "Z MART"

	Z		Space		M		A		R		T	
EBCDIC	F	Z	4	0	D	4	C	1	D	9	E	3
	1111	0010	0100	0000	1101	0100	1100	0001	1101	1001	1110	0011
	5	2	4	1	A	D	A	1	B	2	B	4
ASCII-8	0101	0010	0100	0001	1010	1101	1010	0001	1011	0001	1011	0100

parity bit: generated such that the number of 1's in a code becomes even (or odd) when appended and denoted by p

the codes are read out like phone numbers like 311 74 27, that is for

$$\underbrace{1111}_{F}\ \underbrace{0010}_{2} \rightarrow F\ 2$$

note that the digits of base 16 system are

0, 1, ..., 9, A, B, C, D, E, F

Applications of encoding

- Using encoding, time multiplexing can be done in communication systems
- In particular, if we need to enable more than 2 people communicate through a 2-line telephone network between two cities

(b) then we use { frequency / time } multiplexing

time multiplexing

1. The speech signal is filtered through a low-pass-filter (LPF) to eliminate the components which are of more than 4 kHz
2. The samples with 8 kHz frequency that is 125 μs apart from each other, are taken
3. The amplitude of each sample is assigned by one of distinct 128 amplitudes and encoded by 7 bits

4. A parity bit is appended to that code as the 8^{th} bit

5. The signals which belong to 32 people in an interval of 128 μs are sampled, encoded and driven to the communication line

6. Therefore, the number of bits fed to the network in 1 sec is

$$1/125\,\mu s = 8000$$

$$\#\text{ of bits} = 32 \times 8 \times 8000$$

$$= 2048 \cdot 10^3$$

$$= 2.048 \cdot 10^6$$

$$= 2\,M \text{ bits}$$

- as the 2 channels out of 32 are used for signalling

(1) the number of speech channel reduces to 30

30 channels	→ 2 M bit	1^{st} level	systems
120 channels	→ 8 M bit	2^{nd} level	
480 channels	→ 34 M bit	3^{rd} level	
1920 channels	→ 140 M bit	4^{th} level	
7680 channels	→ 565 M bit	5^{th} level	

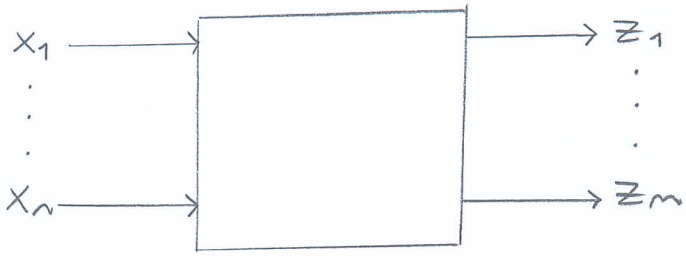
Classification of logic circuits

- The logic circuits are divided into 2 broad classes :

1. Combinational
2. Sequential

} logic circuits

- In general, a logic circuit has n-inputs and n-outputs



where

$x_1, \ldots, x_n$: free inputs which correspond to independent (voltage/current) sources in analog circuits

and $x_i, z_j \in \{0, 1\}$, $i = 1, \ldots, n$; $j = 1, \ldots, m$

Combinational logic circuit

- If the inputs at some instant are able to entirely determine the outputs, that is

$$z_i = f_i(x_1, \ldots, x_n) , \quad i = 1, \ldots, m$$

then the circuit is referred to as combinational (static, memoryless) logic circuit

Sequential logic circuit

- If the outputs of a circuit at some instant depend also on the past inputs then the circuit is so called as sequential (dynamic, memory)

logic circuit

state : the information that evolves from past to future is called state

- thus in a sequential logic circuit, we have

$$z_i = f_i(x_1, \ldots, x_n, d_j)$$

where

d_j : denotes the state at any given time

- the state is usually determined by the outputs of the flip-flops (memory elements) in sequential logic circuits

- let these flip-flop outputs be $y_1, \ldots, y_s$

then we have

$$z_i = f_i(x_1, \ldots, x_n, y_1, \ldots, y_s), \quad i = 1, \ldots, m$$

State in analog circuits

- The state variables specify the state in analog circuits

e.g. Consider the linear time-invariant discrete-time system

$$x(k+1) = Ax(k) + Be(k)$$
$$y(k) = Cx(k) + De(k)$$

- that is, the input and state at any given time determine the next state and output

Note that;

- that works out the same in sequential logic

circuits

However;

-the vericbles can tcke only the values 0 cnd 1

(1) the number of various inputs and outputs cre different and the number of stctes cre finite

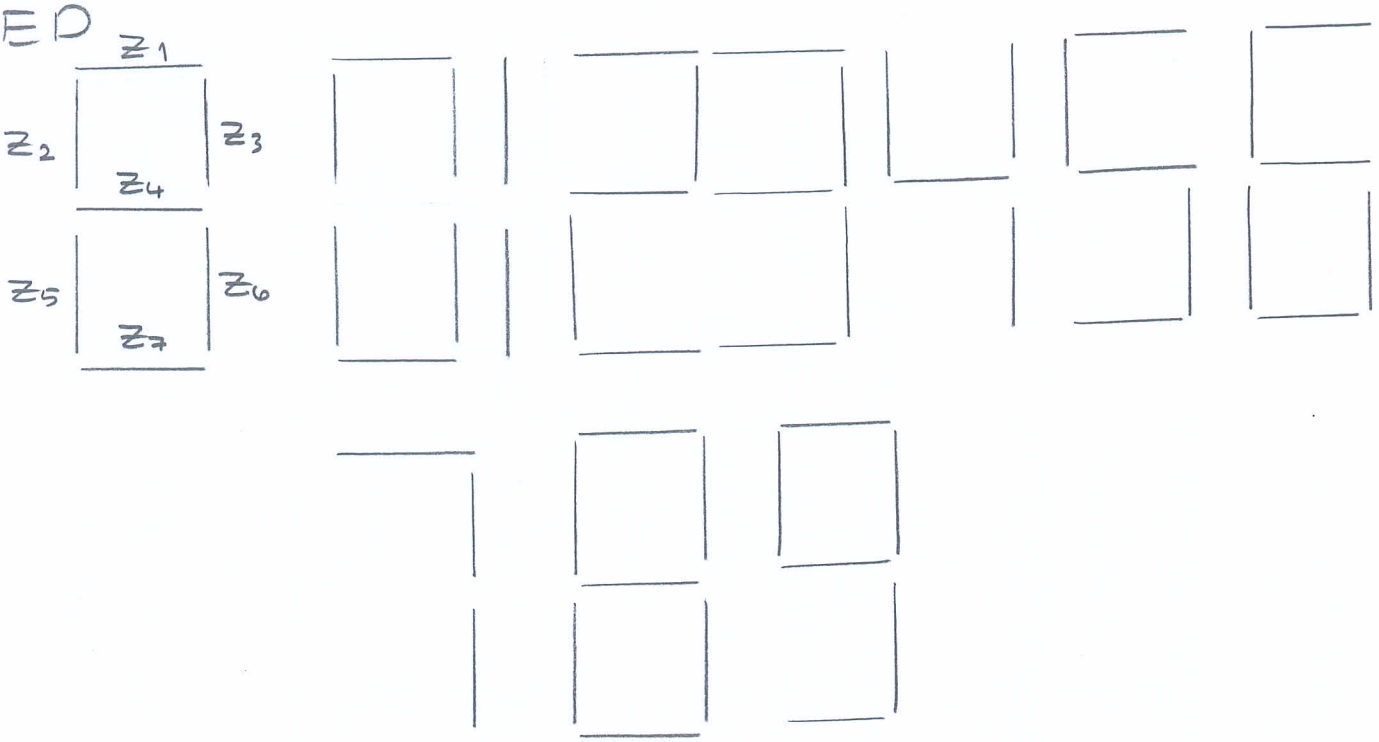
A combinctionc l logic circuit exemple

-We consider the seven-segment display unit

- a 4-input, 7-output combinctionc l logic circuit

-the input $n = x_1 x_2 x_3 x_4$ is one of the digits 0, 1, ..., 9 represented in bcse 2

-each of the seven outputs drive c single LED



- we shall list the truth table as follows

X_1	X_2	X_3	X_4	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	0	1	1	1	0	1	1	1
0	0	0	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0	1
0	0	1	1	1	0	1	1	0	1	1
0	1	0	0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1	1	1
0	1	1	1	1	0	1	0	0	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

A sequential circuit example

- We consider a parity-bit generator
- a single input, single output sequential logic circuit
- let the code length of a code word, say

$X_3 X_2 X_1$ be 3

- the sequential logic circuit specifies the number of 1's as even or odd in an incoming code word of 3 bits in length
- if the number of 1's is odd then the circuit generates 1 as the 4^{th} bit (parity bit)

$P X_3 X_2 X_1$

- if the number of 1's is even then it produces

0 as the 4th bit (parity bit)

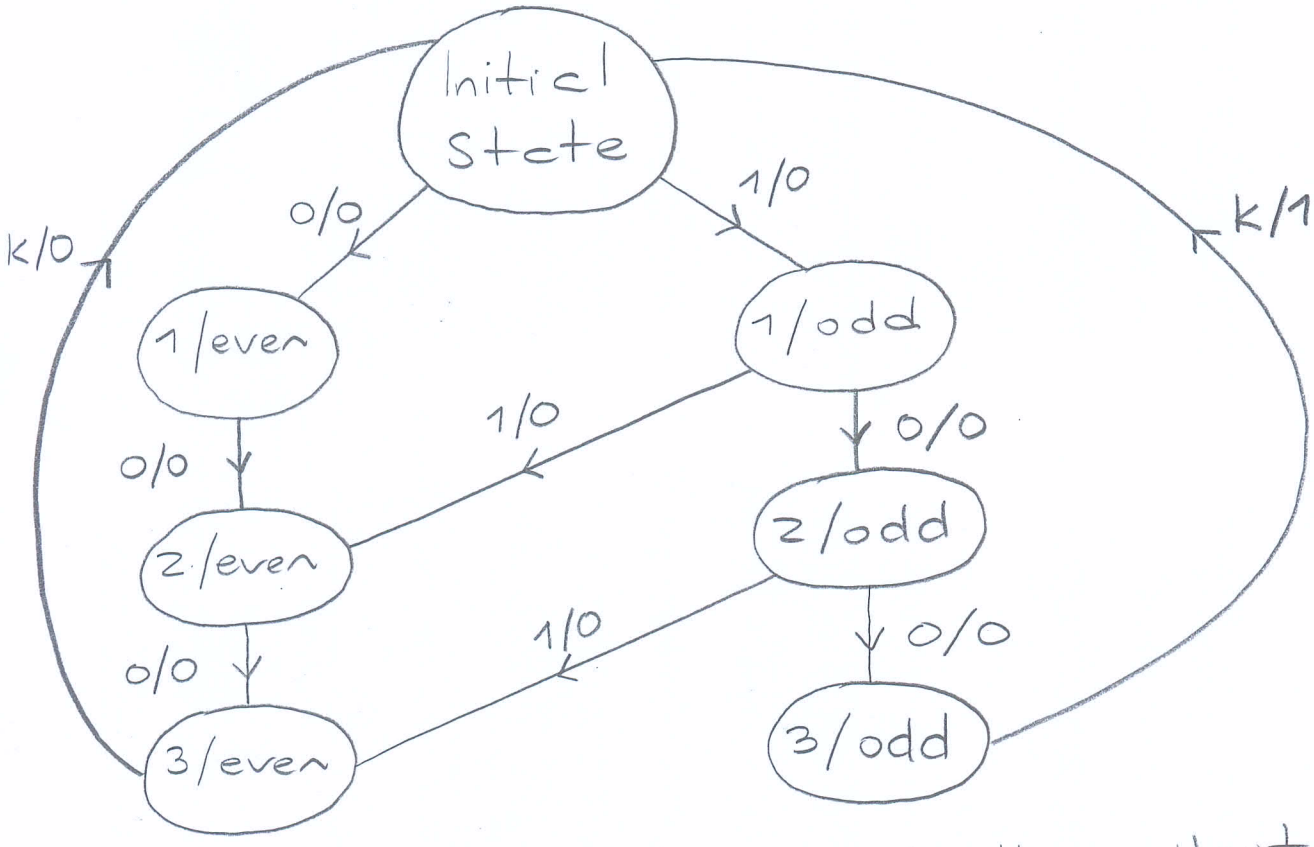
observations

- for the first 3 inputs, the output of the circuit can be 0 or 1

(!) only the 4th output is important

- the 4th output is independent from the 4th input

state diagram



- it can thus be seen that the output can be 0 or 1 when the input is 1

(!) underlining that the input does NOT determine the output on its own

- if both input and state are given then the output can be specified

-as the number of states is +

(1) then the number of state varicble is

$$\lceil \log_2 7 \rceil = \lceil 2.8074 \rceil$$

$$= 3$$

-this implies that a totcl of 3 flip-flops (memory elements) need to be used in the mplemented circuit

Electricel cnd Electronics Industry

-fundcmentcl pert of flight, textile, cutomotive, cnd quite mcny other fields of industry

-the cnclog electricel circuits cre most often eplcced by digitcl circuits

Advcntcges of digitcl circuits

1. Suitcble for the production cs integrcted circuits
2. Highly secured operction
3. Possess high dynemics (the retio of the lergest cmplitude signcl to the smcllest nonzero cmplitude signcl)
4. Reproducible signcls
5. Ecsiness of sevins information
6. Less noise sensitivity
7. Possible to cetch the incorrect signcl thet occur throush encoding cnd correct approprictely

8. Advanced systems can be realized with microprocessors and some other elements

Advantages of analog circuits

1. No quantization noise exists
2. Suitable for real-time operation (no need for encoding)
3. Preferrable to be operated for a while instead of replacing with digital counterparts

Binary numbers and binary arithmetic

- Binary number system refers to a base-2 system with two digits : 0 and 1
- A binary number is expressed with a string of 1's and 0's and possibly a binary point

e.g. 10101.0101

- The decimal equivalent can be found by expanding the number as a power series in base-2

e.g. $(11010)_2 = 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0$

$= (26)_{10}$

$(110101.11)_2 = 1.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 1.2^{-1} + 1.2^{-2}$

$= (53.75)_{10}$

- some terminology follows as

$2^{10} = 1024 \rightarrow$ K (Kilo)

$2^{20} = 1.048.576 \rightarrow$ M (Mega)

$2^{30} = 1.0737.10^9 \rightarrow$ G (Giga)

Conversion of a decimal number to binary

- We successively subtract powers of 2 from the decimal number:

1. Find the greatest number that is a power of 2 but when subtracted from N, produces a positive number.
2. Let the difference be N_1.
   Find again the greatest number that is a power of 2 and when subtracted from N_1, produces a positive difference N_2.
3. Continue this process until the difference is zero.
4. The equivalent binary number is obtained from the binary coefficients of a power series that forms the sum of the components

e.g.

$$625 - 512 = 113 = N_1 \qquad 512 = 2^9$$
$$113 - 64 = 49 = N_2 \qquad 64 = 2^6$$
$$49 - 32 = 17 = N_3 \qquad 32 = 2^5$$
$$17 - 16 = 1 = N_4 \qquad 16 = 2^4$$
$$1 - 1 = 0 = N_5 \qquad 1 = 2^0$$

$$\Rightarrow (625)_{10} = (1001110001)_2$$

Octal and hexadecimal numbers

- The octal (base-8) and hexadecimal (base-16) systems are useful

  (1) because their bases are powers of 2

  that is ;

$$2^3 = 8 \quad , \quad 2^4 = 16$$

- octal digits : 0, 1, 2, 3, 4, 5, 6, 7

  (1) can be represented with 3 binary digits

- hexadecimal digits : 0, 1, ..., 9, A, B, C, D, E, F

- these base systems are more compact and more convenient for people to use

e.g. Consider an octal number

$$(127.4)_8 = 1 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 + 4 \cdot 8^{-1}$$

$$= (87.5)_{10}$$

e.g. Consider a hexadecimal number

$$(B65F)_{16} = 11 \cdot 16^3 + 6 \cdot 16^2 + 5 \cdot 16 + 15 \cdot 16^0$$

$$= (46687)_{10}$$

Conversion from binary to octal

- We partition the binary number into groups of 3 bits each
- the corresponding octal digit is assigned to each group

e.g.

$$(010\ 110\ 001\ 101\ 011.111\ 100\ 000\ 110)_2$$

$$= (2\ 6\ 1\ 5\ 3.7\ 4\ 0\ 6)_8$$

Conversion from binary to hexadecimal

- We decompose the number into groups of 4 digits
- then the corresponding hexadecimal digit is assigned to each group

e.g.

$$(0010\ 1100\ 0110\ 1011.1111\ 0000\ 0110)_2$$

$$= (2\ C\ 6\ B.F\ 0\ 6)_{16}$$

Reverse conversion

- Each octal / hexadecimal digit is converted to its three / four bit binary equivalent

Arithmetic operations

- Arithmetic operations with numbers in base r follow the same rules as for decimal numbers
- but we can use only r allowable digits

Addition in binary numbers

Carries :	00000	1 0 1 1 0 0
Augend :	01100	1 0 1 1 0
Addend :	+ 10001	+ 1 0 1 1 1
Sum :	11101	1 1 1 1 0 1

- the sum digit in any position can be only 1 or 0
- a carry occurs if the sum in any bit position is greater than 1

Subtraction in binary numbers

Borrows :	0 0 0 0 0	0 0 1 1 0
Minuend :	1 0 1 1 0	1 0 1 1 0
Subtrahend :	− 1 0 0 1 0	− 1 0 0 1 1
Difference :	0 0 1 0 0	0 0 0 1 1

- A borrow into a given column adds 2 to the minuend bit
- In the case when subtrahend > minuend, we reverse the process and give the result a minus sign

Multiplication in binary numbers

- let us consider the following example

Multiplicand : 1 0 1 1

Multiplier : 1 0 1

$$
\begin{array}{r}
\times \\
1\,0\,1\,1 \\
0\,0\,0\,0 \\
+\;1\,0\,1\,1 \\
\hline
1\,1\,0\,1\,1\,1
\end{array}
$$

Arithmetic operations in base-r system

- An alternative way for the arithmetic operation of two numbers in base r :

↳ - convert them to decimal
- operate in decimal
- convert the result to base-r system

Conversion of decimal fractions to binary

- Multiply the decimal fraction repectedly to give an integer and a fraction
- terminate until the fractional part is zero

e.g.

$0.6875 \times 2 = \boxed{1}.3750$

$0.3750 \times 2 = \boxed{0}.7500$

$0.7500 \times 2 = \boxed{1}.5000$

$0.5000 \times 2 = \boxed{1}.0000$

MSB
|
⋮
↓
LSB

$\Rightarrow (0.6875)_{10} = (0.1011)$

Remark. Converting a decimal fraction to base-r needs to repectedly multiply by r