Name :- Ishant Khurana

Roll No :- 19010727

EE213M

Digit circuits

Notes Assignment - 1

Lecture - 1

Background and number systems

Benefits of using digital representation

1. Signals processed with minimal degradation

Advancement of technology

Computers can be used to process data

Data can be stored in a variety of storage media

Economy systems can be produced at a very low cost

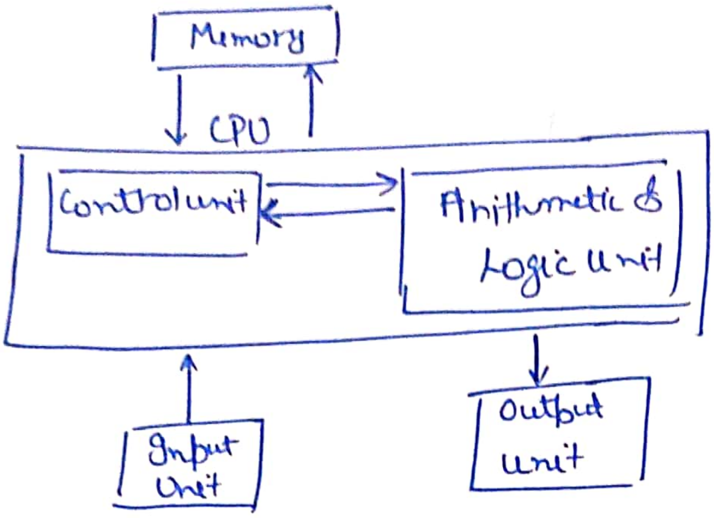
Reproducibility - for same set of input we get the same output in 2 similar digital circuit.

Programmability using HDLs (Hardware description languages)

Flexibility and functionality while using digital representation

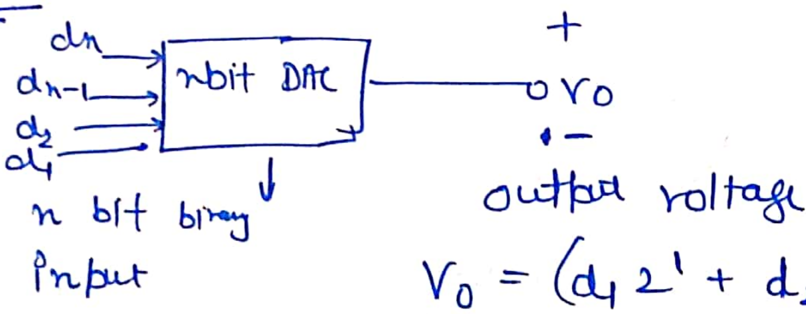
Ease of design

Block Diagram of a Generic Digital Computer :-



Digital systems :- Most real life systems are analog. They are converted to digital for storing/processing using ADCs (Analog to digital converters) & reconverted back to analogy using DAC's (digital to analog converters) for further use.

Eg DAC :



$$V_0 = (d_1 2^{1} + d_2 2^{-2} + \dots + d_n 2^{-n}) V_{ref}$$

Smallest possible voltage change by DAC $= (2^{-n} V_{ref})$

Egs. Microwave oven, digital camera

Small computer/micro computers which are more prevalent, less powerful & only make parts of products (enclosed within them) are called embedded systems

Embedded System

Either Analog input devices (photodiode/tachometer/thermistors) could be used in embedded systems or digital input devices could be used. Analog would require an ADC.

Pre-defined software functions specific to the product called embedded software

we do not interact with these systems or limited interaction with it

→ output devices can be (LED displays, relays, stepper motor etc.)

Number systems →

Binary → Base 2 → (Polynomial in power of 10)

Octal → Base 8

Hexa-decimal → Base 16

Decimal → Base 10

Conversion :-

$$(312.4)_5 = 3\times 5^2 + 1\times 5^1 + 2\times 5^0 + 4\times 5^{-1} = (82.8)_{10}$$

$$(11010)_2 = 1\times 2^4 + 1\times 2^3 + 0\times 2^3 + 1\times 2^1 + 0\times 2^0 = (26)_{10}$$

Scanned with CamScanner

→ 2^{10} = Kilo
2^{20} = Mega
2^{30} = Giga
2^{40} = Tera

→ $(625)_{10} = (??)_2$

→ Subtract highest power of 2 from table untill it becomes 0

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

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

Decimal to any other Radix :-

Eg. 53 to binary

$2 \overline{)53}$ rem = 1 = a_0
$2 \overline{)26}$ rem = 0 = a_1
$2 \overline{)13}$ rem = 1 = a_2
$2 \overline{)6}$ rem = 0 = a_3
$2 \overline{)3}$ rem = 1 = a_4
$2 \overline{)1}$ rem = 1 = a_5
0

$\therefore 53_{10} = 110101_2$

Eg. 0.7 to binary

.7
× 2
(1) .4
× 2
(0) .8
× 2
(1) .6
× 2
(1) .2

repeat

$\therefore 0.7_{10} = (0.10110\ 0110\ 0110..)$

Scanned with CamScanner

Lecture - 2

N bits can represent values from 0 to $(2^n - 1)$

Method → Decimal to Binary (Power of 2s method)

Eg. $(625)_{10} = (??)_2$

512	256	128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	0	0	1

$625 - 512 = 113$

$113 - 64 = 49$

$49 - 32 = 17$

$17 - 16 = 1$

$1 - 1 = 0$

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

fill one's for powers we used 0 for others

Octal system → Radix $= 8 = 2^3$ 3 bit binary = 1 octal no.

Binary to Octal Conversion :- Integer → LSB to MSB, 3 bits group.

Fraction → MSB to LSB, 3 bits groups

Eg. 1. $(1111011001)_2 = \underset{1}{1}\ \underset{7}{111}\ \underset{3}{011}\ \underset{1}{001} = (1731)_8$

Octal to Binary → Each digits is represented by its 3 digited binary value.

Hexadecimal system → Radix $= 16 = 2^4$

0 1 - - - 9 A B C D E F → Every 4 bits binary no. → 1 hexadecimal.

Scanned with CamScanner

Binary to Hexadecimal →

$(0010 \quad 1100 \quad 0110 \quad 1011 \quad 1111 \quad 0000 \quad 0110)_2$

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

Octal to Hex :-

1. Use binary as intermediate

Eg. $(3541)_8 = ??_{16}$

→ $\overset{3}{011}\ \overset{5}{101}\ \overset{4}{100}\ \overset{1}{001}$

$0111 \quad 0110 \quad 0001 = \boxed{(761)_{16}}$

Hex to Octal

Eg : $1F0A.A8_{16} = ??_{16}$

$0\ 001 \quad 1111 \quad 0000 \quad 1010 \quad 1010 \quad 1000$

$0001 \quad 111 \quad 100 \quad 001 \quad 010 \quad 101 \quad 010 \quad 000$

$1 \quad 7 \quad 4 \quad 1 \quad 2 \quad 5 \quad 2 \quad 0$

$= (17412.52)_8$

Main use of octal / hexadecimal → to represent binary no.s compactly

No. of bits	Term
1 bit →	Bit
4 →	Nibble
8 →	Byte
16/32/64 →	words

Scanned with CamScanner

Ranges

1. Unsigned Nos → 16 bit unsigned integer range →

n bits → 0 to $(2^n - 1)$

∴ 16 → 0 to 65,535

16 bit unsigned fractions

$2^{-1}+2^{-2}+2^{-3}$

Max. fraction value = 0.111 111 111 . .

$$\therefore S_n = \left(\tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} - - - \tfrac{1}{2^{n-1}}\right) + \tfrac{1}{2^n}$$

$$2S_n = \left(1 + \tfrac{2}{4} + \tfrac{2}{8} - - -\right) = 1 + \left(\tfrac{1}{2} + \tfrac{1}{8} \cdot \cdot\right)$$

$$= 1 + \left(S_n - \tfrac{1}{2^n}\right)$$

$$\boxed{S_n = 1 - \tfrac{1}{2^n} = \frac{2^n - 1}{2^n}}$$

→ Fraction values range upper limit.

∴ 16 → 0.0 to 65,535 / 65,536

16 → 0.0 to 0.99984 7412

Arithmetic operations :-

* Binary addition :-

Carries	00000
Augend	01100
Addend	+1 0001
Sum	11101

```
  0 1100
  1 0110
  1 0110
 1 01101
```

* Binary Subtraction :-

Borrows	00000
Minuend	10110
Subtrahend	-10010
Difference	00100

```
  00110
  10110
 -10011
  00011
```

Eg. Subtrahend > Minuend → reverse no.s & add -ve sign.

Eg.
```
  10011
 -11110
 ------
 -01011
```
⇒
```
   10110
   11110
 - 10011
 -------
   01011
```
$\xleftarrow{-ve}$

Binary Multiplication :-

Multiplicand (1011)
Multiplier × 101

```
       1011
 x      101
 ----------
     1 011
 +00 000
 +101 1
 ----------
   110 111
```

→ For base b arithmetic operations, convert to decimal, do the operation & convert all sum & carry to corresponding base b.

Hexadecimal addition :-

$(59F)_{16} + (E46)_{16}$

carry=1 carry=1

```
 1      1
    5   9   15
   14   4    6
 -----------------
 19=16+3  14  21=16+5
```

Multiple of 16

∴ $(13E5)_{16}$

190107027

Octal Multiplication :-

$$(762)_8 \times (45)_8 = (45772)_8$$

```
  762
   45
-------
①4 5 7 2
3 7 1 0 x
-------
4 3 7 7 2
```

Octal 5×2 — Decimal $10 = 8+2$; Multiplying from octal 12

$5\times6+1 = 31 = 24+7$

$5\times7+3 = 38 = 32+6$

$4\times2 = 8$

$4\times6+1 = 25 = 24+1$

$4\times7+3 = 31 = 24+7$

$11 \rightarrow 8+3$

13

Binary Long division :-

Eg: Divisor 11001, quotient 10110, dividend 1000100110

```
          10110
11001 ) 1000100110
        11001
        ------
         100101
          11001
         ------
          11001
          11001
          ------
          000000   Remainder
```

To verify → convert to decimal divided convert back.

Lecture 83.

Solns are 5 & 8. What is base?

$x^2 + 21x + 104 = 0$

$(x-5)(x-8) = x^2 - (5+8)x + 5 \times 8$

$13 = 5+8 = 2 \times b^1 + 1 \times b^0 = b = 6$

$5 \times 8 = 40 = (104)_6$

$\Rightarrow 4 \times 6^0 + 0 \times 6^1 + 1 \times 6^2$

Range for signed nos :- Out of n bits, 1 is reserved for sign, n−1 for storing nos.

$\therefore$ signed nos. range $= \underline{-(2^{n-1}-1) \text{ to } (2^{n-1}-1)}$

3 ways to represent −ve nos :-

1. sign is method
2. 1s complement

} Range $\rightarrow -(2^{n-1}-1)$ to $+(2^{n-1}-1)$
zero can be represented as −0 & +0

3. 2s complement $\rightarrow$ Range is $\longrightarrow -2^{n-1}$ to $(2^{n-1}-1)$
only (+0)

Left most bit $\rightarrow$ 0 then +ve value
1 then −ve value

Overflow :- when result falls out side the specified range

1. Sign & Magnitude representation :-

Range : $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

Eg : n=4

Positive	Negative
0000(+0)	1000(−0)
0001(+1)	1001(−1)
0010(+2)	1010(−2)
⋮	⋮

Scanned with CamScanner

190107027

Drawback : ▶ (+0) & (-0) representation is un-necessary & thus this method is not the best

1s Complement →

Negative no., denoted by 1s complement of N, denoted by N'

$$N' = (2^n - 1) - N$$

1s Complement of N is obtained by complement N bit - by bit

Eg. $n = 4$

Positive	Negative
0000	1111
0001	1110
⋮	⋮
0111	1000

$(16-1) - 0 = 15 = 1111$
$(16-2) - 1 = 14 = 1110$

Advantage :-

1. Subtraction can be done using addition →

$A.B = \underbrace{A+B'}$

→ if no carry, -ve no. (in 1s complement form)
→ if carry, +ve no. This is called end-around carry
↳ add carry back to A+B'.

Eg. $A = 2$; $B = 0$

A+ B

$$\begin{array}{r} 0010 \\ +\ 1001 \\ \hline 1011 \\ \hline 100 \end{array} = \boxed{-4}$$

Scanned with CamScanner

2s complement →

→ positive integer +N by 0 followed by magnitude N.
Negative integer −N represented by 2 s complement i.e. N^*.

$$N^* = 2^n - N$$

i.e. $\boxed{\text{1s complement} + 1}$

+ve	−ve
0 000 (+0)	—
0001 (+1)	1111 (−1)
0010 (+2)	1110 (−2)
0 011 (+3)	1101 (−3)
⋮	⋮

0010 (+2)
↓
1101
+ 1
1110 (−2)

Note: $N^* = (2^n - N)$ directly on Binary nos?

eg $n = 7$ → largest integer $(2^7 - 1) = 1111\ 111$
↳ For 2^7, 8 bits necessary.

∴ To avoid this, eqⁿ is rewritten as → $N^* (2^n - 1 - N) + 1$

eg. N = 0101100 $\quad 2^n - 1 = 1111\ 111$
$\quad - 0101100$

$2^n - 1 - N = 1010011$
$+ \ 0000001$
$N^* = 1010100$

Easier way: Leaving all zeroes from right of the first one, complement all other digits of N bit by bit eg. N = 0101100

$\boxed{N^* = 1010100}$

NOTE :- (a) Weighted no. representation of 2's complement, but with MSB having weight -2^{n-1}

Eg:- 1101 ⟶ -ve no. (-3) in 2's complement

$-1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = -8+5 = -3$

(b) Shift left by k positions with 0 padding, multiplies the number by 2^K ⟶

Eg:- 00010011 = +19

↓ shift left by 1 position ↓ $\times 2^1$

00100110 = +35

↓ $\times 2^2$

01001100 = +76

(c) Shift ~~left~~ right by K positions with zero padding divides no. by 2^K (one padding for -ve nos)

Eg:- 00110100 = +52

00011010 = +26 $\frac{52}{2^1}$

00001101 = +13 $\frac{52}{2^2}$

00000110 = +6 $\frac{52}{2^3}$

(d) Sign bit copied as many times required in beginning to extend size of no. ⟶ AKA sign extension

Eg:- 8 bit ⟶ N = 00110000 = +48

31 instead of 0, 1 is copied 2^n times.

32 bit ⟶ N = 00000000 00000000 00000000 00110000

Scanned with CamScanner

190107027

Subtraction using 2s Complement :-

A. No overflow →

1. $C = A - B$ obtain $A + B^*$
2. If carry is 1, ignore it → result is +ve no
3. else result is -ve no. in 2s complement from inc.

$A = 3 \quad B = 5$

$\therefore A + B^* = 0011$

$\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;1011$

$\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;\;1110 = -8 + 4 + 2 + 0 = -2$

$A = 5, B = 3$

$A + B^* = 0101$

$\;\;\;\;\;\;\;\;\;\;\;\;\;\;+1101$

$\;\;\;\;\;\;\;\;\;\;\;\;\;\;10010$

ignore +2

B. Overflow →

↓ In 4 bit repersentation, range is +7 to -8

can occur when → 1. sign of 2 no.s is same
2. sign of sum is diff^n from sign of either of no.s

Eg. $A = -6, B = -3$

1010

1101

$A + B = 10111 = \boxed{+ve\ No.}$ wrong ans

Eg $+5 + +6 = +11$

$0101 + 0110 = 1011 = -5$

$+7 + 7 = +14$

$0111 + 0111 = 1110 = -2$

Scanned with CamScanner