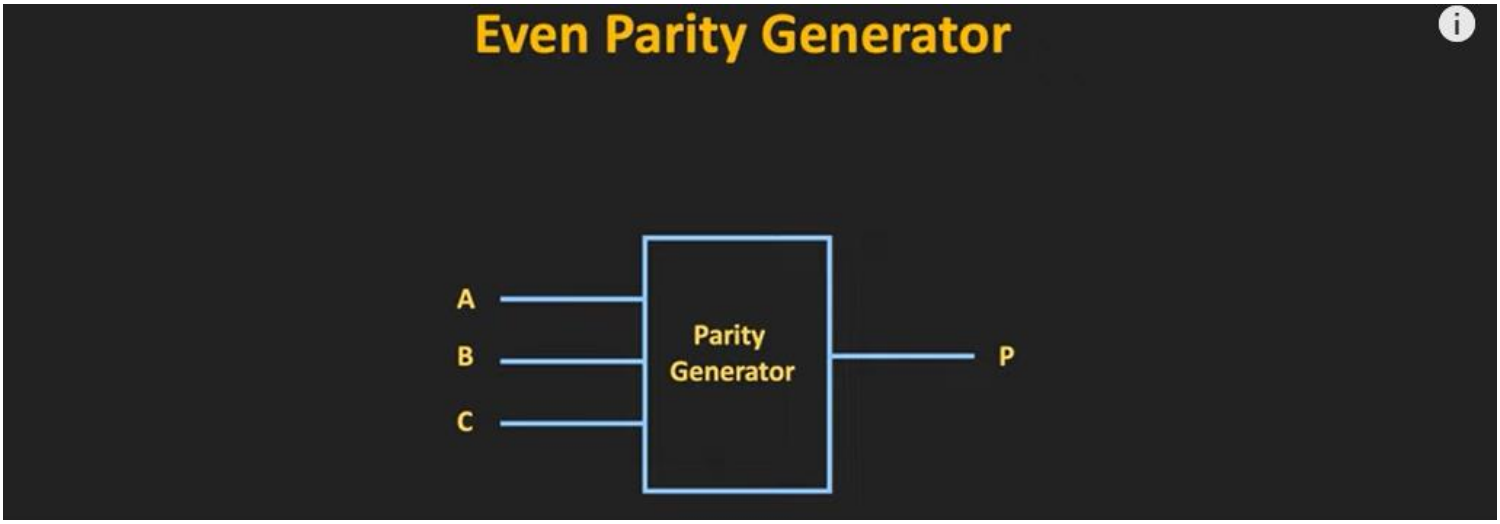
No E-Mail submissions will be accepted.
Submission formats and file naming:
File name : Pts_firstName_lastName_lab_7
File format: pdf or MS Word format
e.g. Pts_Donald_Trump_lab_7.pdf

Reading materials

Use the following link and write a one page summary about the movie.

Parity Generator and Parity Checker Explained
https://youtu.be/c8qAti1zBVQ



This video explains how parity generators and parity checkers help detect errors in data transmission. When data is sent between computers or devices, small mistakes can occur due to interference or noise. To catch these errors, an extra bit, called the parity bit, is added to the data. There are two types of parity: even parity, where the total number of 1s ,including the parity bit , must be even, and odd parity, where the total number of 1s must be odd. When the receiver gets the data, it checks whether the number of 1s still matches the expected parity. If it doesn't, an error is detected, alerting the system that something messed up.

The parity generator circuit creates the parity bit based on the data bits. The video demonstrates how a 3-bit even parity generator works. It takes three data bits, A, B, and C and determines whether the total number of 1s is even or odd. If the number of 1s is even, the parity bit is 0, and if the number of 1s is odd, the parity bit is 1. The circuit for this is built using XOR gates, since an XOR operation outputs 1 when the number of 1s is odd. The formula for this even parity bit is A XOR B XOR C. For odd parity, the logic is just the opposite, so the output is the inverse of the even parity bit and can be created by adding a NOT gate.

Once the data is sent along with the parity bit, the parity checker at the receiving end verifies if the data has been altered. It does this by checking the total number of 1s, including the parity bit. If the parity check does not match the expected

value, it means an error has occurred. However, while this method is effective at detecting single-bit errors, it cannot correct them. Also, if two bits flip, the parity remains unchanged, and the error goes undetected. This is a limitation of simple parity checking.

To simplify parity operations, special integrated circuits or ICs like the 74180 IC are available. These chips can automatically generate and check parity for multiple bits at once. Parity checking is widely used in computer systems, networking, and storage devices as a basic form of error detection, ensuring data integrity in many applications.

1) The word "Covid19" is given.
(a) Represent in an 8-bit ASCII code.

Covid19 => 01000011 01101111 01110110 01101001 01100100 00110001 00111001
Or 0x43 0x6F 0x76 0x69 0x64 0x31 0x39

Letter	ASCII (dec)	ASCII (bin)	ASCII
C	67	01000011	0x43
o	111	01101111	0x6F
v	118	01110110	0x76
i	105	01101001	0x69
d	100	01100100	0x64
1	49	00110001	0x31
9	57	00111001	0x39

(b) Add <u>even parity bit</u> to each character (from left) and represent each character in hex,

	ASCII (Hex)	Code word = Parity bit + data Binary format	Code word = parity bit + data Hex format
K	0x4B	001001011	0x04B
C	0x43	101000011	0x143
o	0x6F	001101111	0x06F
v	0x76	101110110	0x176
i	0x69	001101001	0x069
d	0x64	101100100	0x164
1	0x31	100110001	0x131
9	0x39	000111001	0x039

C/ 0x43 (0100 0011) -> 101000011 odd i/ 0x69 -> 001101001 even

O/ 0x6F -> 001101111 even d/ 0x64 -> 101100100 odd

V/ 0x76 -> 10111011 odd 1/ 0x31 -> 10011 0001 odd

Q/ 0x39 -> 0 0011 1001 even

2) Use the given formula below to obtain the 7-bit Hamming codeword for the following 4-bit data words:

Data word	= abcd	x = a⊕b⊕d
Parity bits	= xyz	y = a⊕c⊕d
Codeword	= xyazbcd	z = b⊕c⊕d

a) 1010 (abcd)

x = 1⊕0⊕0 = 1

y = 1⊕1⊕0 = 0

z = 0⊕1⊕0 = 1

hamming code = 1011010

b) 1100 (abcd)

x = 1⊕1⊕0 = 0

y = 1⊕0⊕0 = 1

z = 1⊕0⊕0 = 1

= 0111100

c) 1110 (abcd)

x = 1⊕1⊕0 = 0

y = 1⊕1⊕0 = 0

z = 1⊕1⊕0 = 0

= 0010110

3) Which of the following 7-bit Hamming codes is corrupted?

a) 1011011 (xyabcd)

x = 1⊕0⊕1 = 0 ✗

y = 1⊕1⊕1 = 1

z = 0⊕1⊕1 = 0

this is corrupted

b) 0011001 (xyabcd)

x = 1⊕0⊕1 = 0 ✓

y = 1⊕0⊕1 = 0 ✓

z = 0⊕0⊕1 = 1 ✓

not corrupted

c) 1010000 (xyabcd)

x = 1⊕0⊕0 = 1 ✓

y = 1⊕0⊕0 = 1 ✗

z = 0⊕0⊕0 = 0

this is corrupted.

4) Obtain the 12-bit Hamming code word for the following 8-bit data word.

1 2 3 4 5 6 7 8
11011011

P1 = D1 ⊕ D2 ⊕ D4 ⊕ D5 ⊕ D7
P1 = 1 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 1 = 1

P2 = D1 ⊕ D3 ⊕ D4 ⊕ D6 ⊕ D7
P2 = 1 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 1 = 1

P3 = D2 ⊕ D3 ⊕ D4 ⊕ D8
P3 = 1 ⊕ 0 ⊕ 1 ⊕ 1 = 1

P4 = D5 ⊕ D6 ⊕ D7 ⊕ D8
P4 = 1 ⊕ 0 ⊕ 1 ⊕ 1 = 1

P1 = 1
P2 = 1
P3 = 1
P4 = 1

1	2	3	4	5	6	7	8	9	10	11	12
P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8
1	1	1	1	1	0	1	1	0	1	1	1

= 111110111011

5) The following string of hex digits encodes extended ASCII characters in an even parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded (**8 data bits + 4 check bits = 12 bits**).

e.g. **49C**
01001001110**0** ⇒ 01001001110**1** ⇒ 01001101 0x4D (77) ⇒ M

1	2 ✓
2	2 ✓
4	1 ×
8	3 ×

Location
4 + 8 = 12

Encoded(hex)	49C
Word	01001101
Location of corrupted bit	12
ASCII Character	M

Encoded(hex)	0D3	DD3	0F2	5C1	1C5	CE3
Word	01100D3	01100D1	01100I0	01101001	0110019	0111011
Location of corrupted bit	12	11	7	9	1	nom
ASCII Character	b	a	b	i	e	S

0D3
1 2 3 4 5 6 7 8 9 10 11 12
0000 1101 0011 → 0110 0010 → 0x62 → b
0111
corrected

1 two 1s ✓
2 two 1s ✓
4 three 1s ×
8 three 1s ×

→ = 4 (bit4) + 8 (bit8)
= 12

DD3
1 2 3 4 5 6 7 8 9 10 11 12
1101 1101 0011 → 11011101 0001 → 01100001 → 0x61 →
a

1 3 1s ×
2 3 1s ×
4 4 1s ✓
8 3 1s ×

→ 1 + 2 + 8 = 11

0F2
1 2 3 4 5 6 7 8 9 10 11 12

0000 1111 0010 → 0000 1110 0010 → 0 110 0010 → 0x62 → b

1 3 is X
2 3 is X → 1+2+4 = 7
4 3 is X
8 2 is ✓

5C/ 0101 1100 0001 → 0x0x 1100 1001 → 0110 1001 → 0x69 → i

1 1 is X
2 2 is ✓ 1+8 = 9
4 4 is ✓
8 1 is X

1C5/ 0001 1100 0101 → 0x01 1100 0101 → 0110 0101 → 0x65 → e

1 1 is X
2 2 is ✓ 1
4 4 is ✓
8 2 is ✓

CE3/ 1100 1110 0011 → 0x00 1110 0011 → 0111 0011 → 0x73 → s

1 4 is ✓
CR 4 is ✓ → no corruption
4 4 is ✓
8 2 is ✓

6) Encode the data bit sequence 100110000 using the generator 1101 and give the codeword.

```
1101 | 100110000
       1101
       -------
       X100100000
        1101
        -------
        X1000000
         1101
         -------
         X101000
          1101
          -------
          X11100
           1101
           -------
           X0110
```

code word = 100110000 110

0000
x110
=110

8 bits:
Example: A (0x41) = 01000001

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL