Digital Design

Ms. Umarah Qaseem

1

10

Chapter 4

Combinational Logic

2

10

Week 10

Adder/Subtractor

BCD Adder

Multiplier

3

Some Cool Stuff

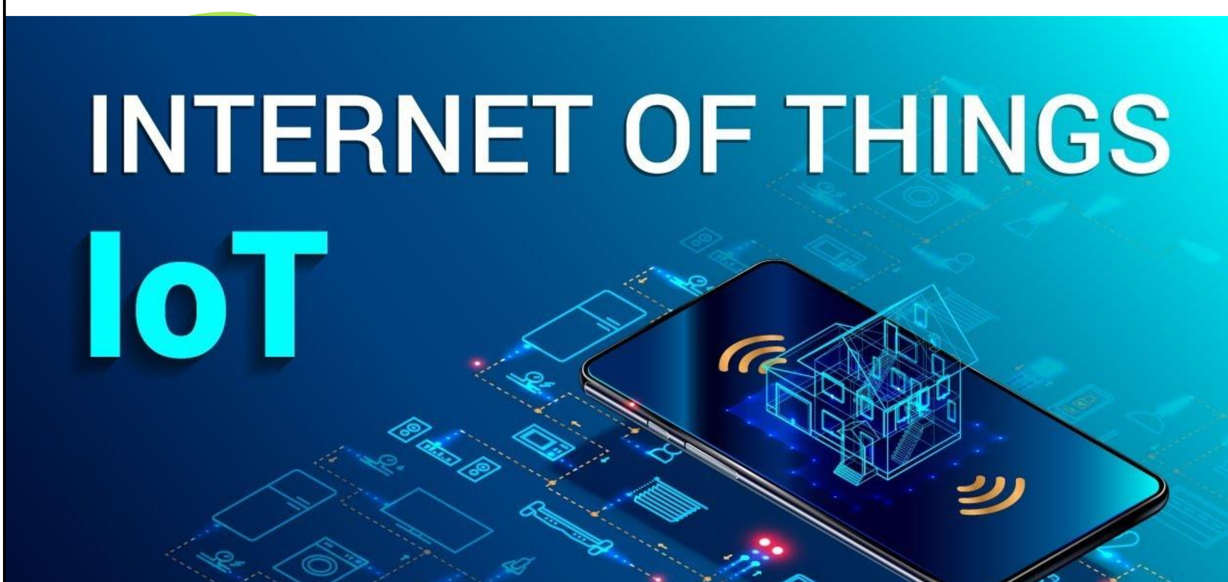
4

ESP8266!
Making a Web server

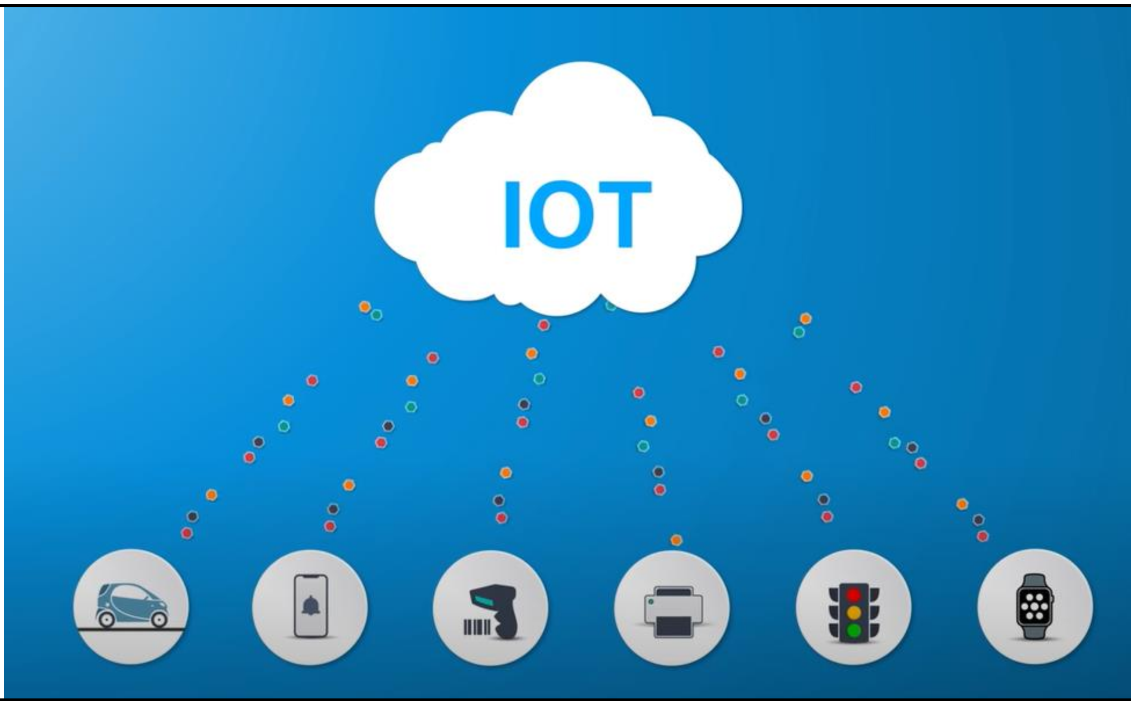
HOW COOL IS THAT

Controlling your circuit through your phone!

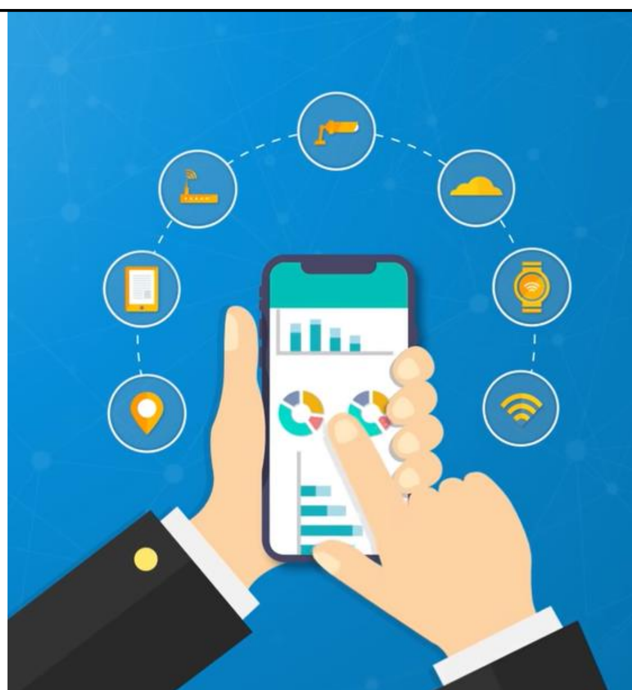
5



6



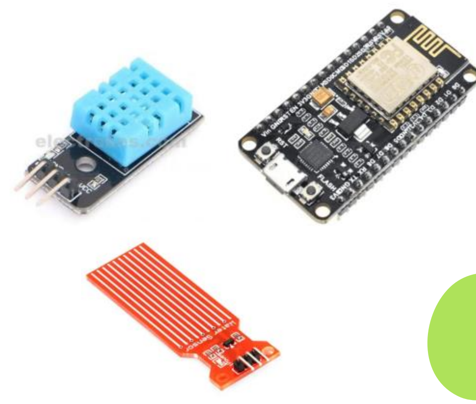
7



8

Material to be used in IOT Lab

- ESP8266 V3 (550/-)
- ESP8266 V2 (650/-)
- DHT11 (240/-)
- SR04 (170/-)
- Water Level Sensor (70/-)



9

9

Binary Adder/ Subtractor

10

Review complements:

- The subtraction of unsigned binary numbers can be done by complements.
- 1's complement can be formed by changing 1's to 0's and 0's to 1's
- 2's complement of a number is done by Taking the 1's complement and adding 1 to the least significant bit in the number.

11

11

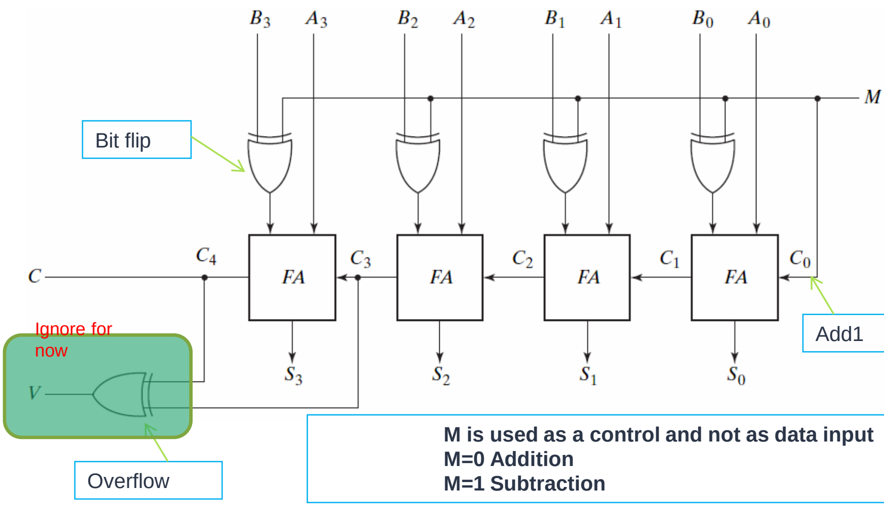
Binary Adder/Subtractor

- How can we implement subtraction?
  - Subtraction is addition of complement
  - *N – M = N + (two's complement of M)*
- How do we determine 2's complement?
  - *1's complement (flip bits) and add 1*
- How can we flip bits?
  - NOT gate (subtraction only)
  - XOR gate (to provide control: Add/Sub):
    - *$x \oplus 0 = x$ (use for Add)*
    - *$x \oplus 1 = x'$ (use for Sub)*
  - How can we add 1?
    - Input carry

12

12

Binary Adder/Subtractor



M is used as a control and not as data input
M=0 Addition
M=1 Subtraction

13

13

Binary Adder/Subtractor

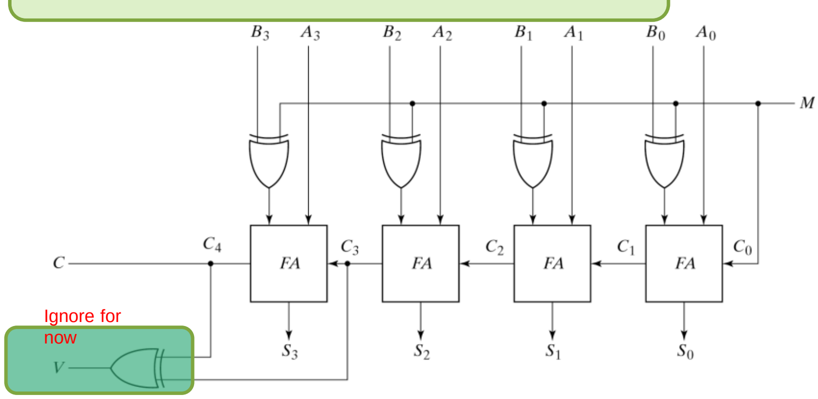
M controls the operation (addition or subtraction)

$M = 0$ → Adder: *A* plus *B*, $C_0 = 0$

$$B \otimes 0 = (B \cdot 1) + (B' \cdot 0) = B$$

$M = 1$ → Adder: *A* plus 2's complement and a $C_0 = 1$ → $A - B$

$M = 1$: $$B \otimes 1 = (B \cdot 0) + (B' \cdot 1) = B'$$



14

14

Overflow

- *n-bit addition can generate (n+1)-bit number*
  - Resulting in "overflow"
  - Problem: Needs to be detected by computer system
- How can we detect overflow?
- For unsigned numbers
  - End carry out of most significant position
- For signed numbers
  - Most significant bit indicates sign
  - If carry into sign position and out of sign position differ, then overflow
  - Detected by XOR gate

15

15

Overflow

8 bits can hold signed numbers: -128 to + 127

	1	1						
+80	0	1	0	1	0	0	0	0
- 70	1	0	1	1	1	0	1	0
+10	0	0	0	0	1	0	1	0

	0	0						
+70	0	1	0	0	0	1	1	0
- 80	1	0	1	1	0	0	0	0
- 10	1	1	1	1	0	1	1	0

Overflow cannot occur if one number is positive and other is negative

16

16

Overflow

8 bits can hold signed numbers: -128 to + 127

	carries	0 1
+70		0 1 0 0 0 1 1 0
+80		0 1 0 1 0 0 0 0
+150		1 0 0 1 0 1 1 0

	carries	1 0
- 70		1 0 1 1 1 0 1 0
- 80		1 0 1 1 0 0 0 0
- 150		0 1 1 0 1 0 1 0

Overflow may occur when both numbers are positive or both are negative

17

17

Overflow

	carries	1 1
+80		0 1 0 1 0 0 0 0
- 70		1 0 1 1 1 0 1 0
+10		0 0 0 0 1 0 1 0

	carries	0 0
+70		0 1 0 0 0 1 1 0
- 80		1 0 1 1 0 0 0 0
- 10		1 1 1 1 0 1 1 0

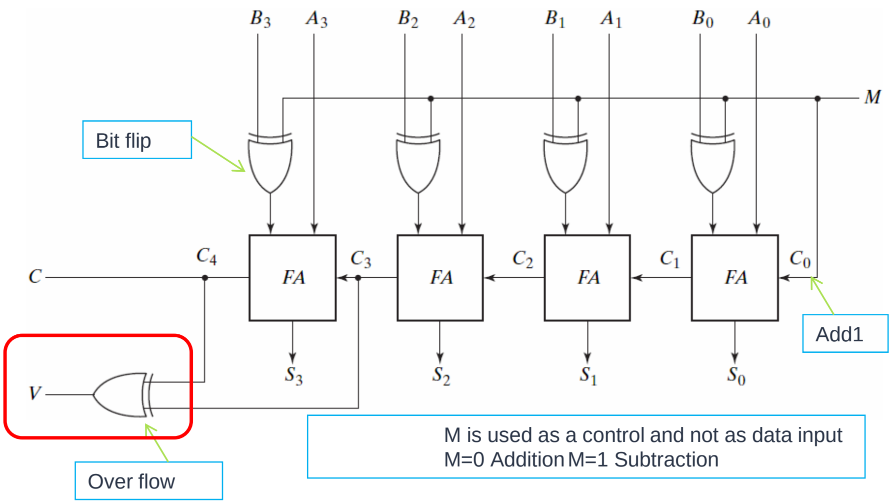
carries	0 1	
	0 1 0 0 0 1 1 0	+70
	0 1 0 1 0 0 0 0	+80
	1 0 0 1 0 1 1 0	+150

carries	1 0	
	1 0 1 1 1 0 1 0	- 70
	1 0 1 1 0 0 0 0	- 80
	0 1 1 0 1 0 1 0	- 150

18

18

Overflow



19

19

BCD Adder

20

BCD Adder

Recal: Conversion & Coding

$(12)_{10}$

Conversion **1100**

Coding (using BCD code for each digit) **00010010**

BCD Adder

Design a circuit that calculates the Arithmetic addition of two decimal digits.

$$\begin{array}{cr} & 9 \\ + & 3 \\ \hline 1 & 2 \end{array}$$

carry

- Maximum sum is 9+9 + 1 = 19

Max digit

Carry from previous digits

BCD Adder

- Full adder ➔ (1+1) Bit + 1 Carry= 3 bits
- BCD➔(4+4) Bits + 1 Carry= 9 bits input
- Max output is 9+9+1=19
- Use 4 bit binary adder
  - Input 2 BCD numbers
  - Sum will be in binary form
  - Output binary number from 0 to 19
  - AIM: Convert Binary back to BCD

23

23

BCD Adder

BCD Sum

Number	C	S8	S4	S2	S1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1

Upto 9, the sum of Binary Adder and BCD Adder is same

24

24

BCD Adder

BCD Sum

Number	C	S8	S4	S2	S1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1
16	1	0	1	1	0
17	1	0	1	1	1
18	1	1	0	0	0
19	1	1	0	0	1

25

25

BCD Adder

BCD adder sum

Number	C	S8	S4	S2	S1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1
16	1	0	1	1	0
17	1	0	1	1	1
18	1	1	0	0	0
19	1	1	0	0	1

Binary sum

K	Z8	Z4	Z2	Z1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1

26

26

BCD Adder

BCD adder sum

Number	C	S8	S4	S2	S1
10	1	0	0	0	0
11	1	0	0	0	1
12	1	0	0	1	0
13	1	0	0	1	1
14	1	0	1	0	0
15	1	0	1	0	1
16	1	0	1	1	0
17	1	0	1	1	1
18	1	1	0	0	0
19	1	1	0	0	1

+6

Binary sum

K	Z8	Z4	Z2	Z1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1

27

27

BCD Adder

- If sum is up to 9
  - Use the regular Adder.
- If the sum > 9
  - Use the regular adder and add 6 to the result

28

28

BCD Adder

If sum exceeds binary 9 then represent it as two separate binary numbers

Add binary 6(0110)

Binary Sum					BCD Sum					Dec
K	Z8	Z4	Z2	Z1	C	S8	S4	S2	S1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

29

29

BCD Adder – When is the Sum > 9

Binary sum

Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

N	K	Z8	Z4	Z2	Z1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1

C = K +

30

30

BCD Adder – When is the Sum > 9

Binary sum

Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

N	K	Z8	Z4	Z2	Z1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1

C = K + Z8*Z4+

31

31

BCD Adder – When is the Sum > 9

Binary sum

Number	K	Z8	Z4	Z2	Z1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1

N	K	Z8	Z4	Z2	Z1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1

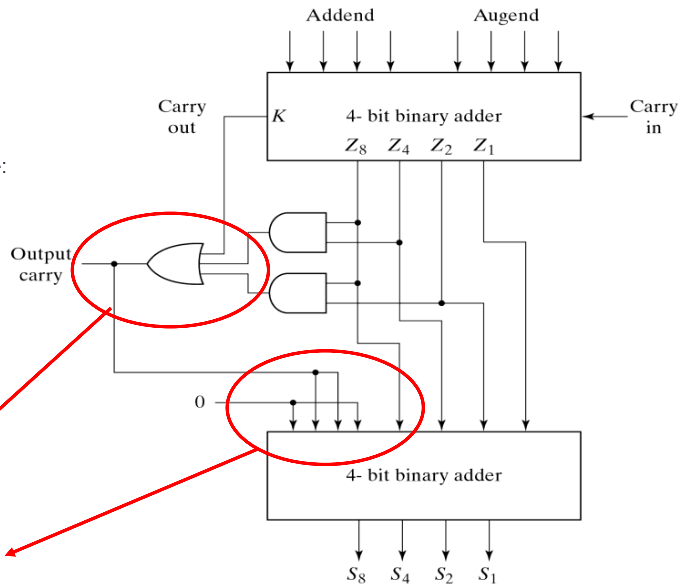
C = K + Z8*Z4 + Z8*Z2

32

32

BCD Adder

Numbers that need correction (add 6) are:
01010 (10)
01011 (11)
01100 (12)
01101 (13)
01110 (14)
01111 (15)
10000 (16)
10001 (17)
10010 (18)
10011 (19)



Decides to add 6?

Adds 6

Fig. 4-14 Block Diagram of a BCD Adder

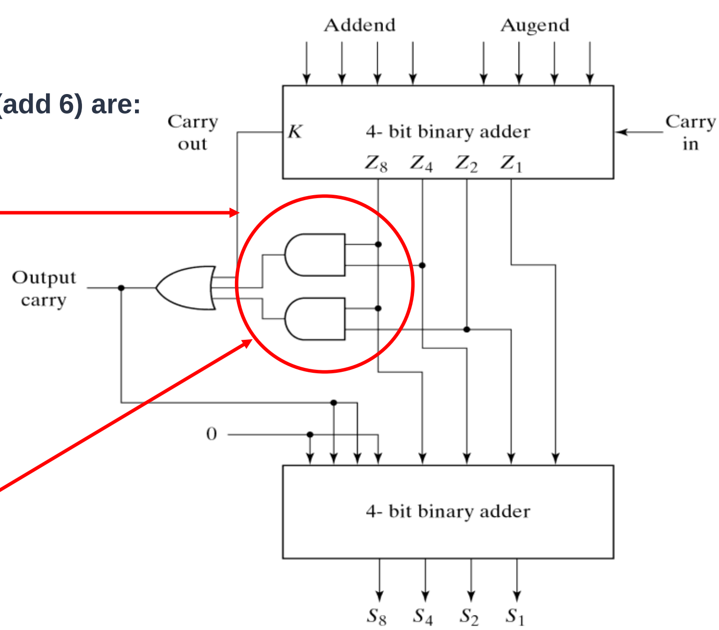
33

33

BCD Adder

Numbers that need correction (add 6) are:

K	Z8	Z4	Z2	Z1	
0	1	0	1	0	(10)
0	1	0	1	1	(11)
0	1	1	0	0	(12)
0	1	1	0	1	(13)
0	1	1	1	0	(14)
0	1	1	1	1	(15)
1	0	0	0	0	(16)
1	0	0	0	1	(17)
1	0	0	1	0	(18)
1	0	0	1	1	(19)



C = K + $\boxed{\text{Z8Z4 +Z8Z2}}$

Fig. 4-14 Block Diagram of a BCD Adder

34

34

Binary Multiplier

- Multiplication of binary numbers is done in the same way as decimal numbers
- Multiplicand *B* is multiplied by the multiplier *A* starting from the LSB.
- Successive partial products are shifted one position from the left and the final product is obtained from the sum of partial products.

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$

35

35

Binary Multiplier

2-Bit by 2-Bit Binary Multiplier

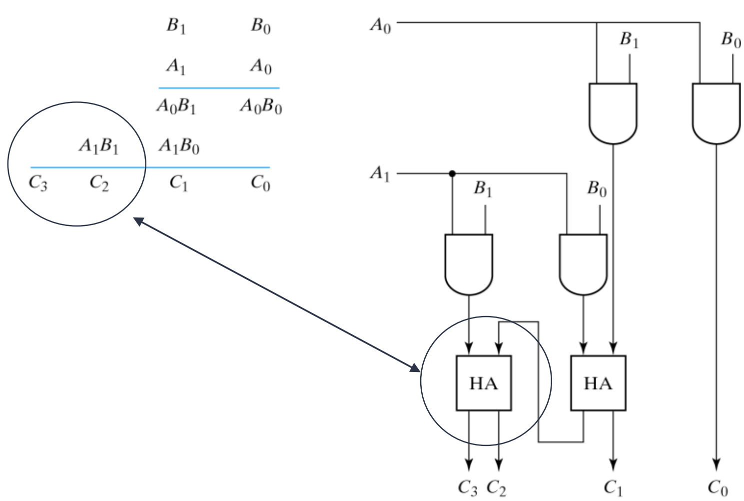


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

36

36

Binary Multiplier

B3 B2 B1 B0
A2 A1 A0

A0B3 A0B2 A0B1 A0B0

A1B3 A1B2 A1B1 A1B0

A2B3 A2B2 A2B1 A2B0

J x *K*

J x *K* AND gates
(*J*-1) *K*-bit adders
Result: *J* + *K* bits

4-Bit by 3-Bit Binary Multiplier

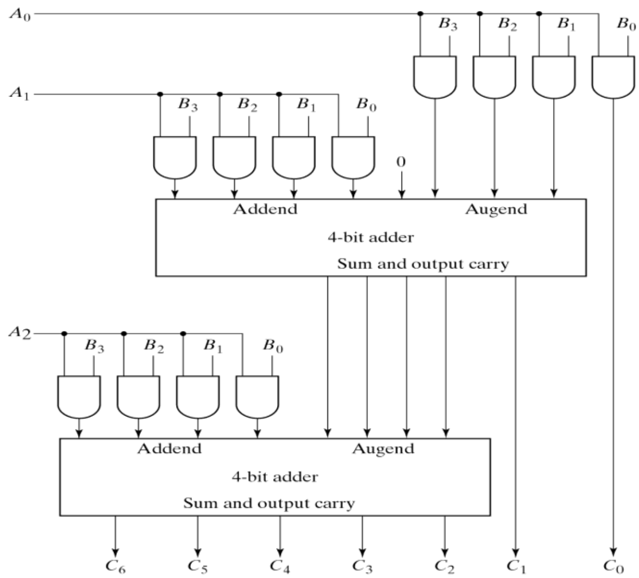


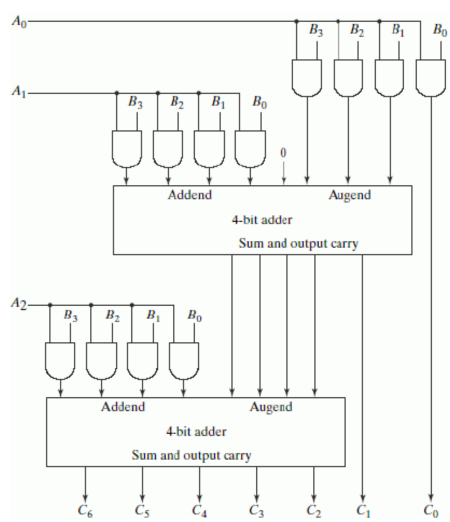
Fig. 4-16 4-Bit by 3-Bit Binary Multiplier

37

37

Multiplier – last Class Queries

- 4-bit x 3-bit multiplier
  - J=3, K=4
- How many AND gates
  - J x K gates
- How many adders
  - (J-1) K-bit adders
- How many output bits
  - J + K bits



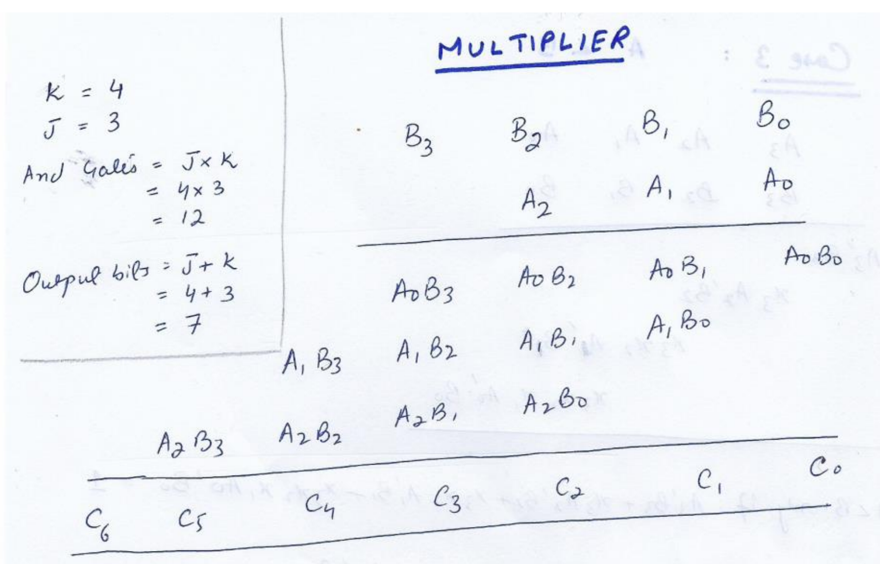
38

Practice Exercises

- Design a 3-bit by 4-bit multiplier
- Design a 4-bit by 4-bit multiplier

39

39



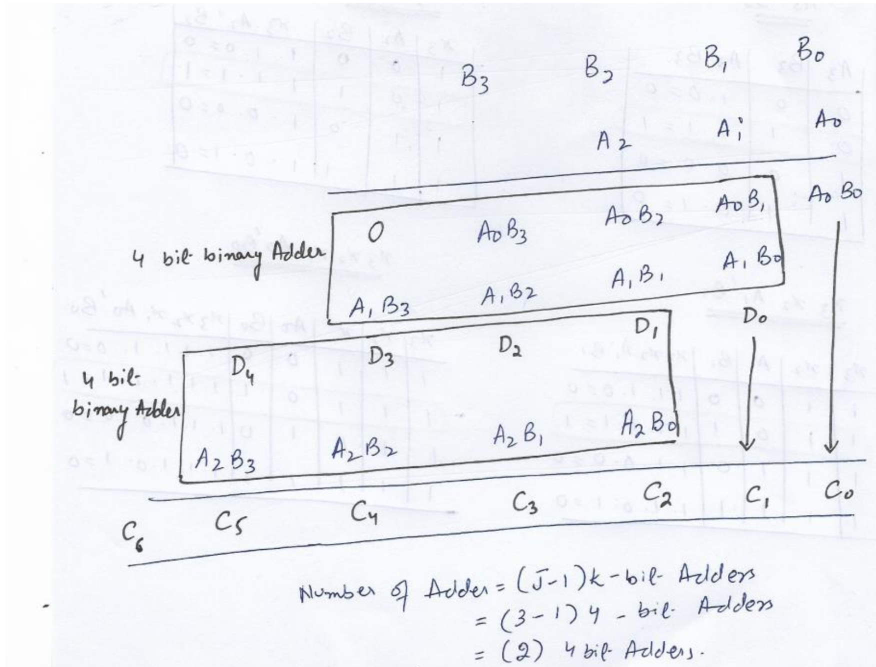
MULTIPLIER

K = 4
J = 3
And Gates = J × K
= 4 × 3
= 12

Output bits = J + K
= 4 + 3
= 7

			B_3	B_2	B_1	B_0
				A_2	A_1	A_0
			A_0B_3	A_0B_2	A_0B_1	A_0B_0
		A_1B_3	A_1B_2	A_1B_1	A_1B_0	
	A_2B_3	A_2B_2	A_2B_1	A_2B_0		
C_6	C_5	C_4	C_3	C_2	C_1	C_0

40



41

References

- Chapter 4 – Digital Design Morris Mano
- Template is taken from slides carnival.

42