CO20-320241

Computer Architecture and Programming Languages

CAPL

Lecture 9

Dr. Kinga Lipskoch

Fall 2019

BCD Addition (1)

- If the sum of each decimal digit is less than 9 then the operation is the same as normal binary addition
- If the sum of each decimal digit is greater than 9 then a binary 6 is added, this will always cause a carry
- 4 bits are used per digit
- BCD subtraction – a more complicated operation – will not be discussed here

BCD Addition (2)

- Seems to work just as normal binary addition

```
   5       ______     (BCD for 5)
  +4      +______     (BCD for 4)
  --       ======
   9       ______     (BCD for 9)
```

- Another example

```
   45       ______   ______     (BCD for 45)
  +33      +______   ______     (BCD for 33)
  ---       ======   ======
   78       ______   ______     (BCD for 78)
```

- Both examples do not create carries

BCD Addition (3)

- Seems to work just as normal binary addition

```
  5   0101     (BCD for 5)
 +4  +0100     (BCD for 4)
 --  -----
  9   1001     (BCD for 9)
```

- Another example

```
 45     0100  0101    (BCD for 45)
+33    +0011  0011    (BCD for 33)
---    -----  ----
 78     0111  1000    (BCD for 78)
```

- Both examples do not create carries

BCD Addition (4)

- But does not work if sum > 9

```
  6   0110    (BCD for 6)
 +7  +0111    (BCD for 7)
 ──  ─────
 13   1101    invalid code group

           0110     (BCD for 6)
          +0111     (BCD for 7)
          ─────
           1101     invalid sum
           0110     add 6 for correction
      0001 0011     (BCD for 13)
```

- 0110 is added to the invalid sum, produces carry

Hexadecimal Arithmetic (1)

Hex addition:

- Add the hex digits in decimal
- If the sum is 15 or less then express it directly in hex digits
- If the sum is greater than 15 then subtract 16 and carry 1 to the next position

```
  58
+ 4B
 ___
 ___
```

Hexadecimal Arithmetic (2)

- Hex subtraction – use the same method as for binary numbers (hex $\rightarrow$ binary $\rightarrow$ 2's complement $\rightarrow$ hex)
- A quicker method to get 2's complement is illustrated below
- Find 2's complement for 73A:

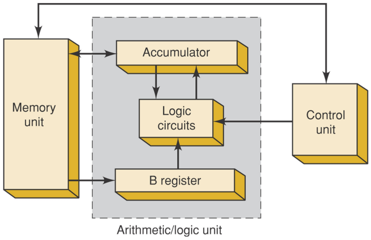
```
  F    F    F
 -7   -3   -A
  8    C    5
           +1
 ------------
  8    C    6
```

Hexadecimal Representation of Negative Numbers

- Negative number: Highest bit is 1 (binary)
- Positive number: Highest bit is 0 (binary)
- If the MSD (Most Significant Digit) in a hex number is 8 or greater then the number is negative
- If the MSD is 7 or less then the number is positive

Arithmetic Circuits (1)

- An arithmetic/logic unit (ALU) accepts data stored in memory and executes arithmetic and logic operations as instructed by the control unit
- Contains at least two flip-flop registers:
  - B register
  - Accumulator register

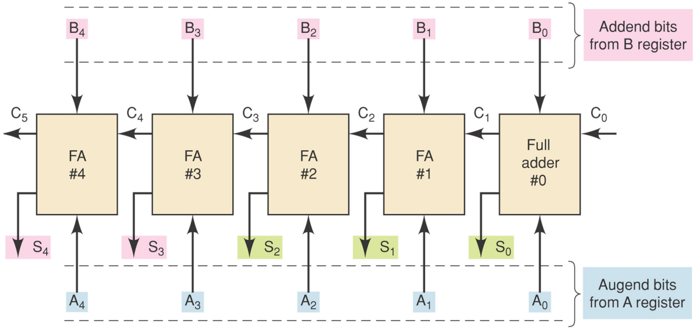


Arithmetic Circuits (2)

- Typical sequence of operations:
  - Control unit is instructed to add a specific number from a memory location to a number stored in the accumulator register
  - The number is transferred from memory to the B register
  - Number in the B register and accumulator register are added in the logic circuit, with sum sent to the accumulator for storage
  - The new number remains in the accumulator for further operations or can be transferred to memory for storage
- Register accumulates the sums when performing successive additions

Parallel Binary Adder

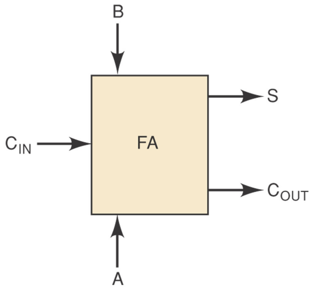
- The A and B variables represent two 5-bit numbers to be added
- The C variables are the carries
- The S variables are the sum bits



Sum appears at S_4, S_3, S_2, S_1, S_0 outputs.

Design of a Full Adder

- Construct a truth table of 3 inputs (2 numbers to be added and carry in) and 2 outputs (sum and carry out)
- Parallel adder: all bits of the augend and addend are present and are fed into the adder circuit simultaneously



Design of a Full Adder: Truth Table

Augend bit input	Addend bit input	Carry bit input	Sum bit output	Carry bit output
A	B	C_{IN}	S	C_{OUT}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Two outputs, circuitry needs to be designed for each individually

Desgin of a Full Adder: The Logic Circuit

Augend bit input	Addend bit input	Carry bit input	Sum bit output	Carry bit output
A	B	C_{IN}	S	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \overline{A}\ \overline{B}\ C_{IN} + \overline{A}\ B\ \overline{C_{IN}} + A\ \overline{B}\ \overline{C_{IN}} + A\ B\ C_{IN}$$
$$S = \overline{A}\ (\overline{B}\ C_{IN} + B\ \overline{C_{IN}}) + A\ (\overline{B}\ \overline{C_{IN}} + B\ C_{IN})$$
$$S = \overline{A}\ (B \oplus C_{IN}) + A\ (\overline{B \oplus C_{IN}})$$

Sum Bit Output

$$S = \overline{A}\,(B \oplus C_{IN}) + A\,(\overline{B \oplus C_{IN}})$$

Note $X = B \oplus C_{IN}$

$$S = \overline{A} \cdot X + A \cdot \overline{X} = A \oplus X$$

$$S = A \oplus (B \oplus C_{IN})$$

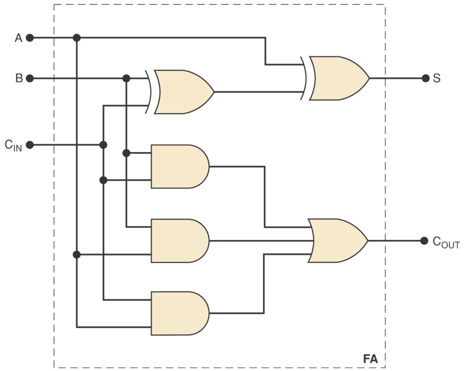
Carry Bit Output

$$C_{OUT} = \overline{A}\ B\ C_{IN} + A\ \overline{B}\ C_{IN} + A\ B\ \overline{C_{IN}} + A\ B\ C_{IN}$$

Use $A\ B\ C_{IN}$ three times, since it has common factors with each other terms

$$\begin{aligned} C_{OUT} &= B\ C_{IN}(\overline{A} + A) + A\ C_{IN}(\overline{B} + B) + A\ B(\overline{C_{IN}} + C_{IN}) \\ &= B\ C_{IN} + A\ C_{IN} + A\ B \end{aligned}$$

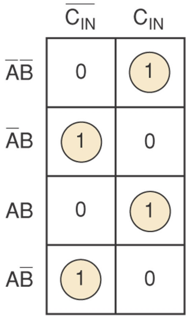
Logic Circuit for Full Adder



$$S = A \oplus (B \oplus C_{IN})$$

$$COUT = BC_{IN} + AC_{IN} + AB$$

K-maps for the Full Adder Outputs

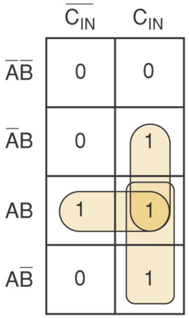


	$\overline{C}_{IN}$	C_{IN}
$\overline{A}\overline{B}$	0	1
$\overline{A}B$	1	0
AB	0	1
$A\overline{B}$	1	0

K map for S

$$S = \overline{A}\overline{B}C_{IN} + \overline{A}B\overline{C}_{IN} + ABC_{IN} + A\overline{B}\overline{C}_{IN}$$

(a)



	$\overline{C}_{IN}$	C_{IN}
$\overline{A}\overline{B}$	0	0
$\overline{A}B$	0	1
AB	1	1
$A\overline{B}$	0	1

K map for C_{OUT}

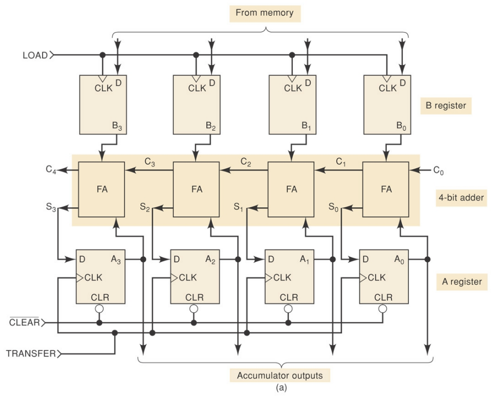
$$C_{OUT} = BC_{IN} + AC_{IN} + AB$$

(b)

Complete Parallel Adder With Registers (1)

- Register notation to indicate the contents of a register we use brackets:
  $[A] = 1011$ is the same as $A_3 = 1$, $A_2 = 0$, $A_1 = 1$, $A_0 = 1$
- A transfer of data to or from a register is indicated with an arrow
  $[B] \rightarrow [A]$ means the contents of register B have been transferred to register A
  This is a common form of notation

Complete Parallel Adder With Registers (2)

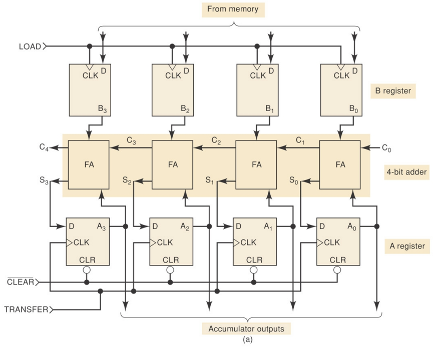


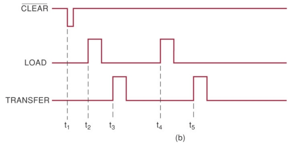
- Augend bits A_3 through A_0 are stored in the accumulator $[A]$
- Addend B_3 through B_0 in $[B]$
- Contents of $[A]$ is added to $[B]$ by four FAs and sum is produced at outputs S_3 through S_0
- C_4 carry of fourth FA, can be used as overflow or as C_{IN} for fifth FA
- Sum outputs connected to D inputs so sum can be stored in $[A]$
- $[A]$ can be transferred elsewhere

Complete Parallel Adder With Registers (3)

The following describes adding binary 1001 and 0101 using the circuit on right

- A CLEAR pulse is applied at t_1
- The first binary number 1001 is transferred from memory to $[B]$ at t_2
- The sum of 1001 and 0000 is transferred to the $[A]$ at t_3
- 0101 is transferred from memory to the $[B]$ at t_4
- The sum outputs are transferred to the $[A]$ at t_5
- The sum of the two numbers is now present in the accumulator



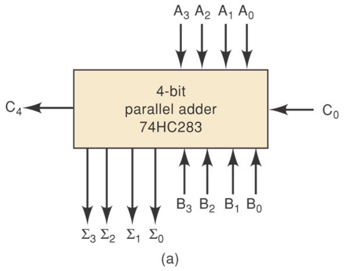


Carry Propagation

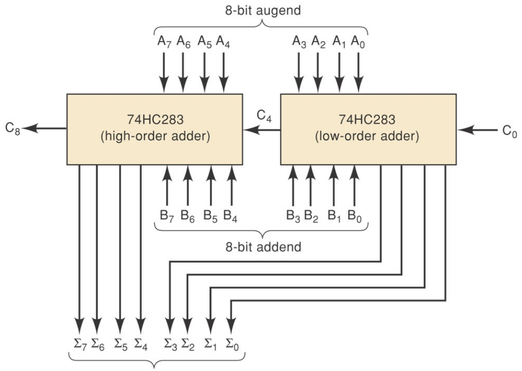
- Parallel adder speed is limited by carry propagation (also called carry ripple)
- Carry propagation results from having to wait for the carry bits to “ripple” through the device
- Additional bits will introduce more delay
- Various techniques have been developed to reduce the delay
- The look-ahead carry, using logic gates to look at lower order bits of augend and addend to see if higher-order carry is to be generated, is commonly used in high speed devices

Integrated Circuit Parallel Adder (1)

- The most common parallel adder is a 4 bit device with 4 interconnected FAs and look-ahead carry circuits
- The A and B lines each represent 4 bit numbers to be added
- The C_0 is the carry in, the C_4 is the carry out, and the $\sum$ lines are the sum of the 2 numbers
- $\sum$ symbol (greek for sigma) is used for S outputs


A_3 A_2 A_1 A_0
4-bit parallel adder 74HC283
C_4
C_0
B_3 B_2 B_1 B_0
Σ_3 Σ_2 Σ_1 Σ_0
(a)

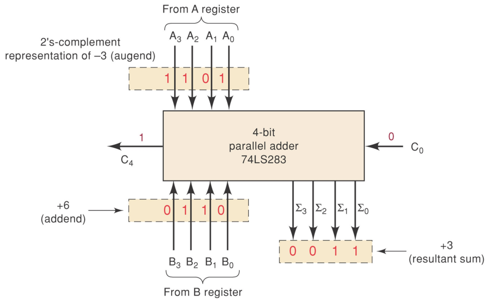
Integrated Circuit Parallel Adder (2)


8-bit augend
A_7 A_6 A_5 A_4
A_3 A_2 A_1 A_0
74HC283 (high-order adder)
74HC283 (low-order adder)
C_8
C_4
C_0
B_7 B_6 B_5 B_4
B_3 B_2 B_1 B_0
8-bit addend
Σ_7 Σ_6 Σ_5 Σ_4 Σ_3 Σ_2 Σ_1 Σ_0

(b)

Parallel adders may be cascaded together as shown to add larger numbers, in this case two 8 bit numbers

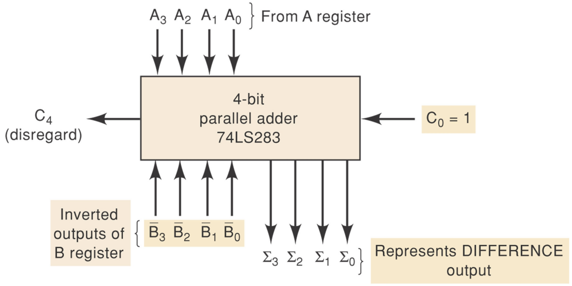
2's Complement System



An adder can be used to perform addition and subtraction by designing a way to take the 2's complement for subtraction as described in figure

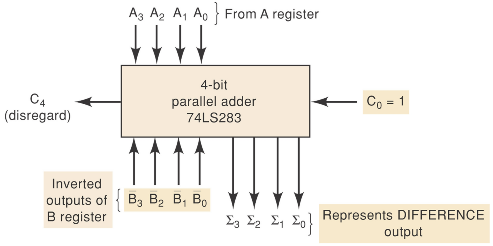
- Addition of negative and positive numbers using adders is done by placing the negative number into 2's complement form and performing normal addition
- Subtraction is done by converting the number to be subtracted (subtrahend) to 2's complement and adding to the minuend

Subtraction with a Parallel Adder


A_3 A_2 A_1 A_0
From A register
C_4 (disregard)
4-bit parallel adder 74LS283
$C_0 = 1$
Inverted outputs of B register
$\overline{B}_3$ $\overline{B}_2$ $\overline{B}_1$ $\overline{B}_0$
Σ_3 Σ_2 Σ_1 Σ_0
Represents DIFFERENCE output

- Parallel adder used to perform subtraction $(A - B)$ using the 2's-complement system
- The bits of the subtrahend (B) are inverted, and $C_0 = 1$ to produce the 2's complement

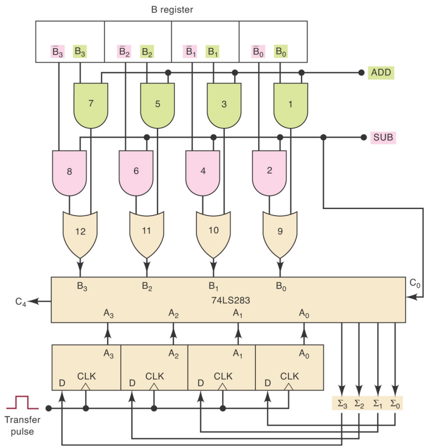
Subtracting 6 from 4


A_3 A_2 A_1 A_0
From A register
4-bit parallel adder 74LS283
C_4 (disregard)
$C_0 = 1$
Inverted outputs of B register
$\overline{B}_3$ $\overline{B}_2$ $\overline{B}_1$ $\overline{B}_0$
Σ_3 Σ_2 Σ_1 Σ_0
Represents DIFFERENCE output

$|A| = 0100 \qquad 4_{10}$
$|B| = 0110 \qquad 6_{10}$
B is inverted and fed with carry $C_0 = 1$ into adder

```
   1   C0
0100   |A|
1001   |B|
1110   |S| = |A| - |B|
```

Combined Circuit to Perform Addition or Subtraction (1)


B register
$\bar{B}_3$ B_3 $\bar{B}_2$ B_2 $\bar{B}_1$ B_1 $\bar{B}_0$ B_0
ADD
SUB
7
5
3
1
8
6
4
2
12
11
10
9
B_3 B_2 B_1 B_0
C_0
C_4
74LS283
A_3 A_2 A_1 A_0
D CLK
Transfer pulse
Σ_3 Σ_2 Σ_1 Σ_0

- Controlled by two control signals ADD and SUB
- Circuits like adder/subtractor are used in computers
- S output lines are usually then transferred into the accumulator
- Accomplished by TRANSFER pulse to the CLK input of register A

Combined Circuit to Perform Addition or Subtraction (2)

Addition

- ADD $= 1$, SUB $= 0$
- SUB $= 0$ inhibits AND gates 2, 4, 6, 8 holding their outputs at 0
- ADD $= 1$ enables AND gates 1, 3, 5, 7 allowing outputs to pass to Bx levels
- B_0 to B_3 pass through OR gate into adder
- Sum appears at the outputs S_0 to S_3, SUB $= 0$ causes carry $C_0 = 0$

Subtraction

- ADD $= 0$ and SUB $= 1$
- ADD $= 0$ inhibits AND gates 1, 3, 5, 7
- SUB $= 1$ enables AND gates 2, 4, 6 and 8 so outputs pass the B_0, B_1, B_2, B_3 levels
- $\overline{B_0}$, $\overline{B_1}$, $\overline{B_2}$, $\overline{B_3}$ pass through OR gates into adder to be added
- $C_0 = 1$, B now contains 2's complement, difference appears at the outputs S_0 to S_3