Arithmetic Circuits II

CS207 Lecture 12

James YU

May 6, 2020

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Subtraction

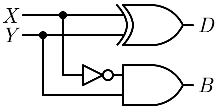
- Subtraction is the other basic function of arithmetic operations of information-processing tasks of digital computers.
  - $0 - 0 = 0$,
  - $0 - 1 = 1$ with borrow of 1,
  - $1 - 0 = 1$,
  - $1 - 1 = 0$.
  - The first, third, and fourth operations produce a subtraction of one digit, but the second operation produces a difference bit as well as a *borrow* bit.
- A combinational circuit that performs the subtraction of two bits as described above is called a *half-subtractor*.
- the subtraction operation involves three bits — the *minuend* bit, *subtrahend* bit, and the *borrow* bit, and produces a different result as well as a borrow.
- The combinational circuit that performs this type of addition operation is called a *full-subtractor*.

Half-subtractor

- As described above, a half-subtractor has two inputs and two outputs.
- Let the input variables minuend and subtrahend be designated as X and Y, and output functions be designated as D for difference and B for borrow.

Input variables		Output variables	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- $D = X \oplus Y$,
- $B = X'Y$.



Full-subtractor

- A combinational circuit of full-adder performs the operation of subtraction of three bits — the minuend, subtrahend, and borrow Z generated from the subtraction operation of previous significant digits and produces the output difference and borrow.

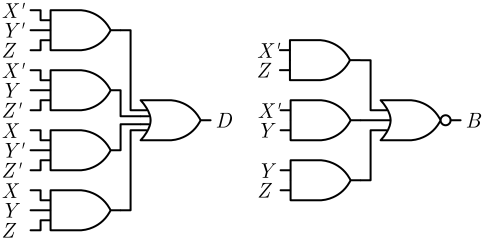
Input variables			Output variables	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full-subtractor

$X \backslash YZ$	00	01	11	10
0		1		1
1	1		1	

$X \backslash YZ$	00	01	11	10
0		1	1	1
1			1	

- $D = X'Y'Z + X'YZ' + XY'Z' + XYZ$,
- $B = X'Z + X'Y + YZ$.

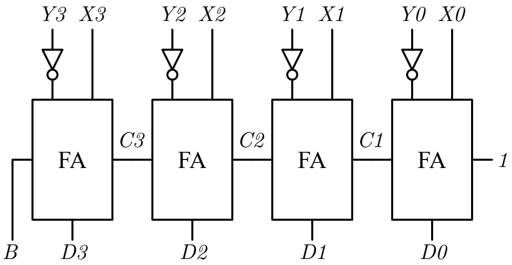


Binary subtractor

- The subtraction of unsigned binary numbers can be done most conveniently by means of complements.
  - The subtraction $A - B$ can be done by taking the 2's complement of B and adding it to A.
  - The 2's complement can be obtained by taking the 1's complement and adding 1 to the least significant pair of bits.
  - The 1's complement can be implemented with inverters, and a 1 can be added to the sum through the input carry.

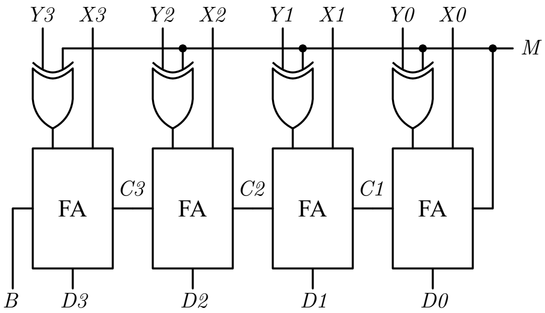
Binary subtractor

- The circuit for subtracting $A - B$ consists of an adder with inverters placed between each data input B and the corresponding input of the full adder.



Binary subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder.



Binary subtractor

- It is worth noting that binary numbers in the signed-complement system are added and subtracted by the same basic addition and subtraction rules as are unsigned numbers.
- Therefore, computers need only one common hardware circuit to handle both types of arithmetic.

Overflow

- When two numbers with n digits each are added and the sum is a number occupying $n+1$ digits, we say that an overflow occurred.
  - When the addition is performed with paper and pencil, an overflow is not a problem, since there is no limit by the width of the page to write down the sum.
- Overflow is a problem in digital computers because the number of bits that hold the number is finite and a result that contains $n+1$ bits cannot be accommodated by an n-bit word.
  - For this reason, many computers detect the occurrence of an overflow, and when it occurs, a corresponding flip-flop is set that can then be checked by the user.
- The detection of an overflow after the addition of two binary numbers depends on **whether the numbers are considered to be signed or unsigned**.
  - When two unsigned numbers are added, an overflow is detected from the end carry out of the most significant position.
  - When two signed numbers are added, the sign bit is treated as part of the number and the end carry does not indicate an overflow.

Overflow

- An overflow may occur if the two numbers are both positive or negative.

carries:	0	1	
$+70$		0	1000110
$+80$		0	1010000
$+150$		1	0010110
carries:	1	0	
-70		1	0111010
-80		1	0110000
-150		0	1101010

- If the carry out of the sign bit position is taken as the sign bit of the result, then the nine-bit answer so obtained will be correct.
- But since the answer cannot be accommodated within eight bits, we say that an overflow has occurred.

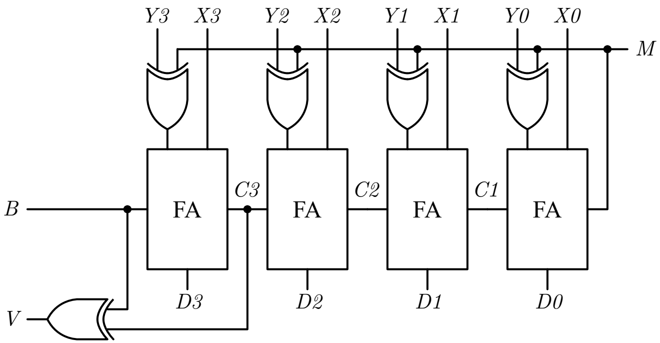
Overflow

- An overflow condition can be detected by observing the carry into the sign bit position and the carry out of the sign bit position.
- **If these two carries are not equal, an overflow has occurred.**

carries:	0	1	
$+70$		0	1000110
$+80$		0	1010000
$+150$		1	0010110
carries:	1	0	
-70		1	0111010
-80		1	0110000
-150		0	1101010

- If the two carries are applied to an exclusive-OR gate, an overflow is detected when the output of the gate is equal to 1.

Overflow



Decimal adder

- Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary coded form.
- An adder for such a computer must employ arithmetic circuits that accept coded decimal numbers and present results in the same code.
- Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage.
  - Since each input digit does not exceed 9, the output sum cannot be greater than $9 + 9 + 1 = 19$, the 1 in the sum being an input carry.
  - Suppose we apply two BCD digits to a four-bit binary adder. The adder will form the sum in binary and produce a result that ranges from 0 through 19.

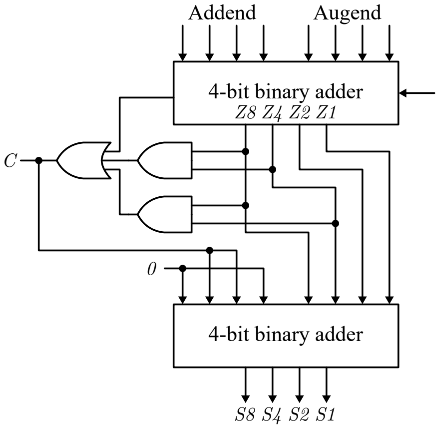
Decimal adder

- When the binary sum is equal to or less than `1001`, the result is a valid BCD code.
- When the binary sum is greater than `1001`, we obtain an invalid BCD representation. The addition of binary 6 (`0110`) to the binary sum converts it to the correct BCD representation and also produces an output carry as required.

K	$Z8$	$Z4$	$Z2$	$Z1$	C	$S8$	$S4$	$S2$	$S1$
0	1	0	1	0	1	0	0	0	0
0	1	0	1	1	1	0	0	0	1
0	1	1	0	0	1	0	0	1	0
0	1	1	0	1	1	0	0	1	1
0	1	1	1	0	1	0	1	0	0
0	1	1	1	1	1	0	1	0	1
1	0	0	0	0	1	0	1	1	0
1	0	0	0	1	1	0	1	1	1
1	0	0	1	0	1	1	0	0	0
1	0	0	1	1	1	1	0	0	1

Decimal adder

- $C = K + Z_8 Z_4 + Z_8 Z_2$.
- When $C = 1$, it is necessary to add `0110` to the binary sum and provide an output carry for the next stage.



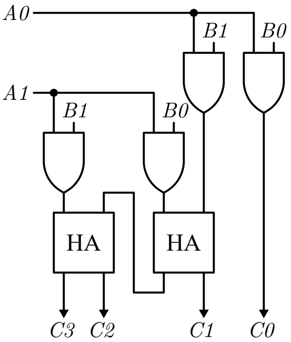
Binary multiplier

- Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers.
- The multiplicand is multiplied by each bit of the multiplier, starting from the least significant bit.
- Each such multiplication forms a partial product. Successive partial products are shifted one position to the left.

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\cline{3-4}
 & & A_0 B_1 & A_0 B_0 \\
 & A_1 B_1 & A_1 B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$

Binary multiplier

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



Binary multiplier

- A combinational circuit binary multiplier with more bits can be constructed in a similar fashion.
    - A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier.
    - The binary output in each level of AND gates is added with the partial product of the previous level to form a new partial product.

Magnitude comparator

- The comparison of two numbers is an operation that determines whether one number is greater than, less than, or equal to the other number.
- A *magnitude comparator* is a combinational circuit that compares two numbers A and B and determines their relative magnitudes.
- The outcome of the comparison is specified by three binary variables that indicate whether $A > B$, $A = B$, or $A < B$.
- Consider two numbers, A and B, with four digits each.
    - Write the coefficients of the numbers in descending order of significance:
    - $A = A_3A_2A_1A_0$,
    - $B = B_3B_2B_1B_0$.

Magnitude comparator

- The two numbers are equal if all pairs of significant digits are equal: $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, and $A_0 = B_0$.
- When the numbers are binary, the digits are either 1 or 0, and the equality of each pair of bits can be expressed logically with an exclusive-NOR function as

$$x_i = A_i B_i + A_i' B_i'.$$

- The equality of the two numbers A and B is displayed in a combinational circuit by an output binary variable that we designate by the symbol $(A = B)$.
  - This binary variable is equal to 1 if the input numbers, A and B, are equal, and is equal to 0 otherwise:

$$(A = B) = x_3 x_2 x_1 x_0.$$

Magnitude comparator

- To determine whether A is greater or less than B, we inspect the relative magnitudes of pairs of significant digits, starting from the most significant position.
- If the two digits of a pair are equal, we compare the next lower significant pair of digits. The comparison continues until a pair of unequal digits is reached.
  - If the corresponding digit of A is 1 and that of B is 0, we conclude that $A > B$.
  - If the corresponding digit of A is 0 and that of B is 1, we have $A < B$.

$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$
$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

Magnitude comparator