BCD decimal digit Addition/subtraction with carry/borrow

Yu Hen Hu

1. The design

A = a₃a₂a₁a₀ and B = b₃b₂b₁b₀: 4-bit unsigned binary number representing BCD code.

S = 0: addition, S = 1: subtraction: It applies to the operand B such that each full adder input yᵢ = S ⊕ bᵢ.

R = 1: borrow from the next digit during subtraction. R = z₄

r = 1: borrowed by the previous digit during subtraction

K = 1: carry out to the next digit during addition. K = z₄ + z₃ + z₂ + z₁

k = 1: carry in from the previous digit during addition.

c₀: carry-in of the BCD adder/subtractor. c₀ = S ⋅ k + S ⋅ r

q₃ = 1 if S = 1 and R = 1. Otherwise, = 0. That is, q₃ = SR

q₂ = 1 if S = 0 and K = 1. Otherwise, = 0. q₂ = S ⋅ K

q₁ = 1 if S = 0 and K = 1 and if S = 1 and R = 1. q₁ = S ⋅ K + S ⋅ r

q₀ = 0.

2. Explanation of the subtraction case.

When r = 1, we are compute A−B−1. However, this is evaluated as:

During subtraction, the computation carried out is (r = 0 or 1)

A−B−r = A + 2C(B) − r = A + [1C(B) + 1] − r
Since this is to perform unsigned binary number subtraction using 2's complement arithmetic, the carry out \( z_4 \) determines whether the result is positive or negative. In particular, if \( z_4 = 1 \), the result is positive, and no borrow out will occur (\( R = 0 \)). If \( z_4 = 0 \), then the result is negative, and a borrow-out will be issued (\( R = 1 \)) and the outcome \( z_i \)'s will be corrected by adding to \( 1010 \) (10), the borrowed amount.

**Example 1.**

Assume \( r = 1 \). \( A = 3, B = 7 \). We should have \( 3 - 7 - 1 + 10 = 5 \) and \( R = 1 \). To implement this, we have

\[
0011 - 0111 - 1 = 0011 + 1C(0111) + 1 - 1 = 0011 + 1000 = (0) 1011
\]

Since the carry out = 0, we have \( R = 1 \). Thus the result needs to be corrected by adding decimal number 10 (binary number 1010) to it:

Answer = 1011 + 1010 = (1) 0101

Dropping the carry out, we have the correct answer 0101.

**Example 2.**

Assume \( r = 1 \), \( A = 3 \), \( B = 3 \). We should have \( 3 - 3 - 1 + 10 = 9 \). Using 4-bit 2's complement adder/subtractor, we have

\[
0011 - 0011 - 1 = 0011 + 1C(0011)+1 - 1 = 0011 + 1100 = (0) 1111
\]

Because carry out = 0, \( R = 1 \). Thus, the result should be corrected to

\[
1111 + 1010 = (1) 1001
\]

After dropping the carry out, we have the correct answer.

**Algorithm**

Perform \( a_3a_2a_1a_0 - b_3b_2b_1b_0 - r = a_3a_2a_1a_0 + \overline{b_3b_2b_1b_0} + 1 - r = z_4z_3z_2z_1z_0 \)

Borrow forward \( R = z_4 \).

If \( R = 0 \), Answer = \( z_3z_2z_1z_0 \)

Else if \( R = 1 \), Answer = \( z_3z_2z_1z_0 + 1010 \) (with carry out discarded).