Practice Problems 2 Solutions

1. For binary numbers, the sign is given by the left most bit. When left most bit is 0, the number is positive. When the left most bit is 1, the number is negative.

In 2's complement system, the complement of an n-bit number is obtained by subtracting it from 2^n.

Based on all these observations, if we see a particular binary number as below,

$B = b_{n-1}b_{n-2}b_{n-3} \ldots\ldots b_1b_0$

$-2^{n-1}\ 2^{n-2}\ 2^{n-3} \ldots\ldots 2^0$ ---→ If we are saying that -2^{n-1} is the weight of the MSB, then we are talking about a negative binary number called -B.

Now, -B can be written as $2^n - B - 2^n$ -----→ where $2^n - B$ is the two's complement of B. If we take the same analogy in weights, for $-B = (2^n - B) - 2^n = 2^{n-1} - 2^n$ (If we take weight of MSB normally as 2^{n-1})

$= -2^{n-1}$ ----→ Hence we have that in two's complement, the magnitude can be computed as for an unsigned number, except that the weight of the MSB is -2^{n-1} instead of 2^{n-1}

2. Overflow is identified when the sign of the result is different from the sign of the two numbers added. Overflow actually occurs when a number exceeds the range of representable numbers with the given number of bits.

So, when performing a 2's complement addition,

  A3A2A1A0

\+ B3B2B1B0

If there is a carry from (A2+B2) which is termed as carry-in into the MSB bits (A3 & B3), and if that carry is 1 (let say).

Also, performing (A3+B3) "independent" of carry-in, might also give a carry. In this case, if carry(A3+B3) is 1, this carry will be added to the previous carry-in and result in a 0 as carry-out, which is different from carry-in 1. We can observe that here this scenario occurs only if A3 and B3 are of same signs, which Is when overflow is said to be occurred.

For example,

+4   0100

+7   0111

-5   1011 -> here carry out = 0, carry in from A2+B2 is 1 -> Overflow occurred. A,B are of same sign

Hence we can say that when carry in and out of the MSB is different, over flow is identified.

3. Consider an n bit 2's complement no:

$A = B_{n-1}B_{n-2}\ldots B_1B_0 \rightarrow$ (n-1)th bit represents the sign.

If A is positive $(A)_{10} = B_{n-2}.2^{n-2} + B_{n-3}.2^{n-1} + \ldots B_0.2^0$

On left-shiting the bits, ignoring the carryout and having 0 at the LSB

$B = B_{n-2}\ldots B_1B_00$ (Note A and B have same no. of bits)

$(B)_{10} = B_{n-2}.2^{n-1} + B_{n-3}.2^{n-2} + \ldots B_0.2^1 + 0 = 2.(B_{n-2}.2^{n-2} + B_{n-3}.2^{n-1} + \ldots B_0.2^0) = 2(A)_{10}$

If A is negative and $B_{n-2} = 1$, $(A)_{10} = -B_{n-1}.2^{n-1} + B_{n-2}.2^{n-2} + \ldots B_0.2^0$

$= -1.2^{n-1} + 1.2^{n-2} + \ldots B_0.2^0$

$= 2^{n-2}(-2+1) + B_{n-3}.2^{n-3} + \ldots B_0.2^0$

$= -2^{n-2} + B_{n-3}.2^{n-3} + \ldots B_0.2^0$

On left-shiting the bits, ignoring the carryout and having 0 at the LSB

$B = B_{n-2}\ldots B_1B_00$ $(B_{n-2} = 1)$

$(B)_{10} = -1.2^{n-1} + B_{n-3}.2^{n-2} + \ldots B_0.2^1 + 0 = 2.(-2^{n-2} + B_{n-3}.2^{n-3} + \ldots B_0.2^0) = 2(A)_{10}$

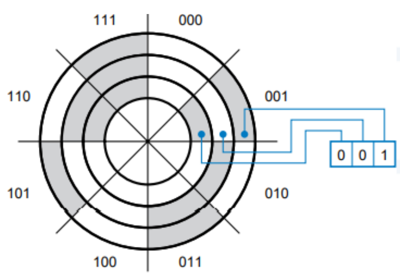
In the other case where B_{n-2} is 0, overflow occurs.

Overflow rule: If two numbers with the same sign (both positive or both negative) are added, then overflow occurs if and only if the result has the opposite sign. In other words, if left shifting changes the value of MSB, we know that overflow has occurred and the resultant no is outside of the range of nos that can be represented by the -bit 2's complementary number.

Ex: If $(6)_{10} = 0110$, on left shiting it becomes 1100, which is $(-4)_{10}$. Hence overflow has occurred.

If $(6)_{10} = 00110$, on left shifting it becomes 01100, which is $(12)_{10}$ ie twice of 6.

4.



This is Figure 2-5 as per the text book.

Here, bad boundaries mean that, when the encoder is producing values based on the black and white areas on the disc as in for black area, it outputs 1, and white area it outputs 0 as in the example in the diagram, certain boundaries will trigger an issue when the proper output cannot be obtained mechanically from the disc.

For example, consider a boundary between 001 and 010. Here 2 bits are changing while crossing the boundary. If the encoder detector is exactly on the boundary, and mechanical corrections might occur in reality, MSB is 0 always, so we don't need to care about this for this boundary atleast. Now, next two bits are 01/10. The both of these bit detector hands might touch 0 or both might touch 1, which leads to output values 000 or 011 respectively which is a false encoding. Such boundaries which cause issues in perfect encoding are called bad boundaries.

So, If we observe clearly, this issue occurs only when 2 or more bits are changing from one boundary to other. So, we can name these boundaries as bad boundaries as per our previous understanding.

Solution – (001 – 010), (100-011), (101-110), (111-000)

5. So, Here we can see that for n bit binary numbers, for each bit if 0 is there, next bit would be 1, and the next bit would be 10..and so on. So here change of 2 or more than 2 bits is occuring for every 2 numbers in the line.

Also, we know that with n bits, we can have 2^n binary numbers and each 2 of them will give one bad boundary. So, in total we have $2^n/2 = 2^{n-1}$ bad boundaries in general.

Solution : 2^{n-1}