FIT1047 Tutorial 2

Topics

- 2's complement, floating point
- Boolean logic, logic gates, logisim

Instructions

- The tasks are supposed to be done in groups. In some tasks, it might be useful to have different roles for people in the group.

Task 1: 2's complement

1.a Assume you have 8 bits to represent binary numbers. What decimal value does the binary number **10110101** have in the different notations?

(i) **unsigned binary number**

(ii) **sign-magnitude notation**

(iii) **1's complement**

(iv) **2's complement**

1.b Create a table with all possible 4-bit binary values in one column and the decimal they represent in the different notations.

1.c Assume you want to implement a bit-wise approach to add 2 n-bit 2's complement numbers x+y=z. You need to check for overflow.
Use the following notation:
$x = x_n x_{n-1} x_{n-2} ... x_1$
$y = y_n y_{n-1} y_{n-2} ... y_1$
$z = z_n z_{n-1} z_{n-2} ... z_1$
You can use a bit-wise + operator with one bit result and one bit for the carry:
$0 + 0 = 0$ with carry bit 0
$0 + 1 = 1$ with carry bit 0
$1 + 0 = 1$ with carry bit 0
$1 + 1 = 0$ with carry bit 1
For $i = 1, .., n$ the carry bit is denoted by c_i.

For such an implementation answer the following questions:

- For the actual addition, what do you need to do in steps $i = 1, ..., n$?
- In which steps do you need to check for overflow conditions? What do you need to check?

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

(a) XOR

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

(b) NAND

1.d In a (rather small) system, there is a floating point representation with one sign-bit, a 3-bit exponent and a 4-bit significand. Assume that the significand is normalized without any assumed bits (i.e. the significand always starts with "1."). The exponent uses 2's complement and exponents with all zeros and all 1s are allowed.

What is the smallest positive number and what is the largest positive number that can be stored on this system?

Task 2: Boolean logic, logic gates, logisim

2.a
- Download *logisim evolution* from Moodle
- Start logisim by clicking on the file or executing java -jar logisim-evolution.jar

2.b Make yourself familiar with logisim by doing a few very easy tasks:

- place one AND gate, 2 inputs and one output
- connect them
- poke the inputs to try simulation
- do the same with OR and NOT
- try some combinations of gates

2c In addition to AND, OR and NOT, there are a number of other Boolean logic operators. Examples are **XOR**, the exclusive or, and **NAND**, a negated AND. Their truth tables are shown in the tables above.

Build XOR and NAND in logisim only using AND, OR, and NOT gates. Use the simulation to test your result.

2d (Additional advanced task:) Use logisim and only AND, OR and NOT to build a circuit for the following function:

$F(A, B, C) = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C + ABC$

How many gates do you need? Simplify the function first, e.g. by using k-maps.