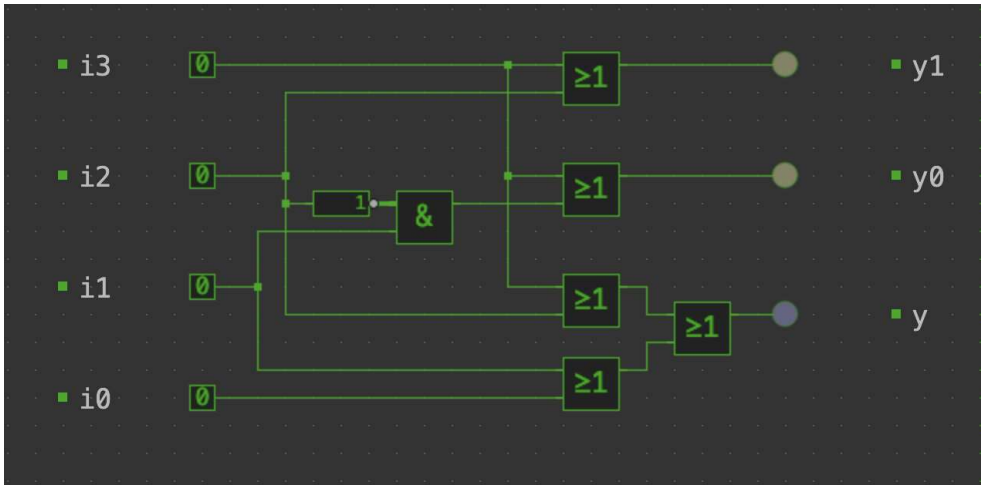
ICS 2022 Problem Sheet #8

Problem 8.1: digital circuit analysis

You are given the following digital circuit.



a) Write down the truth table defining the outputs y_0, y_1, and y.

$y_0 = i3 \vee (\neg i2 \wedge i1)$
$y_1 = i3 \vee i2$
$y = (i0 \vee i1) \vee (i2 \vee i3)$

I3	I2	I1	¬i2	(¬i2 ∧ i1)	y_0
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
1	0	0	1	0	1
0	1	1	0	0	0
1	1	0	0	0	1
1	0	1	1	1	1
1	1	1	0	0	1

I3	I2	y_1
0	0	0
0	1	1
1	0	1
1	1	1

I3	I2	I1	I0	(i0 ∨ i1)	(i2 ∨ i3)	y
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	1	1
1	0	0	0	0	1	1
0	1	1	0	1	1	1
1	1	0	0	0	1	1
1	0	1	0	1	1	1
1	1	1	0	1	1	1
0	0	0	1	1	0	1
0	0	1	1	1	0	1
0	1	0	1	1	1	1
1	0	0	1	1	1	1
0	1	1	1	1	1	1
1	1	0	1	1	1	1
1	0	1	1	1	1	1
1	1	1	1	1	1	1

b) Write down short boolean expressions defining y_0, y_1, and y.

$y_0 = i3 \vee (\neg i2 \wedge i1)$
$y_1 = i3 \vee i2$
$y = (i0 \vee i1) \vee (i2 \vee i3)$

c) Describe in your own words what the circuit is doing and how it might be used.

The circuit is a priority encoder, that compresses multiple binary inputs into a smaller number of outputs. The output of a priority encoder is the binary representation of the index of the most significant activated line, starting from zero. They are often used to control interrupt requests by acting on the highest priority interrupt input.

Priority encoders can be easily connected in arrays to make larger encoders, such as one 16-to-4 encoder made from six 4-to-2 priority encoders - four 4-to-2 encoders having the signal source connected to their inputs, and the two remaining encoders take the output of the first four as input. The priority encoder is an improvement on a simple encoder circuit, in terms of handling all possible input configurations.

Problem 8.2: dice display

Too many students waiting inside the coffee bar to obtain drinks and snacks was found to be problematic and as a consequence the number of people waiting to be served got limited to seven.You got the task to create a display showing how many students are inside and you decided to build a display out of light emitting diodes (LEDs) that can be powered by a very tiny solar panel. Your display resembles the form of a dice with LEDs positioned as follows:

Your display is driven by three input lines x_2, x_1, x_0 indicating a binary number.

a) Write a truth table defining the necessary boolean functions.

I started by writing the 8 numbers in binary, using three bits, that will represent the three input lines x_2, x_1, x_0.

	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0
2	0	1	0	0	0	1	0	1	0	0
3	0	1	1	0	0	1	1	1	0	0
4	1	0	0	1	0	1	0	1	0	1
5	1	0	1	1	0	1	1	1	0	1
6	1	1	0	1	1	1	0	1	1	1
7	1	1	1	1	1	1	1	1	1	1

b) Provide (simple) boolean expressions for the boolean functions.

$a = g = x_2$

$b = f = (x_2 \wedge x_1 \wedge \neg x_0) \vee (x_2 \wedge x_1 \wedge x_0) =$

$= (x_2 \wedge x_1) \wedge (\neg x_0 \vee x_0) =$

$= (x_2 \wedge x_1) \wedge 1 =$

$= x_1 \wedge x_2$

$c = e = (\neg x_2 \wedge x_1 \wedge \neg x_0) \vee (\neg x_2 \wedge x_1 \wedge x_0) \vee (x_2 \wedge \neg x_1 \wedge \neg x_0) \vee (x_2 \wedge \neg x_1 \wedge x_0) \vee$
$(x_1 \wedge x_2 \wedge \neg x_0) \vee (x_1 \wedge x_2 \wedge x_0) =$

$= (\neg x_2 \wedge x_1) \wedge (\neg x_0 \vee x_0) \vee (x_2 \wedge \neg x_1) \wedge (\neg x_0 \vee x_0) \vee (x_1 \wedge x_2) \wedge (\neg x_0 \vee x_0) =$

$= (\neg x_2 \wedge x_1) \wedge 1 \vee (x_2 \wedge \neg x_1) \wedge 1 \vee (x_2 \wedge x_1) \wedge 1 =$

$= (\neg x_2 \wedge x_1) \vee (x_2 \wedge \neg x_1) \vee (x_2 \wedge x_1) =$

$= (\neg x_2 \wedge x_1) \vee x_2 \wedge (x_1 \vee \neg x_1) =$

$= (\neg x_2 \wedge x_1) \vee x_2 \wedge 1 =$

$= (\neg x_2 \vee x_2) \wedge (x_1 \vee x_2) =$

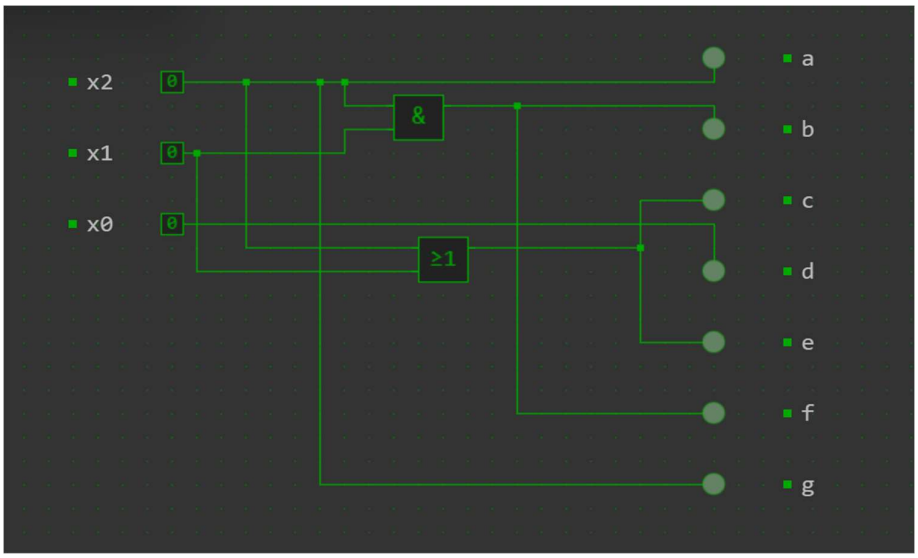
$= 1 \wedge (x_1 \vee x_2) =$

$= x_1 \vee x_2$

$d = x_0$

c) Create a digital circuit using https://simulator.io/.

https://simulator.io/board/KFze8pskJF/1



Problem 8.3: decimal to binary and binary to decimal (haskell)

Implement a functions to convert decimal numbers into binary notation and back.

a) Implement a function dtob :: Int -> String converting a non-negative integer number into a String (consisting of the characters '0' and '1') representing the integer number as a binary number. It is not necessary to handle negative integers in a meaningful way.

```
dtob :: Int -> String
dtob 1 = "1"
dtob 0 = "0"
dtob n
    | even n = dtob(n `div` 2) ++ "0"
    | otherwise = dtob(n `div` 2) ++ "1"
```

b) Implement a function dtob :: String -> Int converting a String (consisting of the characters '0' and '1') representing a binary number into the corresponding non-negative integer number. It is not necessary to handle unexpected strings in a meaningful way.

```
btod :: String -> Int
btod "0" = 0
btod "1" = 1
btod xs
    | head xs == '1' = 2 ^ (length xs - 1 ) + btod(tail xs)
    | otherwise = btod(tail xs)
```