Digital Electronics Practical 4 - Karnaugh Maps & Adders (week 5)

Previous Next

Solutions will be uploaded at the end of week 8.

SimCir

You can practice the techniques introduced in the lectures using SimCir (https://kazuhikoarase.github.io/simcirjs/)to simulate circuit creation.

Karnaugh Maps question

Before you start: Make sure you have completed work from last time in the DE Practical 3 item! Also, check the solutions to Digital Electronics' practical 3. If you don't understand anything - ask!

Objectives: The main objective is to enhance your understanding of Karnaugh maps and how to use them to simplify Boolean formulae. As always, it is important that you confirm your understanding by showing your work to a demonstrator.

Exercise:

Create a circuit which takes a 4-bit binary input, given by toggle switches, and displays a one digit '7-segment display' output.

1. Follow the approach in lectures to create a truth table for each segment, turn it into a Karnaugh map and produce a minimised Boolean formula.
2. In SimCir arrange seven LEDs in the appropriate pattern, and four toggle switches to control the input.
3. Connect up the toggle switches according to your given formula, and check that the display is correct for 0-9.

Food for thought... (and if you have time, work it out after you finish the rest of the questions below on Adders!)

- How many gates have you used for each segment? How many gates would be used if you simply took the Sum of Products expression for each segment and turned it into

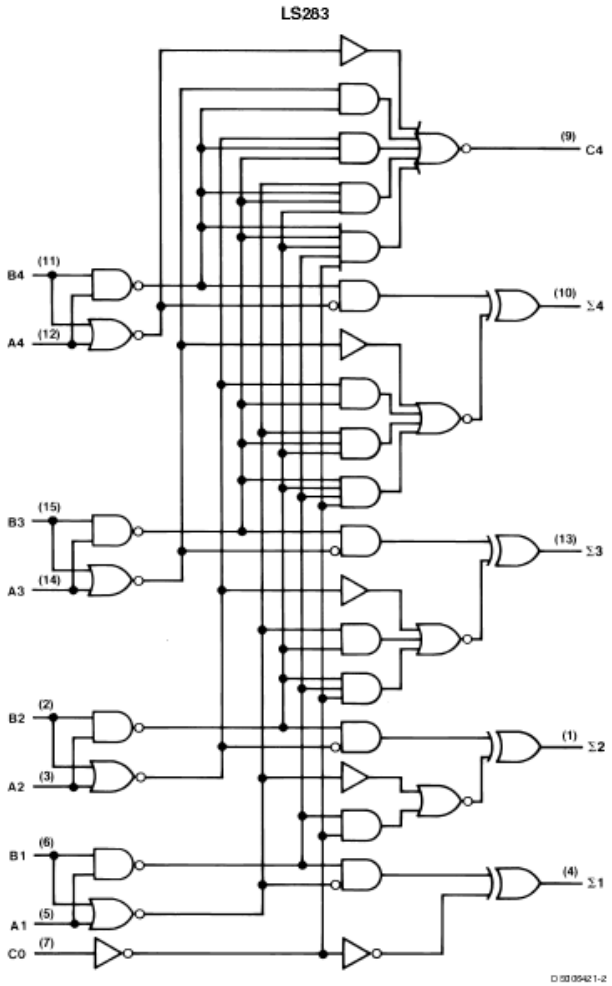
circuitry?

- Now you wish to make efficient use of the binary input - we want inputs corresponding to 10-15 to give the appropriate hexadecimal symbol (A, b, C, d, E, F). This will remove any 'doesn't matters' in the Karnaugh maps. Rework the circuitry to display correctly for inputs 0-15.
- If you wished to make a multi-digit display, how easy would it be to use your decimal circuitry to create one for 00-99 or 000-999? How easy would it be to use your hexadecimal circuitry to create one for 00-FF or 000-FFF etc.?

Adders questions

Exercise 1.

Take the circuit diagram of the MSI 74x283 chip below, which is a 4-bit adder. Set the input C0 A1A2A3A4 B1B2B3B4 to 0 1011 0111. On paper, follow through the logic gates and check the output is what you would expect. Recall that a circle at the input (or output) to a gate inverts the value coming in (or out).



Exercise 2.

Build a simple 4-bit ALU, as shown in the lecture on circuits. Look through the lecture notes and reconstruct the necessary pieces. Put them together to make a simple ALU.

Please do try this for yourself, but if you need help there are some instructions below. The instructions serve also as a solution to the exercise so you can check if you got everything right by verifying whether you have done the steps below.

1. Open this version of SimCir (https://kazuhikoarase.github.io/simcirjs/).
2. You will first need to create the modules that will be put together in the ALU.
3. Using SimCir we will create 4-bit NOT.
    a. Click 'Create new circuit' to start a new circuit.
    b. Drag 4 inputs ('In') and 4 outputs ('Out') into the workspace.
    c. Label these A0 to A3 and Y0 to Y3, or similar.
    d. Connect each Ai to Yi via a NOT gate.
    e. If you are happy with it, save it as a chip.
4. Repeat to create a 4-bit AND and a 4-bit OR (each done bitwise). These will require 8 inputs each, and 4 outputs.
5. Create a 4-bit 2-1 MUX. This requires 9 inputs - A0-3, B0-3 and S - and 4 outputs. Look at the single bit MUX in the lecture notes - you will replicate it for 4-bits.
6. Create a 4-bit binary adder. You may want to do this by first creating a single bit full adder, saving it as a chip and then combining 4 of these. Don't forget we will use the carry in, so include it in the inputs. If you are feeling like some extra fun, make it 4-bit Carry Look-Ahead adder.
7. Now it is time to put everything together. You will have to lay it out neatly to get it to fit in the workspace.
    f. Create 4 toggle switches for input A, 4 for input B and 2 for the switches C0 and C1. Connect these to power sources. If you want, you can add some LEDs or 7-segment displays to show the current setting of the input.
    g. Drag in the 4-bits Adder, AND gate and OR gate you have created.
    h. Connect input A to the A inputs of each.
    i. Connect input B to the B inputs of the AND and OR.
    j. Drag in a 4-bit NOT and a MUX.
    k. Connect input B to the NOT and to the MUX.
    l. Connect the output of the NOT to the other input on the MUX.
    m. Connect output of the MUX to the Adder.
    n. Use two more MUXs to filter the outputs of AND, OR and Adder down to one - connect this to some LEDS or a 7-seg display.
    o. Now connect the selectors (C0 and C1) to the MUXs and the carry in to choose the correct operations. (C0 connects to the first MUX, carry in and the AND/OR MUX, C1 to the final MUX.)
    p. It should all work - test it and see if you get the outputs you expect. If not, check that the carry in is 1 when you have NOT B as the input to the Adder - this is easy to get the wrong way round.