
Name **SID #**

EEM16/CSM51A (Fall 2017)

Logic Design of Digital Systems

Prof. Ankur Mehta : mehtank@ucla.edu

Problem set 2 | **Assigned Monday Oct. 16, 2017**
Show all work. | **due 4pm Monday Oct. 30, 2017**

Instructions

This homework is to be done individually. You may consult with others to share thoughts and ideas, but all of your submitted work must be yours alone. Be sure to indicate with whom you've collaborated and in what manner.

You may use any tools or refer to published papers, books, or course notes. You're allowed to make use of online tools such as Logisim, WolframAlpha, etc., provided you properly cite them in the space below.

You must submit this cover sheet plus all pages of your solutions based on the procedure below. Please write clearly and neatly — if we cannot easily decipher what you have written, you will get zero credit.

Submission procedure

You need to submit your solution online at Gradescope:

`https://gradescope.com/`

Please see the following guide from Gradescope for submitting homework. You will need to upload a PDF and mark where each question is answered.

`http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf`

Collaborators

Identify with whom you've collaborated and in what manner, if any.

Online resources

Identify which online tools you've used, if any.

1 Five input adder

Each neuron in our neural network computes a weighted sum of each neuron in the previous layer along with a bias term. Since our input and hidden layers each consist of 4 neurons each, we will be adding together 5 values within each neuron. We can condense the necessary circuit by building a custom addition module to handle this task.

1.1 5+2 full adder

With 5 input bits to be summed, we will need 2 carry bits (*why?*). Design and analyze an optimized FA5 gate that takes in 5 one-bit inputs and 2 one-bit carry-ins and computes a sum bit and two carry-out bits. Assume any single CMOS gate (with any number of inputs) has a constant propagation delay t_{PD}.

1.1(a). Clearly draw and label an implementation of FA5 using only four standard 2+1 full adder (FA) blocks.

1.1(b). Recall the propagation delay of FA is $3t_{PD}$. What is the propagation delay of FA5 (in terms of t_{PD})?

1.2 Ripple-carry adders

Our numerical summands will each be n=16 bits wide, but for the sake of conciseness in this pset, we will consider only n=4 bits. We can add together five 4-bit input values by stringing together full adder blocks into ripple-carry adders.

1.2(a). How many bits should the sum have to prevent overflow?

1.2(b). Draw and label a block diagram of a single 4-bit five-input ripple-carry adder by connecting four FA5 blocks and two FA blocks (*note: HA blocks would also work*).

1.2(c). What is the propagation delay of the resulting adder? How many FA blocks did we use total?

1.2(d). Recall the propagation delay of a two input n-bit ripple-carry adder is $3nt_{PD}$. If we had instead generated the sum by serially adding each subsequent 4-bit input using the standard two input ripple-carry adder, what would the propagation delay have been? How many FA blocks were needed?

1.3 Optimization (extra credit)

We don't need to wait for an entire addition operation to complete before starting on the next sum; since the lowest order bits are computed first in a ripple-carry adder, we can add onto those as soon as they are ready. Also note that the carry and sum outputs in an FA have different propagation delays. Come up with a topology of FA modules that computes the sum of five 4-bit inputs as quickly as possible. What is the propagation delay of this adder?

2 Maximum index

The output layer of our neural network will generate 26 likelihood values, one for for each possible letter. In order to determine the decoded letter, we need to identify which neuron has the highest value.

2.1 Full comparator

We will start by comparing two 8-bit unsigned integers in 4 segments of 2 bits each. Design a full comparator FCMP2 that takes in two 2-bit input values $A = A_1A_0$ and $B = B_1B_0$ and a 1-bit carry E, and outputs a single bit 0 if $A > B$, 1 if $A < B$, and E if $A == B$.

2.1(a). Draw a Karnaugh map for FCMP2. On this map, identify all prime implicants, and indicate which (if any) are essential prime implicants.

2.1(b). Write a boolean expression for a minimal implementation for FCMP2.

2.1(c). Is this implementation lenient? Why or why not?

2.2 Half comparator

If, instead of passing through the carry E, we **don't care** what the output is when $A == B$, we can create a simpler circuit for the resulting half comparator HCMP2.

2.2(a). Draw the resulting 4 input Karnaugh map.

2.2(b). Use that to generate a minimal boolean expression for HCMP2.

2.2(c). Build this circuit out of 12 MOSFETS using one 2-transistor CMOS inverter and one 10-transistor CMOS gate.

2.3 Arg max

We can combine four FCMP2 blocks to compare two 8-bit values (similar to a ripple-carry adder) to identify which is larger. Using that result as the select of two muxes, we can build the module MAM to extract the maximum of the two values (max function) and the label associated with that value (argmax function). This block can then be nested to decode our desired letter.

2.3(a). Combine four FCMP2 blocks into an 8-bit comparator. Draw the resulting block diagram.

2.3(b). Use the output of that comparator along with two muxes to build the MAM max/argmax block. Draw the resulting block diagram.

2.3(c). Build a tree of these MAM blocks to determine the 5-bit value identifying the letter with the maximum likelihood value given the 26 8-bit likelihood values. Draw the resulting block diagram.

3 Display

Once we have identified the decoded letter as a 5-bit value, we would like to display it. To do so, we can use a 7-segment LED display as shown in Fig. 1. For invalid addresses, we will display a dash (i.e. only segment G illuminated).

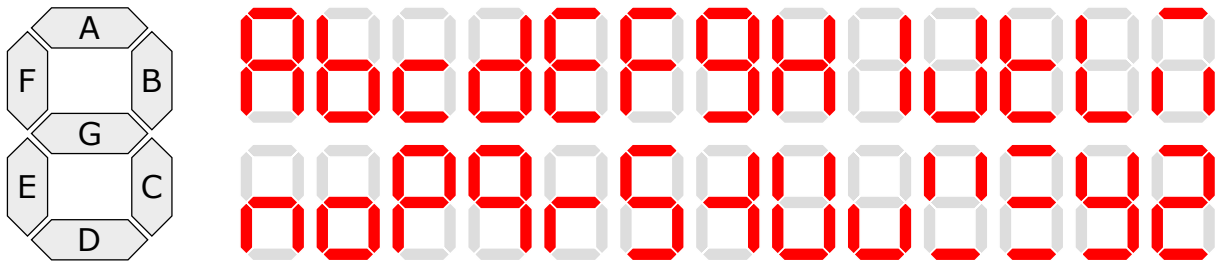

Figure 1: 7 segment display alphabet.

3.1 ROM

The desired state of the 7 segment display can be stored in a ROM as the 7-bit value $GFEDCBA$, with a 1 bit indicating that the corresponding segment is illuminated.

3.1(a). How many rows and columns must this ROM have?

3.1(b). Create the lookup table for this ROM with addresses and data as hexadecimal values.

3.2 Muxes

The 7-bit display value can also be generated using muxes with the 5-bit letter value as the select line.

3.2(a). If we were to have one mux per segment in the display, what size muxes would we need?

3.2(b). Draw the corresponding mux implementations for segments A and G.

3.2(c). If we only had 8-1 and 4-1 muxes, what is the minimum number of muxes we would need per segment in the display?

3.2(d). Draw the corresponding mux implementations for segments A and G.

4 Your turn

It's often said that you don't truly understand a subject until you can teach it. What was a topic that you struggled with so far in this class? Write and solve a pset problem that sheds light on this particular topic.