| Name | SID # |
|---|---|

# EEM16/CSM51A (Fall 2017)

## Logic Design of Digital Systems

Prof. Ankur Mehta : mehtank@ucla.edu

**Problem set 4** | **Assigned Monday Nov. 27, 2017**
**Extra credit** | **due 4pm Wednesday Dec. 6, 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 Faster multiplication

The neuron in lab 3 problem 3.1 uses our sequential multiplier from lab 2 problem 2.1; this takes (up to) 8 clock cycles to compute the product of two 8-bit numbers. This is fine if a new set of inputs comes no more than once every 8 clock cycles. However, if, for example, we would like our "dit" length to be 1 clock cycle, a stream of 'E' characters would result in a new set of inputs every 4 clock cycles. We will explore options for designing a multiplier with faster throughput.

In the following problems, follow the rules for single-clock logic: you may use any combinational logic blocks along with any number of d-registers all driven by the same clock, but no other memory elements or feedback. Also, recall that according to these rules, all outputs to sequential blocks must be generated directly as outputs to d-registers.

## 1.1 Combinational multiplier

1.1(a). Recall our implementation of a combinational multiplier using AND gates and Full Adder blocks. Assuming a $t_{PD}$ of 2ns per AND gate, 3ns per FA block, and 3ns for a d-register, along with a setup time of 6ns for the d-register, what clock period would we need to be able to use our 8-bit multiplier in a pipelined circuit? What is the latency and throughput of this multiplier?

1.1(b). Design a k-pipeline for this combinational multiplier to increase the throughput. Draw the resulting block diagram with lines indicating where the pipeline registers should go. (*Hint: see the pre-lecture video for Week 9 Wednesday*)

1.1(c). What is the resulting latency and throughput of the pipelined combinational multiplier?

## 1.2 Parallel multipliers

1.2(a). Design a **distribute** module according to the rules for single-clock logic that takes in a one-bit flag (indicating `new` or `reset`) that pulses high for one clock cycle (e.g. when a new input is ready), and alternately pulses one of two one-bit outputs $\texttt{new}_1$, $\texttt{new}_2$.

That is, the first time the input `new` pulses high, $\texttt{new}_1$ will get pulsed high, but $\texttt{new}_2$ must remain low. The very next time `new` pulses high, $\texttt{new}_2$ will get pulsed high, but $\texttt{new}_1$ must remain low. The outputs continue to alternate in that fashion.

Draw a block diagram of this module using any single-clock logic combination of wires, CMOS gates, muxes, and/or d-registers.

1.2(b). Recall that the multiplier block produces a one-bit output `ready` that is high when its complete product has been generated; assume the multiplier has been designed so that `ready` is high for exactly one clock cycle (exactly 8 cycles after its `new` input pulses). Design a **join** block that interleaves alternating outputs of two parallel multipliers onto a single output with a single one-bit `ready` output.

Draw a block diagram of this module using any single-clock logic combination of wires, CMOS gates, muxes, and/or d-registers.

1.2(c). Use the above distribute and join modules with two parallel multiplier blocks to create the fast parallel multiplier. Draw the resulting block diagram. What is the fastest latency (in terms of the clock period $t_{CLK}$) and maximum throughput (in $1/t_{CLK}$) of this system?

1.2(d). How does the area of this circuit (in terms of number of transistors) compare to the pipelined combinational multiplier above?