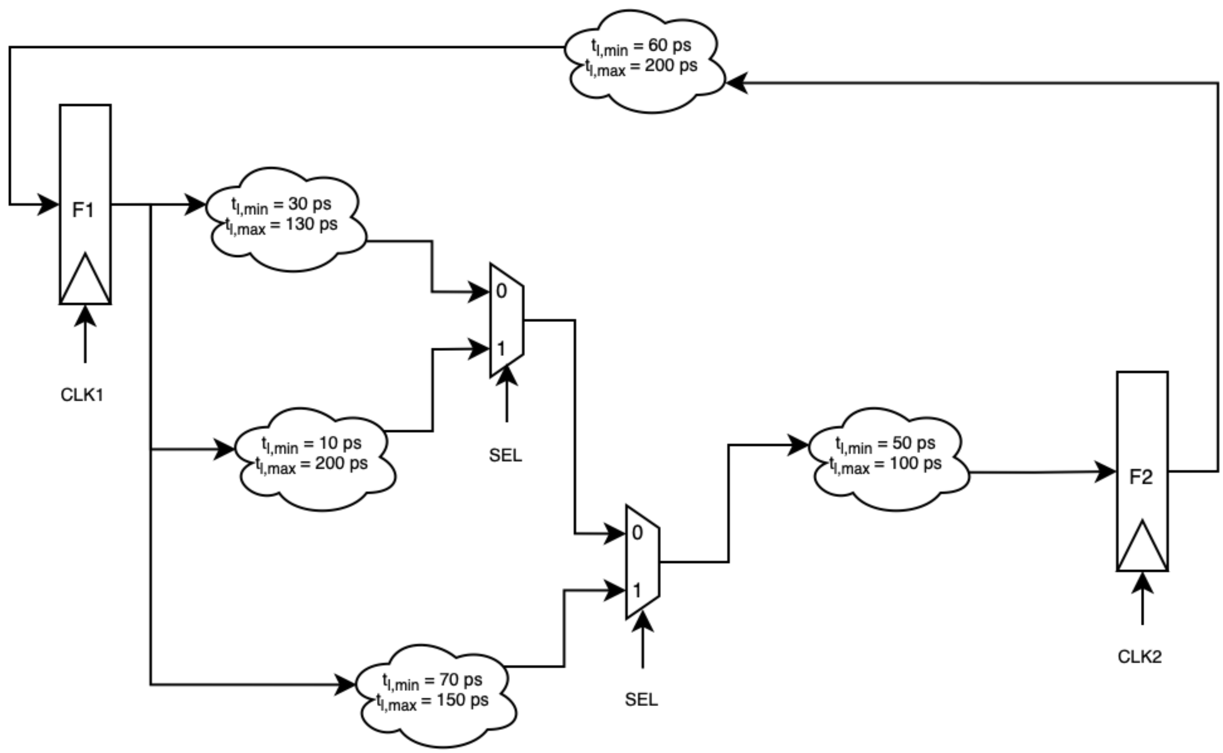
EECS 151/251A Homework 9

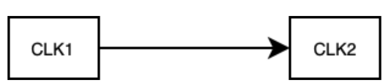
Due Friday, December 2rd, 2022 11:59PM

Problem 1: Excuses, Excuses, Ek-skew-ses ...

Consider the following circuit diagram. R1 and R2 are rising-edge triggered flip-flops. The maximum and minimum combinational delays are listed for each path. You can assume that $t_{clk\text{-}q} = 50ps$, $t_{su} = 75ps$, and $t_{hold} = 60ps$, and the multiplexers have negligible delay. SEL is a single control bit that is stable during any given clock cycle. CLK2 is a clock signal derived from CLK1 with some amount of delay, as indicated in each subpart.



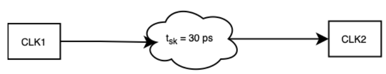
(a) Assuming that the clock source and clock tree are ideal, answer the following questions.



(i) What is the minimum clock period, T_{min}, that this circuit may operate at correctly?

(ii) Does this circuit have a hold-time violation?

(b) Considering the indicated clock skew $t_{sk} = 30ps$ in the clock tree, answer the following questions.

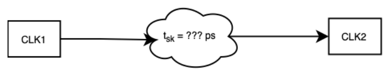


(i) What is the minimum clock period, T_{min}, that this circuit may operate at correctly?

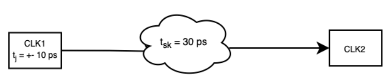
(ii) What is the hold slack for this circuit?

(iii) Does this circuit have a hold-time violation?

(c) [251A Only] You may now set the value of t_{sk}. What would you set t_{sk} to, for the shortest clock period possible? What is that T_{min}? Ignore hold-time violations for this part.



(d) Considering the indicated clock skew $t_{sk} = 30ps$ in the clock tree, and cycle-to-cycle jitter at the source $t_j = \pm 10ps$ answer the following questions.



(i) What is the minimum clock period, T_{min}, that this circuit may operate at correctly?

(ii) What is the hold slack for this circuit? (Think carefully here, hold time considers the following case: Observing some rising clock edge i for a pipeline, the value at an FF's input before i needs to be held stable for hold time past i. However, at the clock edge, the pipeline starts computing a new value. This value shouldn't reach that FF's input within hold time past the **same** clock edge i.)

(iii) Does this circuit have a hold-time violation?

Problem 2: ScRAMbling for Answers

Fresh off the blocks from 151/251A, you've decided to take on the tapeout course. Halfway through the semester, you realize that the ever-important SRAM-based memory blocks on your RV64GC SoC are not behaving correctly. Time to investigate!

(a) The spec requires a byte-addressable memory with a capacity of 64 words of storage. Additionally, the design of the memory should have a 1:1 aspect ratio. (What is the size of each word for this ISA?)

  (i) What is the total capacity of your memory in bits?

  (ii) How many rows and columns of SRAM cells should this memory have?

  (iii) What is the total capacity of your memory in bytes?

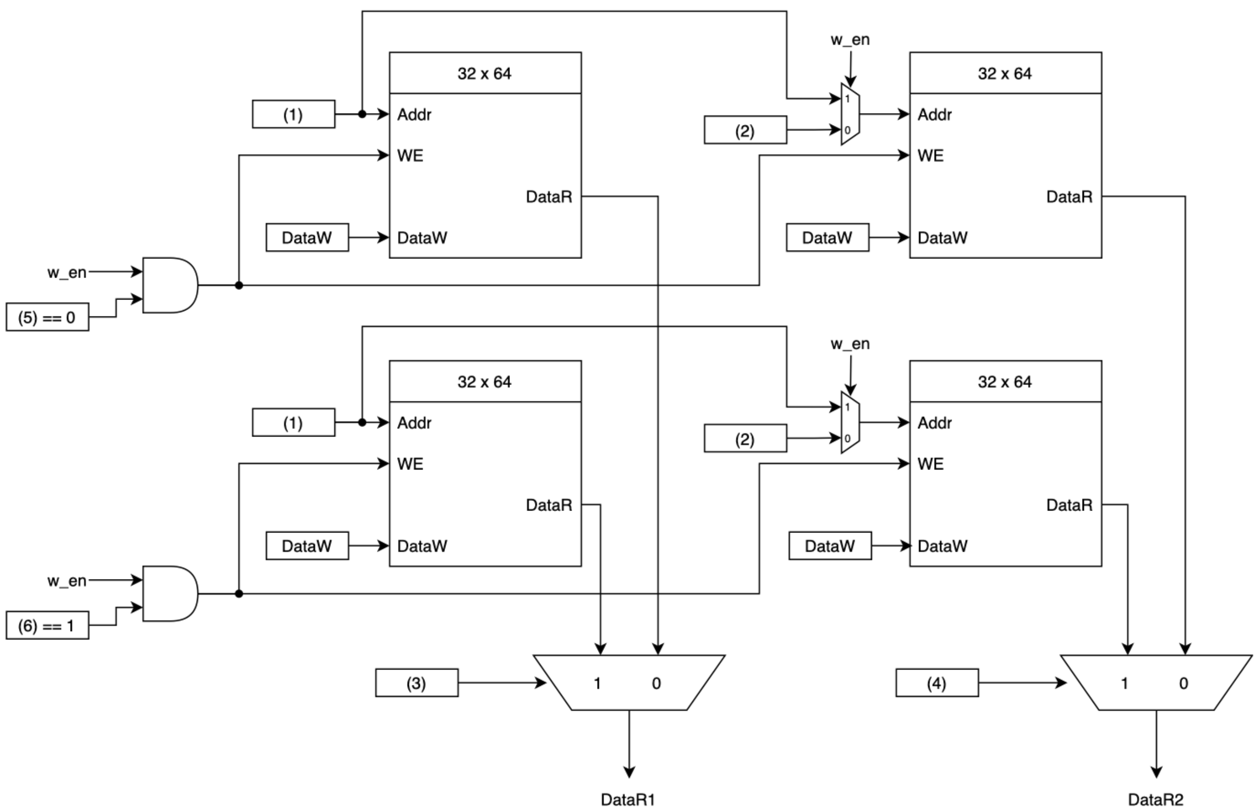
  (iv) How many total address bits do you need for this memory? How many address bits are used by the row-decoder? How many address bits are used by the column-decoder? (Keep in mind that the memory is byte-addressable. Reason about how would access a single byte from such a memory.)

(b) When you design a chip, you must do so for a particular technology - think Intel 16, TSMC 28 and so on. Physical design of SRAMs is tricky because of their large size and heavy density of bit cells. Therefore, SRAMs are generally provided as many fixed-size macros with a process development kit (PDK). To design an SRAM-based memory (that is not the same size as the macros in the PDK), you often have to combine different SRAM macro blocks together.

You are given a 32 x 64 SRAM macro with that supports 1R xor 1W at a time. You need to use multiple instances of this macro to design a 64 x 64-bit memory that supports 2R xor 1W at a time. The diagram below is a possible implementation. Using the following address signals, and possibly indexing into them, fill in the blanks:

(1) addr1rw - address used to read from port 1 or write
(2) addr2 - address used to read from port 2

Assume we follow little-endian convention, and this particular memory block is word-addressable in a larger byte-addressable memory, i.e., the lowest 3 bits of the address are used to find the right byte in the 64 bit (8 byte) word outside of this system.



(1)

(2)

(3)

(4) $\boxed{}$

(5) $\boxed{}$

(6) $\boxed{}$

(c) Noticing the high density and power consumption of the SRAMs on your SoC, you consider using 5T bit cells instead of the standard 6T bit cells we saw in discussion. After all, eliminating 1 bit line per column and 1 access transistor per bit cell can be quite beneficial. Let us analyze 5T SRAM cells further.

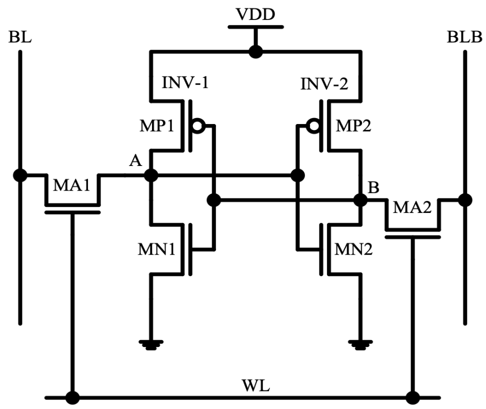


Figure 1. Circuit diagram of the standard 6T SRAM cell.

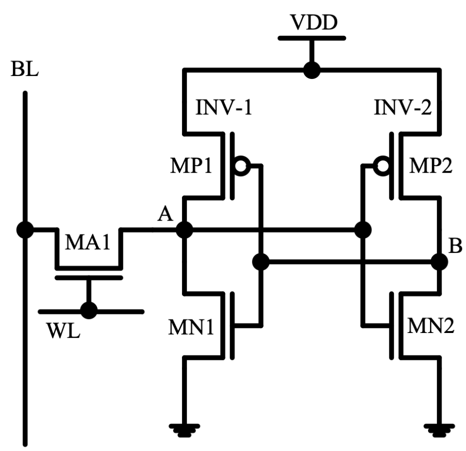


Figure 2. Circuit diagram of the traditional 5T SRAM cell.

Here $Q := A$ and $\overline{Q} := B$.

To read from a 5T SRAM cell, you pre-charge the bit line to V_{DD}. Then the word line is held high, to enable the NMOS access transistor. If $Q := 0$, the bit line will drop voltage. If $Q := 1$, the bit line will stay high.

(i) In order to guarantee read stability, what constraints exist on the relative strength of the transistors? (Think about what happens when we read a 0.)

(ii) In order to guarantee writability, what constraints exist on the relative strength of the transistors? (Think about what happens when we write a 1 and when we write a 0.)

(iii) (True or False) In this design, improving read stability directly worsens writability and vice-versa.

Given these contradictory conditions, using 5T SRAM cells reliably requires techniques that revolve around modifying the involved word line voltage, bit line voltage, and supply voltage, when reading v. writing. However, these can reduce speed and drive current of the cell, or require significant peripheral circuitry. All of this makes them difficult to use.

This problem uses the following paper for reference: Yu, Chien-Cheng, and Ming-Chuen Shiau. "Design of High Performance Single-Port 5T SRAM Cell." IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) 11.5 (2016): 14.