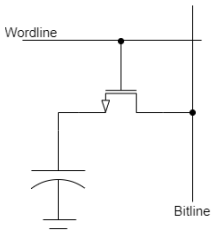
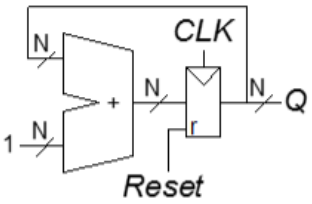
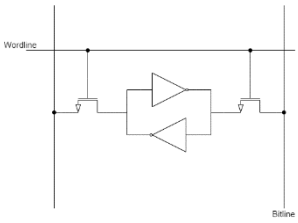
Lesson 26 – Sequential Building Blocks and Memory







UNITED STATES AIR FORCE ACADEMY

ECE 281

Lesson 26 Notes

Objectives:

1. Demonstrate the ability to analyze or design a circuit using sequential and combinational building blocks
2. Understand the importance of memory arrays and their three important characteristics
3. Be able to identify the differences between RAM and ROM
4. Be able to identify the three types of RAM and their tradeoffs
5. Demonstrate how to implement or analyze combinational logic using memory (LUT)

Review

What is the key difference between Sequential and Combinational Logic Circuits?

Combinational logic circuit – consists only of

Sequential logic circuit - consists of and

In words that we have used before, the output for a **combinational logic circuit** is

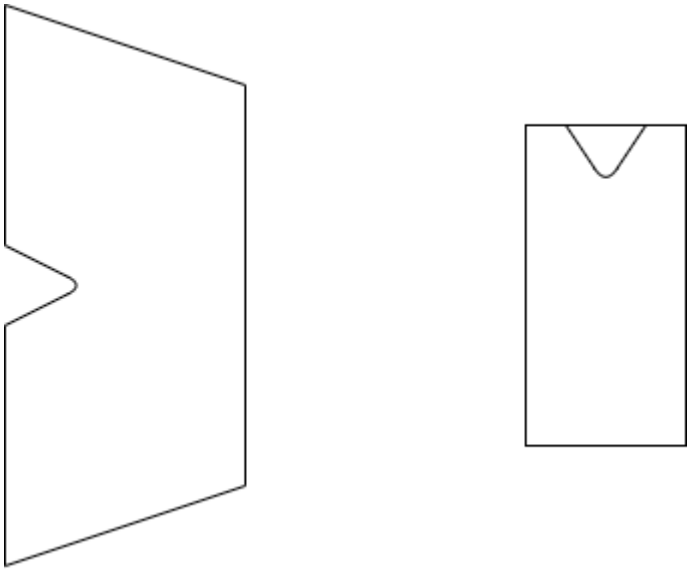
dependent only on the

Where as the output for a **Sequential logic circuit** is dependent on both the

Common Applications of Sequential Logic

In your zyBooks reading, several examples of sequential logic applications were provided, such as counters, timers and shifters. We are going to break those building blocks down on the following pages.

Counter: an N-bit counter is a logic circuit that simply increments a stored value by a specified amount. Let's analyze this below:



Question: How would you modify the circuit above to count by different values, or decrement instead of increment?

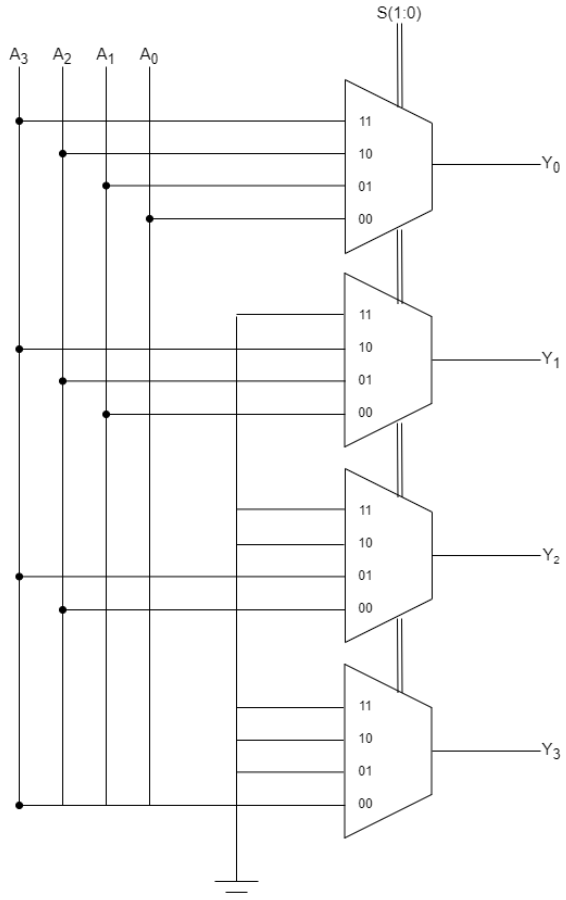
Applications of Counters:

- **Digital Clocks**
- **A to D Converters**
- **Mathematical algorithms**
- **Etc.**

Shifters and Shift Registers: in CS210, we introduced bitwise operations in C where you could execute left and right shifts of the specific binary representation of data. This was accomplished using the << (left-shift) or >> (right-shift) operator. We will now look at how to do this with hardware.

Example: The logic circuit below has 4-input bits (A_3, A_2, A_1, A_0) and 2-select bits (S1, S0).

a. Is the following circuit an example of combinational or sequential logic?
b. What is the function of this circuit?
c. How would you reverse the order of the operation



A_3	A_2	A_1	A_0	S1	S0	Y_3	Y_2	Y_1	Y_0
1	0	0	1	0	0				
1	0	0	1	0	1				
1	0	0	1	1	0				
1	0	0	1	1	1				

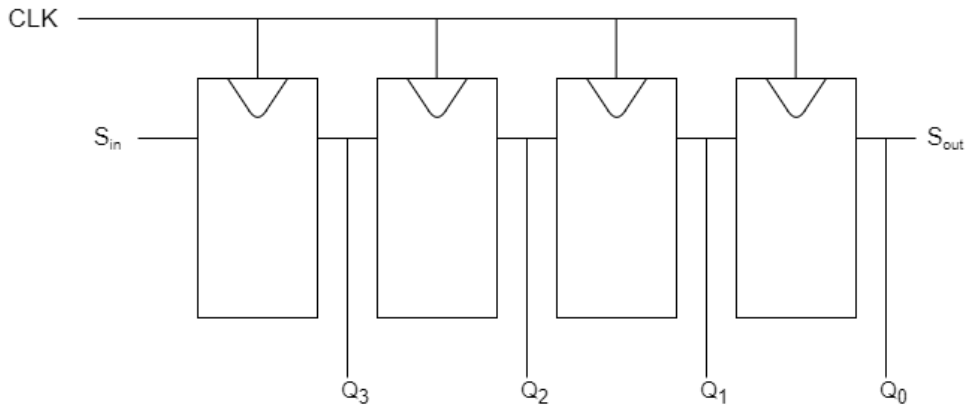
A_3	A_2	A_1	A_0	S1	S0	Y_3	Y_2	Y_1	Y_0
1	1	0	1	0	0				
1	1	0	1	0	1				
1	1	0	1	1	0				
1	1	0	1	1	1				

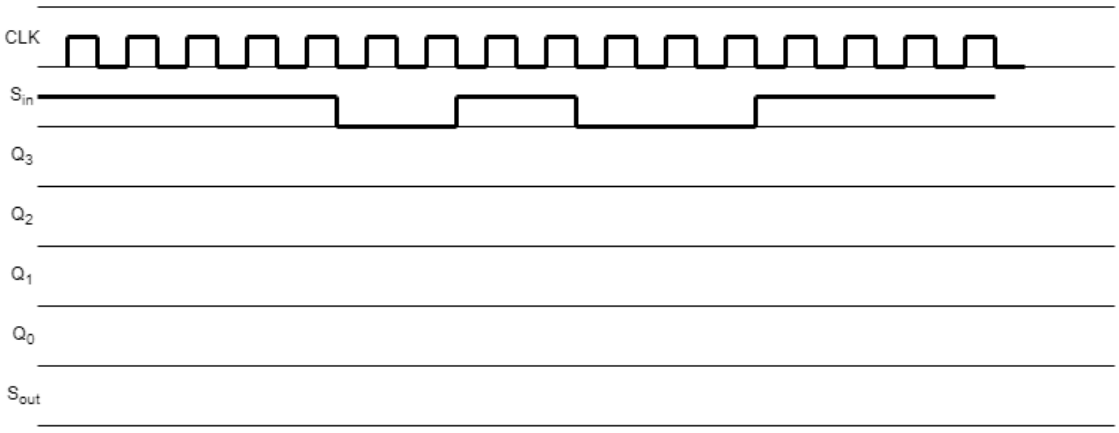
The example above is a _____, and it is purely _____ logic. The circuit has no memory, and the outputs are based purely on the present inputs.

Example: Now let's take a look at a shift register. Let's assume we want the logical circuit to have some memory. In CS210, a great example was to use the shift right to implement the divide by 2 function.

For example: 1010_2 -> 10_{10} if I now execute a right shift by 1, I get 0101_2 -> 5_{10}

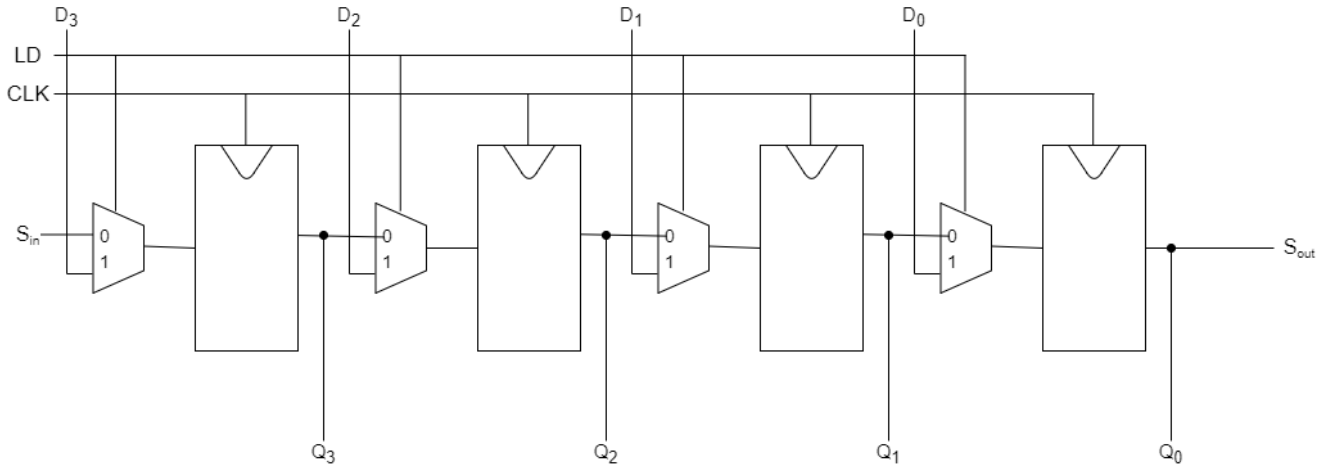
How could we now implement a right shifter where I don't lose track of the previous bit inputs? I want to keep track (i.e. memory) of previous inputs but shift them to the right on each clock cycle.

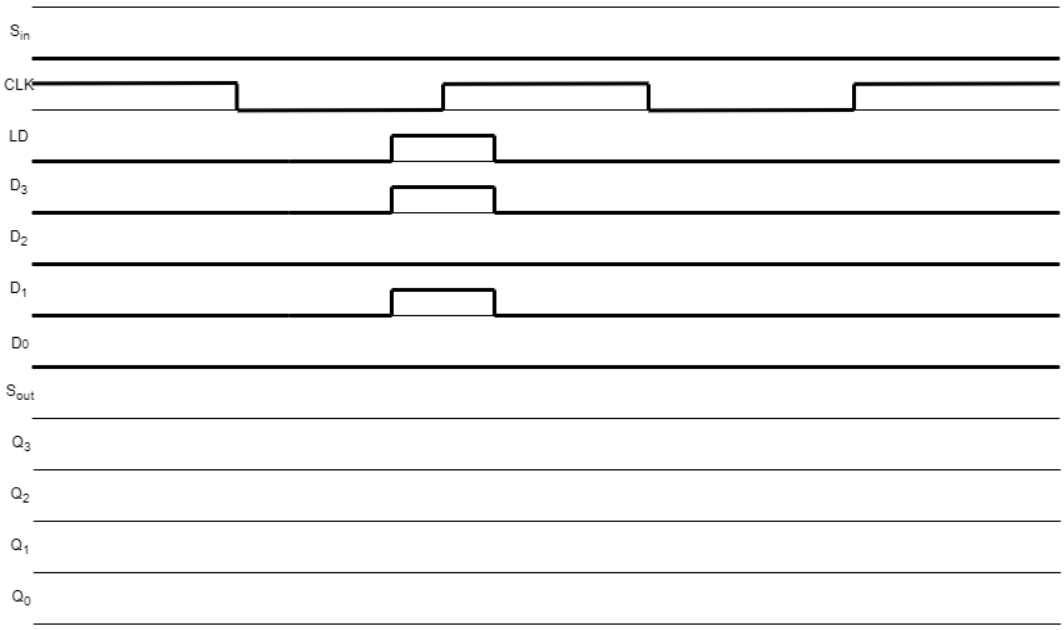




With the example above, We could only load one bit at a time into the register. How could we modify this circuit to load 4 bits at a time. For instance if I wanted to implement the ability to divide a 4-bit number by 2 in just 2 clock cycles, how could I do that?

Example: use a shift register with load capability to divide 1010_2 -> 10_{10} by 2 executing a single left shift to get 0101_2 -> 5_{10}





When LD = 1, acts as a

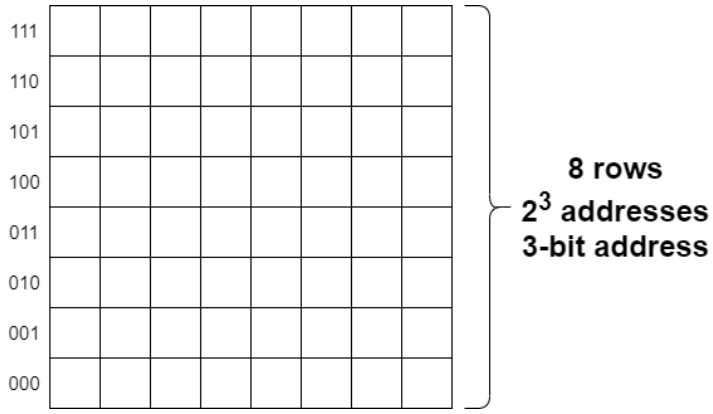
When LD = 0, acts as a

Notice, you could use the above design as a very simple serial to parallel converter ( ) or a parallel to serial converter ( )

Your zyBooks lesson presented a bunch of information on memory. We will summarize key concepts here for your quick reference. First, let's start with definitions:

Definitions:

Address – indicates row, where there are 2^N rows. Notice, this is significantly different than how we taught you to think of memory in CS 210.



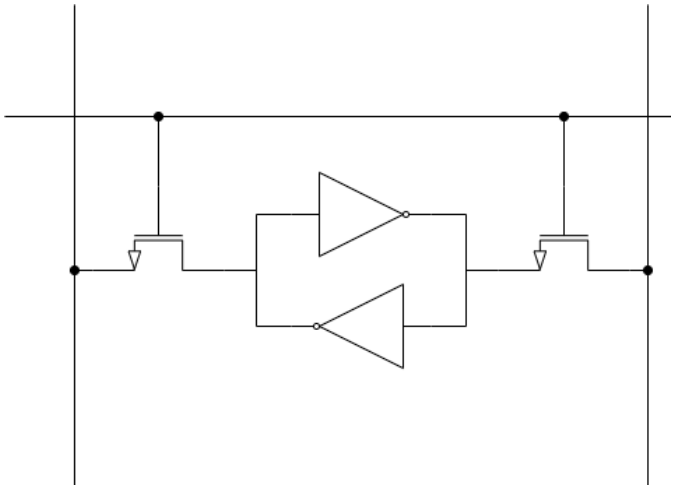
Data – Value read or stored

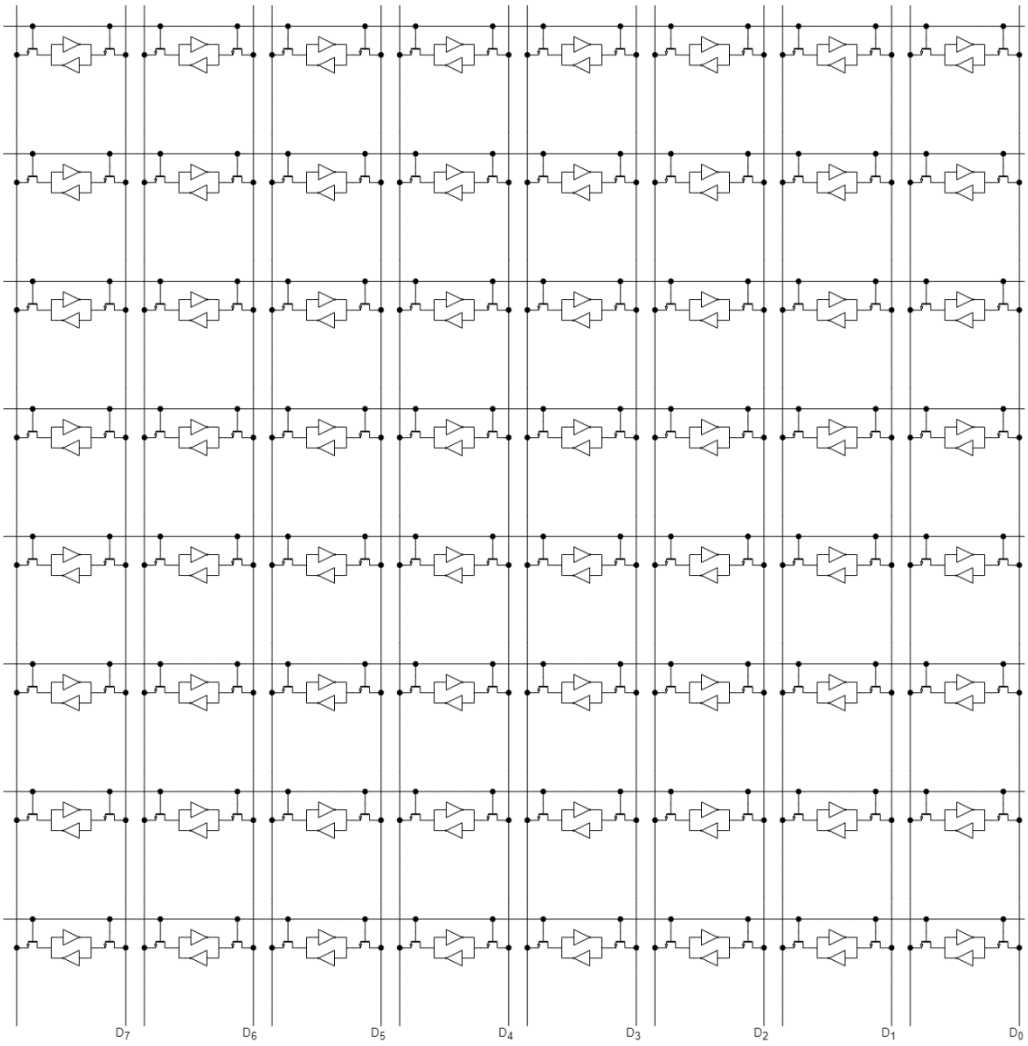
Word – Size of one row (M-bits). The memory above is ___-bit memory

Depth – Number of rows (N-rows). The memory above has a depth of ___

Wordline – word enable activates row

Bitline – data line write/read when wordline enabled





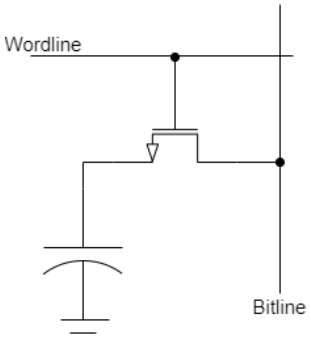
The memory array above would be controlled via logic such as a decoder. The decoder could be used to set a specific wordline as active. Once that wordline is set active, the bits D(7:0) would take on the values stored in each memory cell.

Additional Definitions:

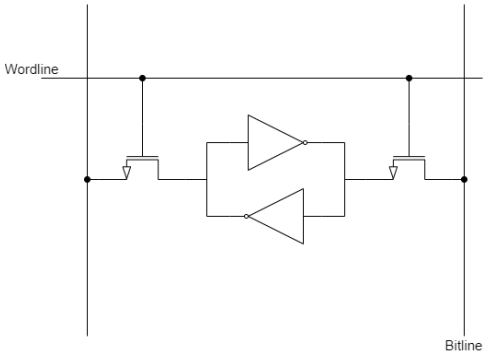
Random Access Memory (RAM) – Same delay for all addresses. Volatile… in other words, memory is erased when power is removed

Read Only Memory (ROM) – non-volatile memory. Name is a bit of a misnomer since you can write to this memory (obviously), but writing is generally significantly slower than reading with this type of memory.

Dynamic RAM (DRAM) – bit stored on a capacitor. Because the capacitor won't hold its charge forever (i.e. it leaks), it must be refreshed. However, this memory is very dense and cheap in comparison to SRAM. In actuality you typically need to refresh on the order of milliseconds.



Static RAM (SRAM) – Doesn't need refreshing. Larger, but also faster.



Programmable Read Only Memory (PROM) – provides a way to connect the transistor with ground

Fuse Programmable ROM – blow a fuse once written to. Cannot be rewritten

Erasable PROM – Uses high voltage to write, and UV light to erase entire PROM at once

Electrically EPROM – Bit cells individually erased, but requires extra circuitry

Flash – Large blocks erased. Cheaper due to efficiencies with erasing circuitry