

# Notes: Sequential Logic Circuits

## A. Why?

Sequential logic circuits perform functions on stored data.

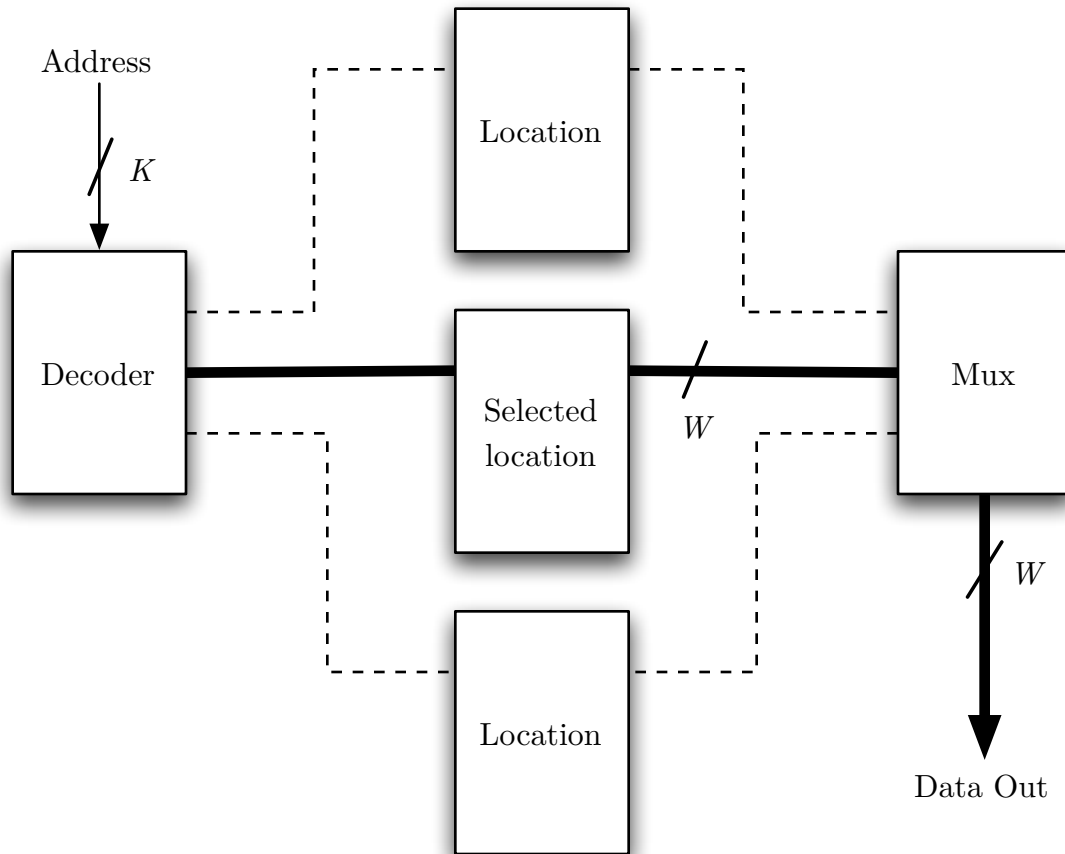
## B. Outcomes

By the end of the class you should

- Understand how decoders, muxes, and latches can be used to form a memory bank.
- Know what a state machine is, what its components are, and how one executes.
- Be familiar with state tables and state diagrams for representing state machines.
- Understand why and how clocks are used with state machines.

## C. Memory

- We've seen that a latch stores one bit of data.
- Typically need many bits of data but we don't want to access all of it all the time.
- Break up memory into pieces, give each piece a name (a location).
  - Addressability = Number of bits per memory location (8? 16? 32? 64?)
  - Address space = Number of memory locations
    - Addresses are unsigned binary integers.
    - If address is  $K$  bits long, then address space is  $2^K$ .
    - Memory typically accessed in words (of 2? 3? ... 8? bytes each)
  - Total number of bits in memory = addressability \* address space
    - Typically expressed in bytes (8 bits/byte)
    - Powers of  $2^{10}$ : kilo-, mega-, giga-, tera-, peta-, exa-
- Read memory
  - Need  $K$ -bit address; goes into decoder to select the location wanted.
  - Each location has its unique line (= wire).
  - When reading the location, its line is ANDed with output of latches for the location.
  - That output is muxed with (all zeros) output of the other memory locations.
  - See textbook Figure 3.21 (2-bit addresses, 3-bits/location)
  - Also Figure 3.22 (shows read 101 from location 11).



Read  $W$  bits from Memory with  $K$ -bit Addresses

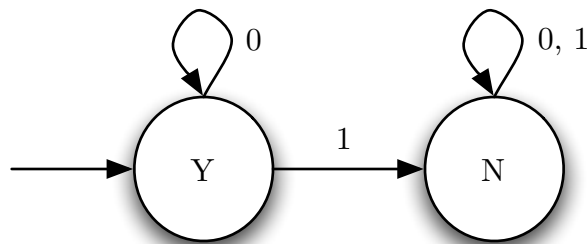
#### D. Sequential Logic Circuits & Finite State Machines

- Sequential logic circuit takes a combinatorial logic circuit, input, and stored data and produces output and updated stored data.
- One characterization uses Finite State Machines
  - The “state” = the contents of stored memory.
  - Assume input and output are (chunks of) bits.
  - Boot machine: memory initialized to “start state”
  - With each input
    - Predetermined function takes input and current state, specifies output and new state.
  - Example: Machine that reads individual bits and tells you if they were all zero.
    - Two states: Y (= “Yes, so far”) and N (= “No”). Y is initial state.
    - No output; answer to question determined by what state machine is in at end of input.
    - In state Y, with input 0, stay in state Y; with input 1, go to state N.

- In state N, with input 0, stay in state N; with input 1, stay in state N.

State	Input	New State
Y	0	Y
Y	1	N
N	0	N
N	1	N

State Table for All-zero recognizer



State Diagram for All-zero recognizer

## E. Clocks and State Machine Transitions

- Different parts of a circuit can take different amounts of time to do their work.
  - Longer wires take more time; a long sequence of gates takes more time.
- Suppose different parts of memory get updated at different rates.
  - Then memory might mix already-computed values with not-yet-computed values.
  - Could cause problems.
- Alternate between using memory and setting memory using a **clock signal**.
  - Clock signal alternates between on and off at a regular speed.
- Master-slave flip-flop uses two latches and clock
  - When clock is 1, data is sent to combinatorial circuit but results aren't stored.
  - When clock is 0, result is stored but data seen by combinatorial circuit doesn't change.