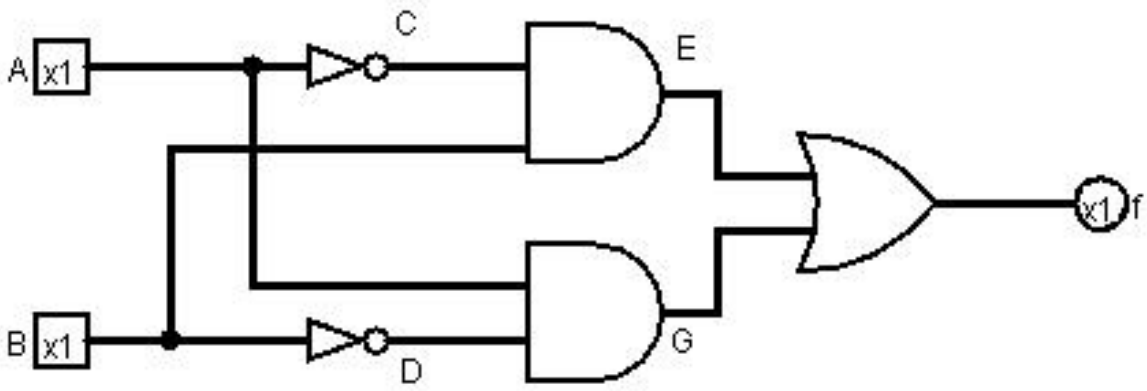
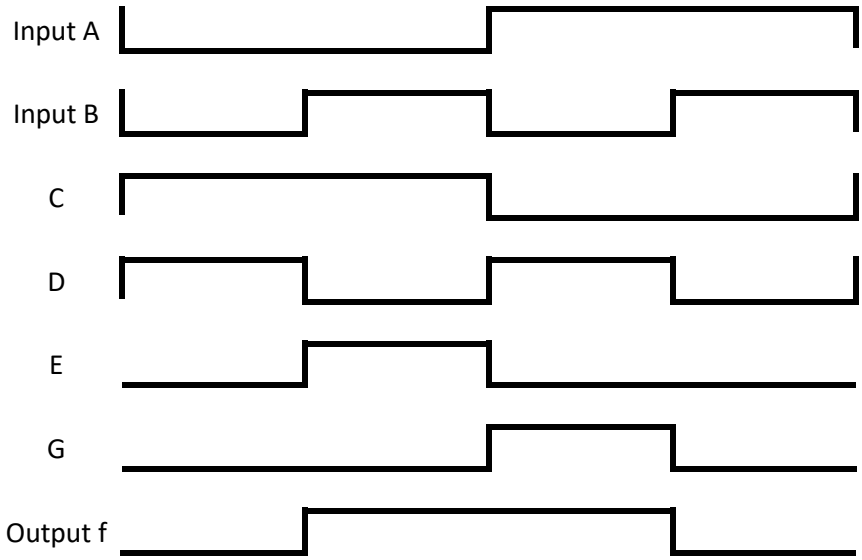
1. More on Karnaugh maps
    a. Further example with don't cares and wrapping
        i. f_3 = m0 + D2 + D5 + D7 + m8 + m10 = $\overline{BD}$

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	d	0	0
11	0	d	0	0
10	d	0	0	1

    b. Whether or not don't cares are included depends on your desired use case
        i. Example: whenever we see an illegal input, raise a flag
        ii. Wouldn't want to include don't cares in this case

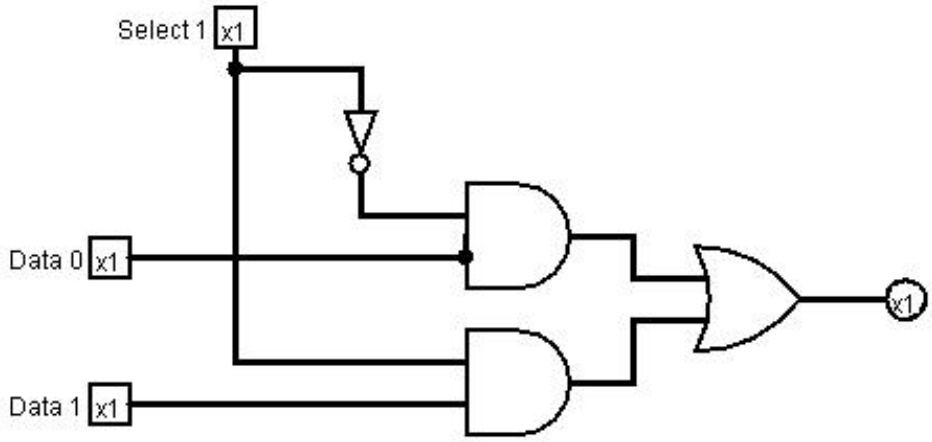
2. Timing
    a. Worst case path
        i. Route through a circuit that takes the longest
            1. "Longest" can be number of gates or time, see below
        ii. In the example on next page, worst case path is 3 gates
            1. Either A -> NOT -> AND -> OR or B -> NOT -> AND -> OR
            2. See below for time
        iii. Worst case path determines clock speed
            1. Need to wait for all gates to finish to get output before we start new clock cycle
        iv. Can also assign times to each type of gate to determine time output is generated
    b. Timing diagram
        i. Visual representation of truth tables at different parts of a digital logic circuit
        ii. Initial values of inputs iterate over all possible combinations (like a truth table)
        iii. Truth values at later points are shown based on each of those combinations
    c. Example
        i. Implementation of XOR using only AND, OR, NOT on next page
            1. A ⊕ B = A * ~B + ~A * B
        ii. Assume that we have the following gate delays
            1. NOT = 3 ns
            2. AND = 5 ns
            3. OR = 4 ns
        iii. What is the clock time of the circuit, if inputs become valid at time 0 ns?
            1. A -> NOT -> AND -> OR (other worst case path has identical times)
            2. 0 ns + 3 ns + 5 ns + 4 ns = 12 ns





3. Combinational circuit building blocks
    a. Talked about gates so far
    b. We now move on to other important pieces of digital design
    c. These pieces build upon the gates that we just learned

4. Multiplexors
    a. Also known as MUXes
    b. MUXes select one output from multiple inputs
        i. Used to control signal and data routing
    c. For *n* data inputs, a MUX has *s* = $\lceil \log_2 n \rceil$ select bits that control which input we select
    d. Implementation
        i. *n* AND gates
            1. Each one has one data input
            2. Also has *s* select inputs
            3. The select inputs are all true for only one AND gate at a time
        ii. One OR gate
            1. *n* inputs from the AND gates
            2. Only one of the AND gates will have a non-zero output (unless that gate's output is all zeroes)
    e. Example
        i. 2 to 1 MUX



        ii. 4 to 1 MUX