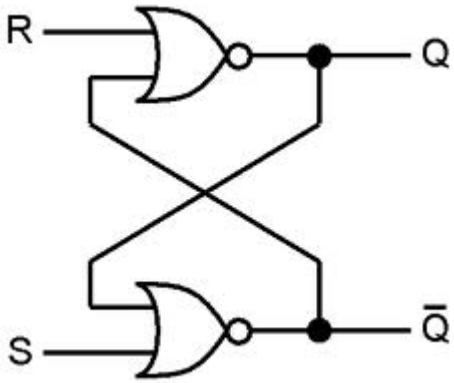
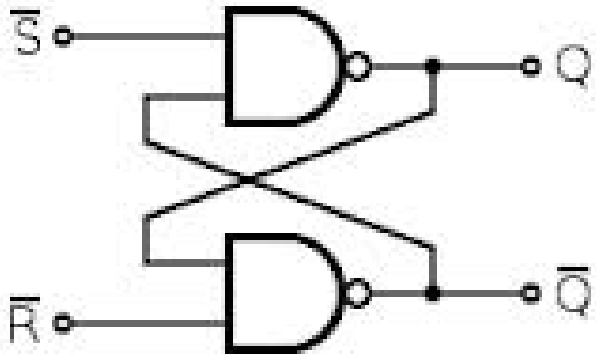
1. More on error correction
    a. From last time - which parity bits check which data bits
        i. For a 3-digit number (say, 100) the first, most significant bit is the 4 bit
        ii. The 0 right next to it is the 2 bit, then the 0 next to it is the 1 bit
    b. Can expand further for larger bit numbers
        i. C0 is in location 1
            1. Checks all bits with bit 1 high (except for itself)
            2. 3 (01**1**), 5 (10**1**), 7 (11**1**), 8, 11, 13, so on
        ii. C1 is in location 2
            1. Checks all bits with bit 2 high (except for itself)
            2. 3 (0**1**1), 6 (1**1**0), 7 (1**1**1), 10, 11, 14, 15, 18, 19, so on
        iii. C2 is in location 4
            1. Checks all bits with bit 4 high (except for itself)
            2. 5 (**1**01), 6 (**1**10), 7 (**1**11), 12, 13, 14, 15, 20, 21, so on
        iv. If we had a C3, it would be in location 8
            1. Would check all bits with bit 8 high (except for itself)
            2. 9, 10, 11, 12, 13, 14, 15, 24, 25, so on
        v. In general, next check bit lies at bit positions that are powers of 2
    c. Why we can XOR to determine the bit position
        i. We are essentially creating a binary number based on all the check bits
        ii. Whether or not a given data position is set or not affects all the check bits that check that location
            1. If a bit is changed in the data, all the check bits associated with that location will be affected
            2. All them will differ by 1 from the original calculation for the check bit at that position
            3. Take the example above: bit 6 changed, so the two check bits that were checking it (C2 and C1) changed as well
            4. Combination of which check bits were affected form the position of which bit was changed
    d. Expanding it further
        i. Expanding to cover a larger data size
            1. $2^K - 1 >= M + K$, where M = data bits, and K = check bits
            2. From above, $2^3 - 1 >= 4$ data bits + 3 check bits
        ii. Covering more errors
            1. Given T errors
                a. Hamming distance between valid code words must be T + 1 to detect
                b. Hamming distance between valid code words must be 2T + 1 to correct

2. Sequential circuits
    a. So far, only discussed combinational circuits
        i. Output strictly dependent on the inputs only
    b. Sequential circuits – output depends on the current inputs as well as the *history* of inputs
        i. Other way to phrase it – they have memory
        ii. History of inputs determines “state” of the circuit
    c. Examples
        i. House alarm that needs to be reset once an intrusion is detected
        ii. Combination lock

3. Latches
    a. Circuit that has two stable states, can be used to store information
    b. Change state via signals applied to the control inputs
    c. S-R latches
        i. Set sets the latch
        ii. Reset clears the latch
        iii. Q and !Q give the output and complement
    d. Implementation



S-R NOR Latch



S-R NAND Latch

    e. Characteristic tables

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Undefined

S-R NOR Latch

S	R	$\overline{S}$	$\overline{R}$	Q_{n+1}
0	0	1	1	Q_n
0	1	1	0	0
1	0	0	1	1
1	1	0	0	Undefined

S-R NAND Latch

    f. State transition table for S-R NOR latch
        i. When creating a state transition table, need to list out all possible current states and inputs, like below
        ii. Combination of inputs and present state form entire list of possible outcomes

Present Inputs		Present State	Next State	Type of Circuit
S	R	Q	Q'	
0	0	0	0	Memory
0	0	1	1	Memory
0	1	0	0	Combinational
0	1	1	0	Combinational
1	0	0	1	Combinational
1	0	1	1	Combinational
1	1	0	?	Combinational
1	1	1	?	Combinational

g. State transition diagram
    i. The values in the circles are Q, the output of the latch
    ii. The values in parentheses (0, 1) are the combination of (S, R) respectively
    iii. The arrows are transitions from one state to the next (or staying in the same state)
    iv. For every state, all possible output combinations need to be listed for transitions to the same or other states
        1. (1, 1) isn't listed here as it leads to undefined behavior