Phys. 334 **Day 8 Lecture Notes**

Assignment: Read in text sections 8.14-8.15 (pp. 493-499)
Read in manual pp. 283-289, 295-299, and optional 301-306
Practice problems (not to be turned in)
1.) #8.13, p. 493 2.) #8.15, p. 494
3.) #8.16, p. 495 4.) #8.21, p. 499

Combinatorial Logic:

To design circuits to perform a logical task, the general approach is:
1.) Write the truth table.
2.) Write the Boolean equation for the table, if possible
3.) Simplify the Boolean equation
4.) Implement the equation with gates.

For 2 inputs, the implementation can usually be done by inspection, such as with our 2-bit adder example or car buzzer example.

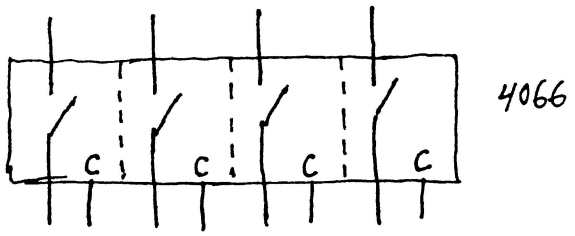
For 3 or 4 inputs, Karnaugh maps are useful (see section 8.13 of the text).

For 5 or more inputs, computer programs are available if reduction of the problem to fewer inputs is not obvious.

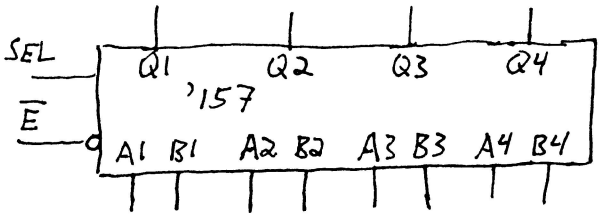
Combinatorial IC's:

In addition to Boolean logic functions that are handled by gates, there are a variety of chips that do other useful functions, such as switching, routing, and arithmetic. We will look at some examples of these.

Transmission Gate: This is a mix of digital and analog: basically an analog switch that is controlled by a digital voltage. For example, the 4066 has 4 (quad) independent switches on the chip: when the digital Control (C) signal is LOW, the switch is open (very high impedance), while when Control is HIGH, the switch is closed (≈100 Ω impedance). Either side of the switch may be considered as the input or output: it's just a digitally controlled electrical connection. The 4066 is called a 'quad bilateral switch'.



2-input Select: These are digital switches: they require digital inputs and they regenerate these inputs to give digital outputs. For example, the 74xx157 is a quad 2-input select shown below. It has 4 pairs of input signals, A_n and B_n and 4 outputs, Q_n. When Enable is HIGH, the chip is disabled and all of the outputs are LOW. When Enable is LOW, the chip is enabled and the Select input selects which inputs are passed to the outputs: Select LOW passes A_n to Q_n, Select HIGH passes B_n to Q_n. Note, unlike the 4066 transmission gate, the single Select line controls all 4 switches and signals are passed only in one direction.

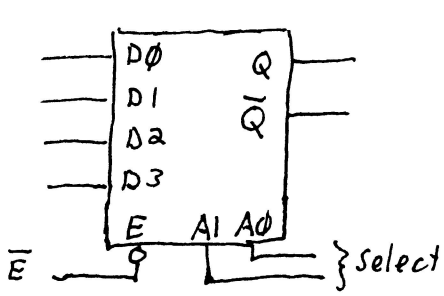


$\overline{E}$	SEL	A_n	B_n	Q_n
H	X	X	X	L
L	L	L	X	L
L	L	H	X	H
L	H	X	L	L
L	H	X	H	H

X = Either H or L

The 74xx158 is similar, except it inverts the selected input signals. The 74xx257 and 74xx258 are similar except with 3-state outputs (when the chip is disabled, the outputs are very high impedance).

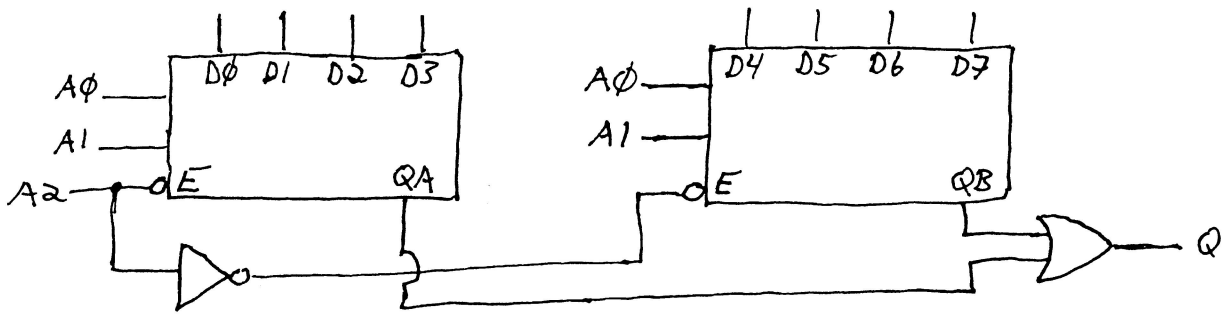
Multiplexer (MUX): These are switches that where one of 2^n inputs is passed to a single output. To do so requires 'n address' lines whose binary number selects one of the 2^n inputs. The 2-input select is just a 2-input MUX (n = 1). A 4-input MUX is shown below, with its truth table. In this case, when Enable (also called Strobe) is HIGH, the chip is disabled and its output Q is LOW ($\overline{Q}$ HIGH). Some MUX's have 3-state outputs.



$\overline{E}$	A1	A0	Q
H	X	X	L
L	0	0	D0
L	0	1	D1
L	1	0	D2
L	1	1	D3

← Not 3-State Output

What if you have 8-inputs, but only 4-input MUX's? Can we make an 8-input MUX? This is called 'expansion' and the answer is yes. We now need an additional address line, A2, and we connect this to the Enable inputs of two 4-input Mux's: When A2 is LOW the first MUX is selected (the second not) and vice-versa when A2 is HIGH. We use an OR gate to combine the two outputs because when one input of an OR is LOW (from the MUX that is disabled), the output of the OR equals the other input: Q OR 0 = Q.



MUX as Arbitrary Truth Table: Although we think of MUX's as digital switches, they can be used other purposes. For example, we can use an N-input MUX to generate an arbitrary 2N-input truth table (with a single output, Q). For example, an 8-input MUX (which has 3 address lines: 2^3 =8) can be used to generate a 16-input truth table (that requires 4 address lines).

To do this, we use the 3 address lines of the MUX as the high 3 address lines of the truth table, and the least significant address line of the truth table will be used as data input to the MUX. Call the 4 address (select) lines of the truth table S3, S2, S1, S0 as shown below.

When S3, S2, S1 = 000, then the MUX connects D0 to Q and there are only 4 possible outputs for S0 = 0 and 1: Q = 0 and 0, Q = 0 and 1, Q = 1 and 0, Q = 1 and 1. Just wire up the data inputs to the MUX accordingly.

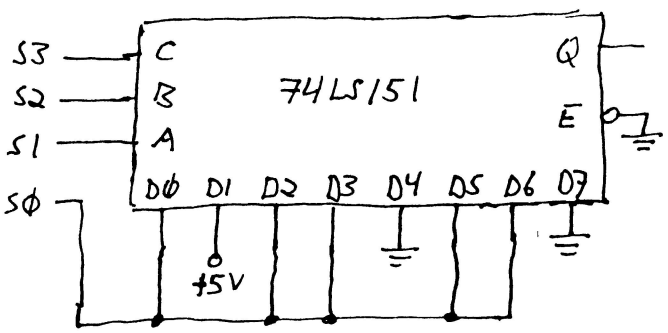
S3	S2	S1	Sϕ	Q	Q	Q	Q
C	B	A	ϕ	ϕ	ϕ	1	1
C	B	A	1	ϕ	1	ϕ	1
Selects D_{CBA} in				"	"	"	"
				ground	Sϕ	$\overline{S\phi}$	+5V

← 4 possible values for Q when Sϕ = 0, 1

← connection to D_{CBA} input of MUX

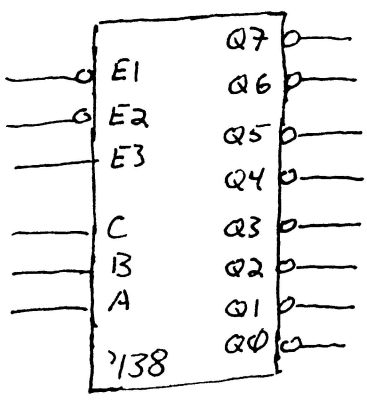
For example, let's use an 8-input MUX, the 74xx151, to tell us which numbers, 0-15, are prime (0 is not, 1 is). The truth table is shown below. S3, S2, S1 select each pair of numbers and the output is either 0 for both, 1 for both, = S0, or = S0. We wire up the data inputs accordingly. You will do a similar circuit in the lab where you use the 74LS151 to tell us whether a month has 31 days or not.

S3	S2	S1	Sϕ	$\#_{10}$	Q (1=prime)	
0	0	0	0	0	0	Sϕ
0	0	0	1	1	1	
0	0	1	0	2	1	+5V
0	0	1	1	3	1	
0	1	0	0	4	0	Sϕ
0	1	0	1	5	1	
0	1	1	0	6	0	Sϕ
0	1	1	1	7	1	
1	0	0	0	8	0	ϕV
1	0	0	1	9	0	
1	0	1	0	10	0	Sϕ
1	0	1	1	11	1	
1	1	0	0	12	0	Sϕ
1	1	0	1	13	1	
1	1	1	0	14	0	ϕV
1	1	1	1	15	0	



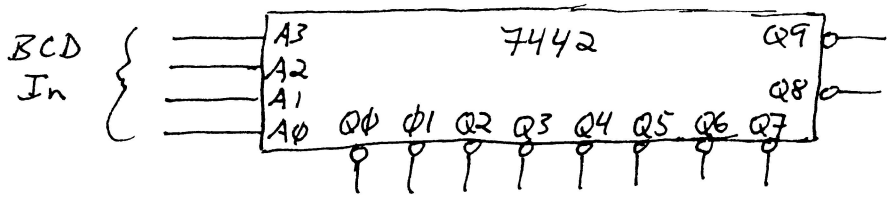
Demultiplexer (DeMUX) and Decoders: These are MUX's in reverse: a single input is routed to one of 2^n outputs, again selected by n address lines. The unselected outputs are either held to their inactive state or are open-circuited, depending on the DeMUX.

A decoder is like a DeMUX except the address lines become the input (there is no additional input). The output selected by the address lines goes to a predetermined active state while the unselected outputs stay in their inactive states. A common example is the 74LS138 1-of-8 decoder: When its 3 enable inputs are all true (E1 and E2 LOW, E3 HIGH) then the output selected by the address lines, C,B,A, goes true (LOW). If the output is not selected or the chip is not fully enabled, then the outputs remain false or HIGH.



$\overline{E1}\cdot\overline{E2}\cdot E3$	C	B	A	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Qϕ
L	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	H	H	H	H	H	H	H	L
H	L	L	H	H	H	H	H	H	H	L	H
H	L	H	L	H	H	H	H	H	L	H	H
H	L	H	H	H	H	H	H	L	H	H	H
H	H	L	L	H	H	H	L	H	H	H	H
H	H	L	H	H	H	L	H	H	H	H	H
H	H	H	L	H	L	H	H	H	H	H	H
H	H	H	H	L	H	H	H	H	H	H	H

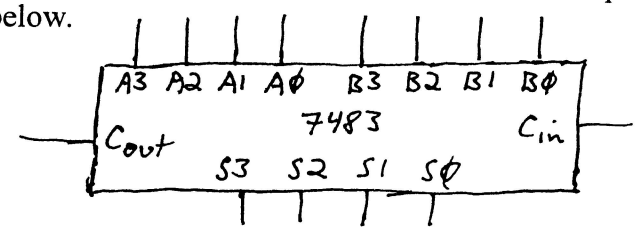
Another decoder that used to be common was the 74LS42: BCD to decimal decoder. A BCD number is a binary number between 0 and 9. If we want to convert this binary number to a decimal number, say on a display chip, we need to decode the binary inputs bits. The binary number appearing on the chip inputs causes the corresponding output to be true (LOW in this case). The other outputs stay false or HIGH. These outputs are then connected to 10 inputs of a '7-segment display' chip, which then displays the decimal digit.



Adders: There are chips to do arithmetic such as addition, subtraction, and multiplication. For example, we will look at chips that can add. Adding binary numbers is just like adding base 10 numbers, except you carry to the next column when the sum in the previous column is greater than or equal to 2. Your result may require an addition bit (column) than the original addends - - this extra column is called the 'carry' bit. For example, to add 5 + 3, see below.

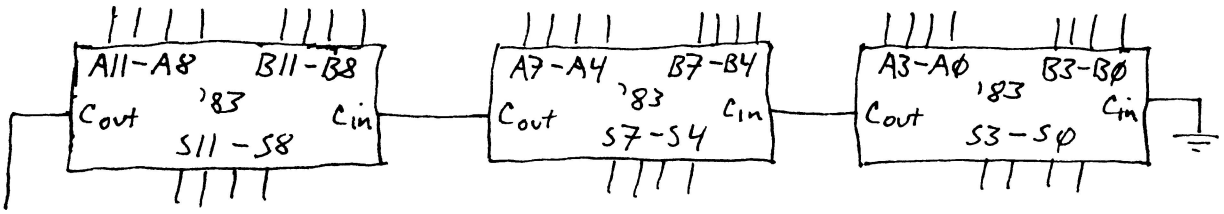
```
  1 1 1
3 = 0 1 1
+5= 1 0 1
--------
8 = 1 0 0 0   - 4th bit of result called "Carry" bit.
```

The 7483 is called a 4-bit full adder. It takes two 4-bit input numbers, A_n and B_n, and a Carry-In bit (C_{in}) and adds them to create the sum output, S_n, and Carry-Out (C_{out}) bit, as shown below.



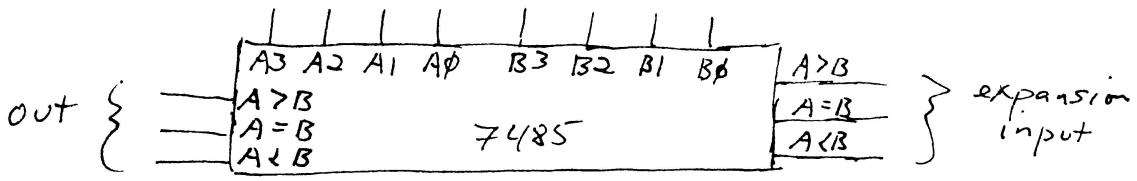

```
   A3 A2 A1 Aφ
+  B3 B2 B1 Bφ
+              Cin
-----------------
Cout S3 S2 S1 Sφ
```

The C_{in} and C_{out} lines allow you to cascade (connect) several chips to add numbers that are larger than 4-bits. Just connect the C_{out} bit from the least significant chip to the C_{in} bit of the next 4-bit sum as a chain, remembering to ground C_{in} of the least significant chip.



13-bit result of adding 2 12 bit #'s.

Magnitude Comparator: The 7485 compares two 4-bit inputs, A_n and B_n, and tells you whether A>B, A=B, or A<B (when considered as a binary number). Expansion inputs are used so the chips can be cascaded to compare inputs with more than 4-bits.

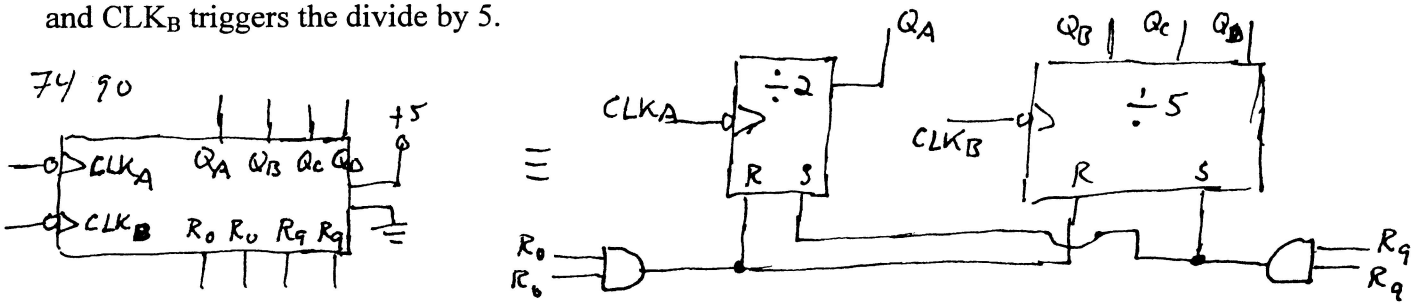


Counters: We've already seen how to make a N-bit ripple counter by a succession of flip-flops (D or J-K) where each output is connected to the CLOCK input of the next higher bit flip-flop. In general, you wouldn't do this; there are a number of counter chips that do things much better. Different counter chips vary in the properties listed below.

1) There are synchronous (all outputs change at the same time) and ripple counters available.
2) There are 4 and 8 bit outputs, configured to divide by 10, 12 or 16 (this means to cycle through that many states) and there is always a means provided to cascade chips to count even higher.
3) There are positive and negative edge triggered clock inputs, and some chips count up, others down, and others both.
4) Some counters have input lines that allow a pre-set count to be loaded. The counter then counts up or down from the pre-set value (modulo n counter).
5) Some counters have latches on the outputs (the outputs don't change until you ask for them), others have 3-state outputs.

We will look at a few examples.

7490 Decade Counter: (divide by 10 ripple counter) This is a counter that contains separate divide by 2 and divide by 5 parts, as shown below. CLK_A triggers the divide by 2 flip-flop and CLK_B triggers the divide by 5.



Q_A is CLK_A divided by 2 (a D-type flip-flop with Q connected to the D input).

CLK_B is divided by 5, making the $Q_D Q_C Q_B$ outputs cycle through the states 000, 001, 010, 011, 100, and back to 000.

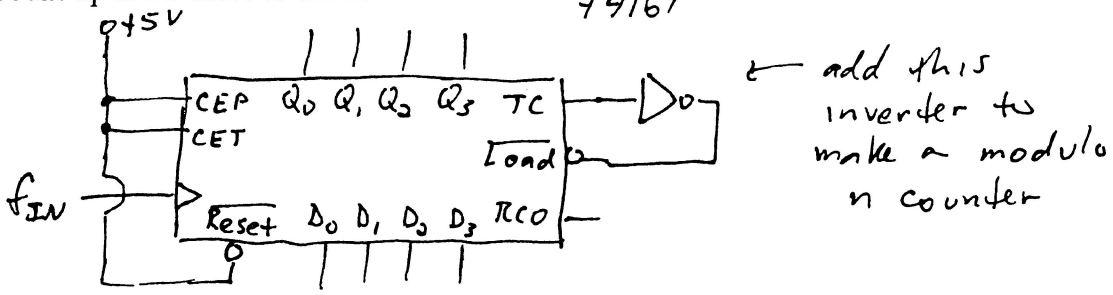
To use the 7490 as a decade counter, you connect Q_A to CLK_B and use CLK_A as the input.

When both R_0's are HIGH (true), the output is reset to 0: $Q_D Q_C Q_B Q_A = 0000$.
When both R_9's are HIGH (true), the output is reset to 9: 1001.

To divide by 100, you connect Q_D from the divide by 10 counter to CLK_A of another 7490: when the divide by 10 counter rolls over from 9 to 0, 1001 to 0000, Q_D goes from HIGH to LOW, creating the negative edge needed for the second 7490 to register a count (in the tens column).

The 74390 is a dual BCD (decade) ripple counter; essentially 2 7490's on a chip with no R_9 pins to conserve the numbers of pins.

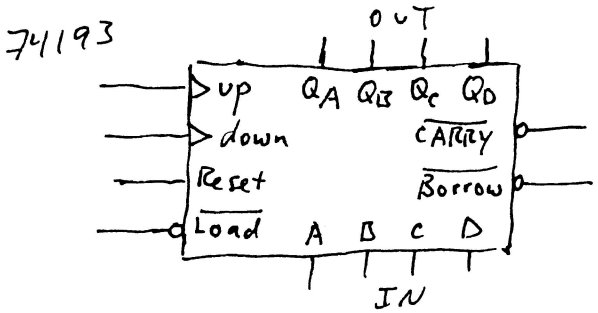
74161 Synchronous Programmable Modulo N Counter: The '161 counts from 0000 to 1111 (0 to 15) and then rolls over back to 0000. If you LOAD a preset value on the input data lines, D_j, by making LOAD LOW (true), then the counter outputs are loaded with this value and count up from there to 1111.



1) The '161 has a synchronous LOAD which means that when $\overline{\text{LOAD}}$ is LOW, the value on the data lines is loaded at the instant of the next positive edge on f_{IN}.
2) The RESET line is asynchronous which means the outputs are set to 0000 as soon as $\overline{\text{RESET}}$ is pulled LOW (true).
3) To make a modulo n counter (a counter that cycles through n states), you place the two's complement of n, called –n, on the data inputs. For example, for n = 5 = 0101, –n = 1010 + 1 = 1011 = 11_{10}. When the count is 15 = 1111, the TC pin (Terminal Count) goes HIGH (true) and you invert this into the $\overline{\text{LOAD}}$ input so that the next f_{IN} edge loads the preset count of eleven again. The states are then 11,12,13,14,15, and back to 11.
4) CEP and CET are used to generate a pulse on RCO (Ripple Counter OUT) when the terminal count is reached, allowing the '161's to be cascaded.

The 74160 is similar to the '161, except it is a BCD (decade) counter (TC is true when the count is 9). The 74162 and 74163 are similar to the '160 and '161 except the RESET is synchronous as well as the LOAD.

74193 Synchronous 4-bit Up/Down Counter: Separate inputs, up and down, allow the counter to count both up and down. Input data lines DCBA along with LOAD allow a preset count to be loaded onto the outputs Q.



positive edge on > up increases count by 1.

positive edge on > down decreases count by 1.

When counting up, if the output rolls over from 15 (1111) to 0 (0000), the CARRY output goes LOW (true). When counting down, if the output rolls over from 0000 to 1111, the BORROW output goes LOW (true). These CARRY and BORROW outputs can then be used as inputs to another '193 to count to larger numbers.