

Homework #3

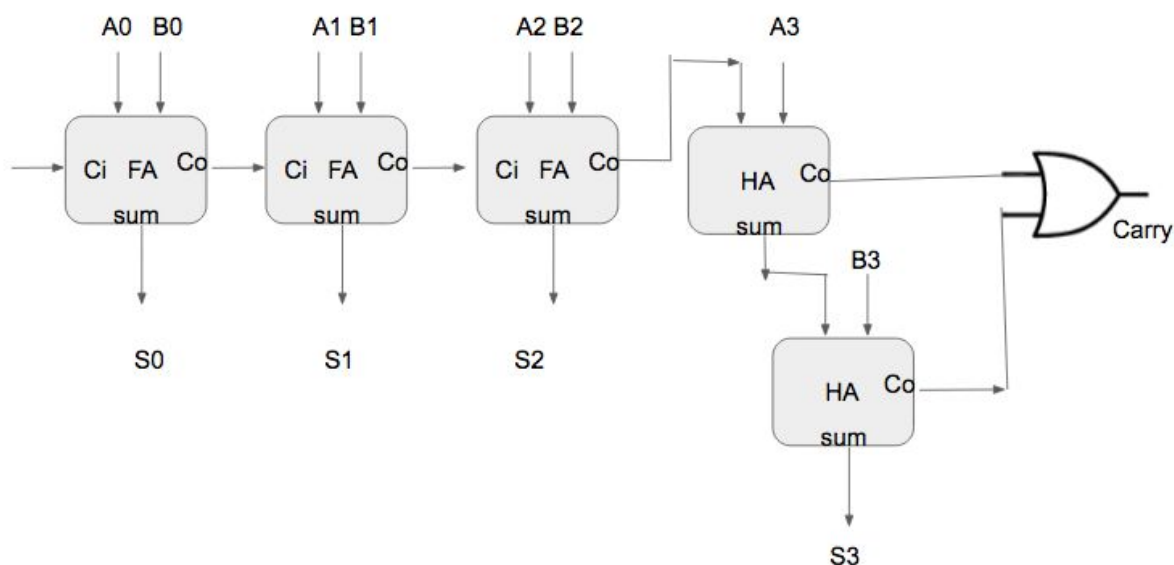
CSE 140 – Summer Session 2 – 2016

Instructor: Pietro Mercati

Only a subset of questions will be graded

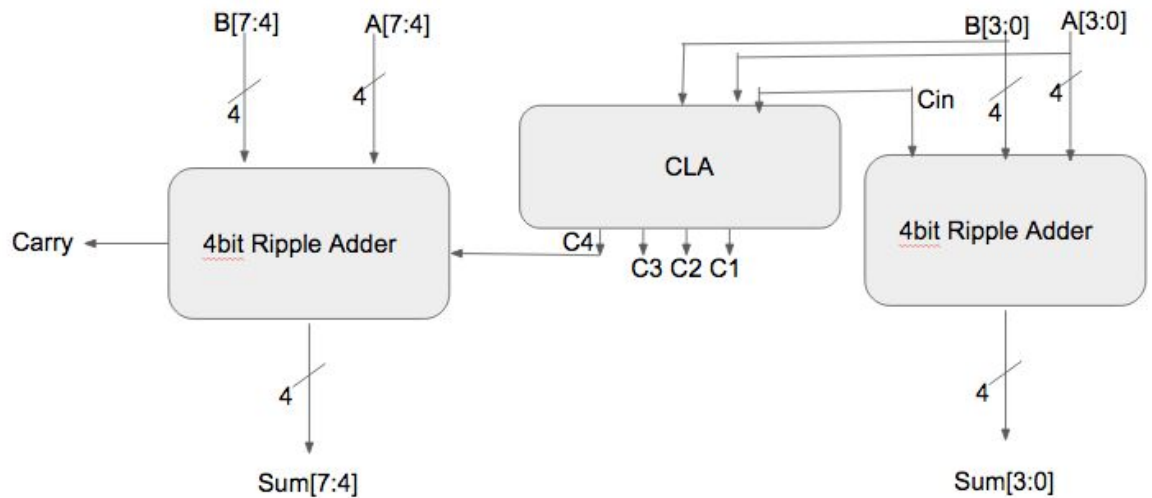
1) Design the following components. For each one, draw a clear diagram.

- a) Design a Ripple-Carry 4-bit adder using 3 Full Adders, 2 Half Adders and an OR gate. You can draw full adders and half adders as blocks labeled with "FA" and "HA" respectively.



- b) Using 4-bit Ripple-Carry adders as building block, design a 8-bit Carry-lookahead adder. Draw the circuit using the Ripple-Carry adders and the Carry-Lookahead logic as building blocks. You are not required to draw the Carry-Lookahead logic circuit into details, but you should show the equation for the carry and clearly label the signals on your circuit.

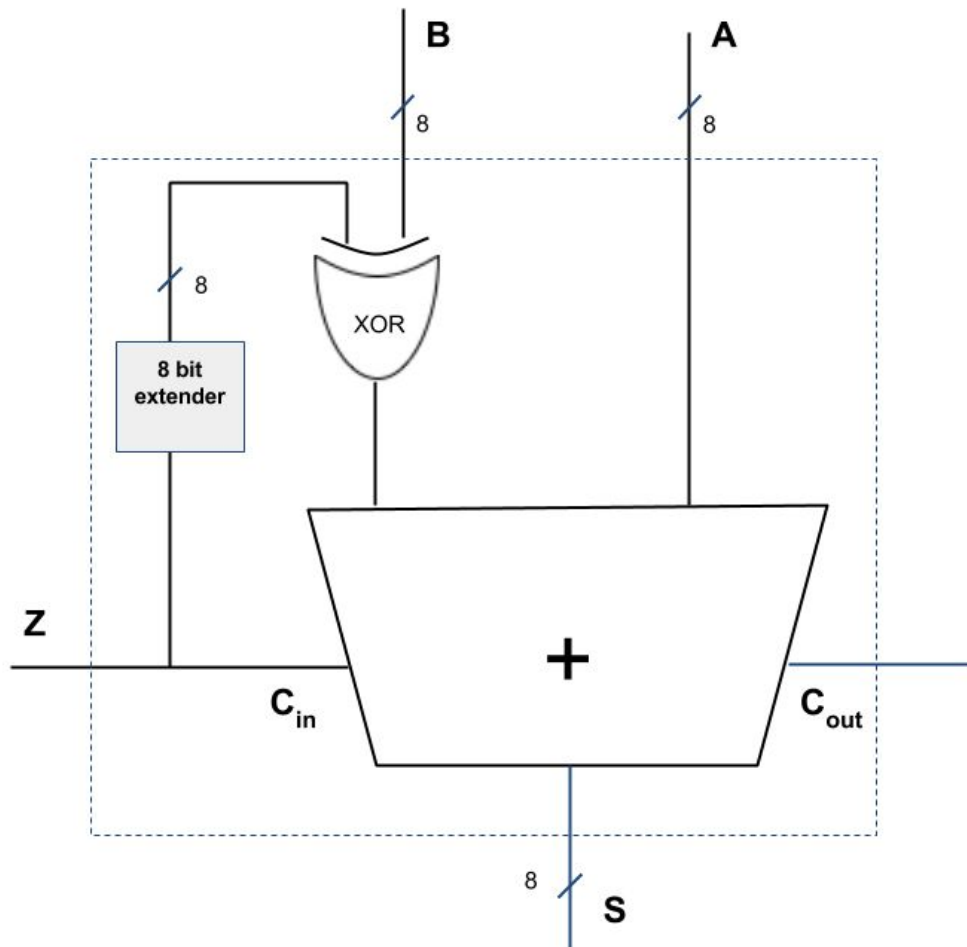
$$C_4 = G_3 + C_3P_3$$



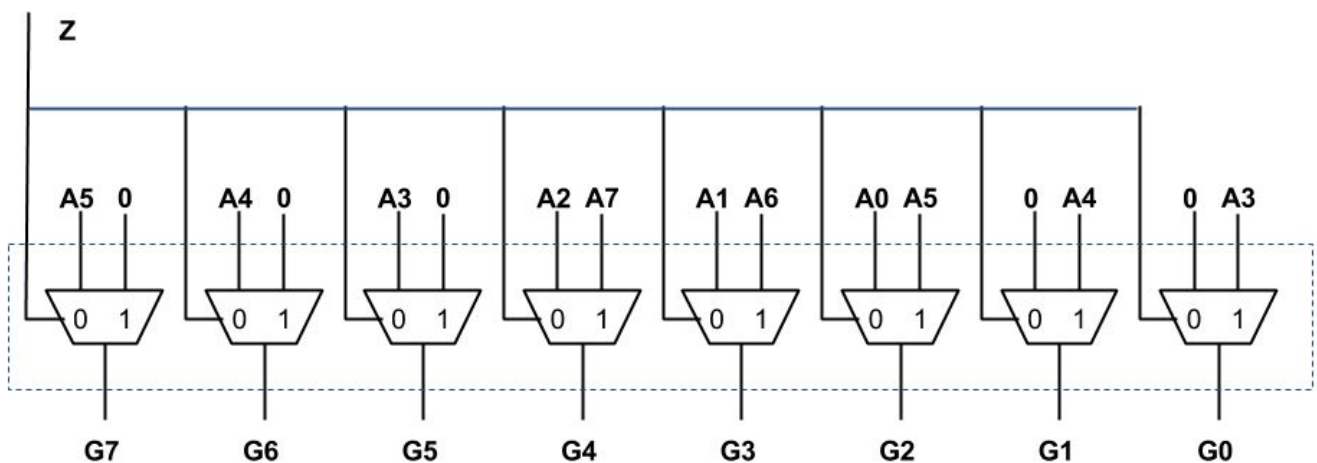
- 2) Design an ALU with three 8-bit inputs A, B and K, and single bit control inputs x, y and z. The ALU should support the operations described in the table below.

Control Inputs			
x	y	z	Operation
0	0	0	A NOR B
0	0	1	A AND B
0	1	0	A + B
0	1	1	A - B
1	0	0	A * 4
1	0	1	A / 8
1	1	0	$(AK^2 + BK) \bmod 256$
1	1	1	Reverse A

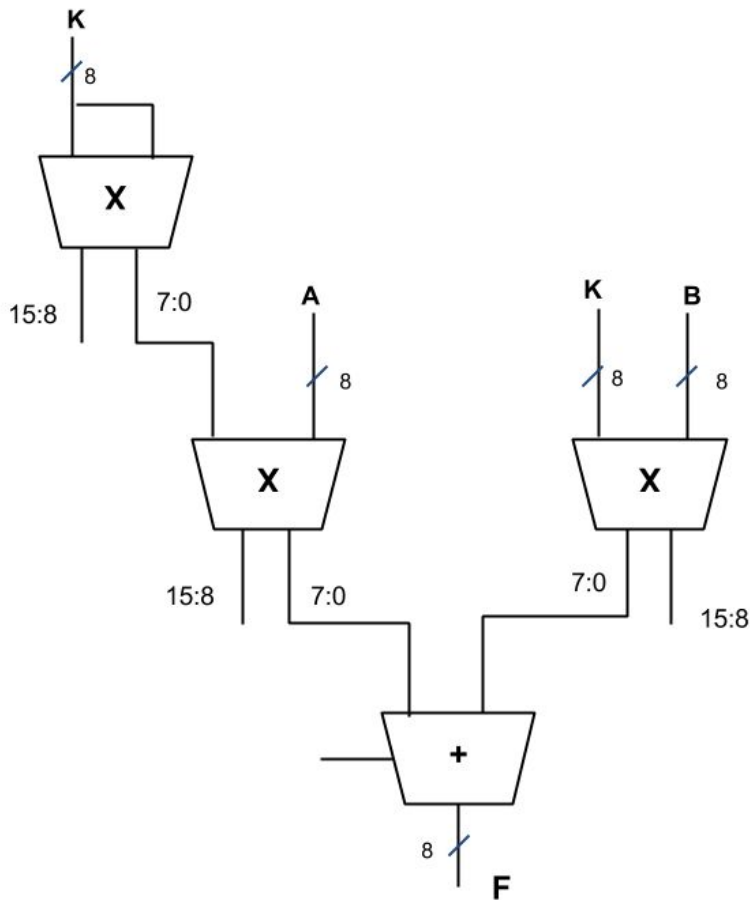
- a) Design of A + B and A - B : Use a single 8-bit adder and logic gates/muxes to implement the logic for both these operations. Hint: Signal Z will be required.



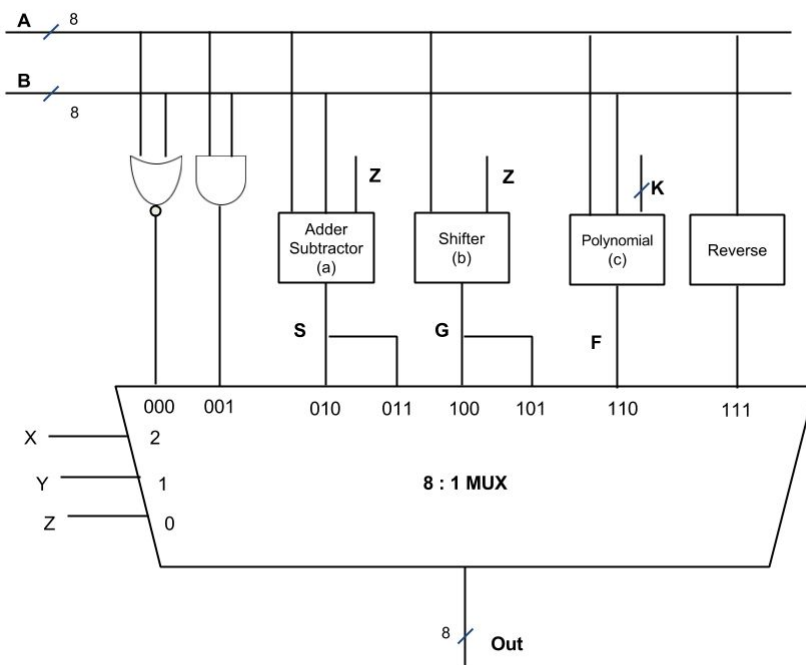
- b) Design of $A * 4$ and $A / 8$: Use a shifter and MUXes to implement both these operations. Hint: Signal Z will be required.

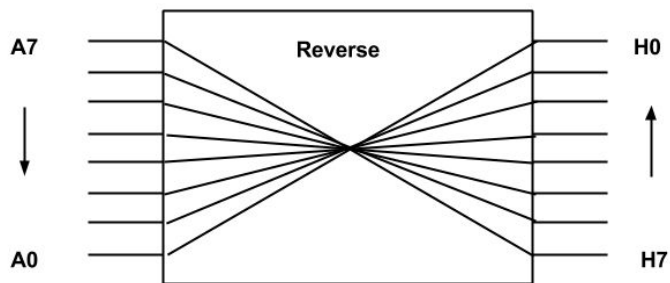


- c) Design of polynomial function $f = (AK^2 + BK) \bmod 256$. 256 is $(100000000)_2$ and the modulus helps us handle overflow. You can use multipliers and only a single adder.

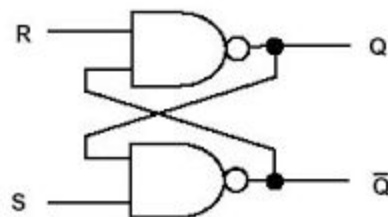


- d) Design of ALU : Use a 8 : 1, 8-input MUX to implement the ALU described above. You can show the components designed in parts (a), (b) and (c) as black-boxes.





- 3) In class we have seen the structure of a SR latch built using NOR gates. Here we want to analyze behaviour of a NAND latch which structure is given to you.

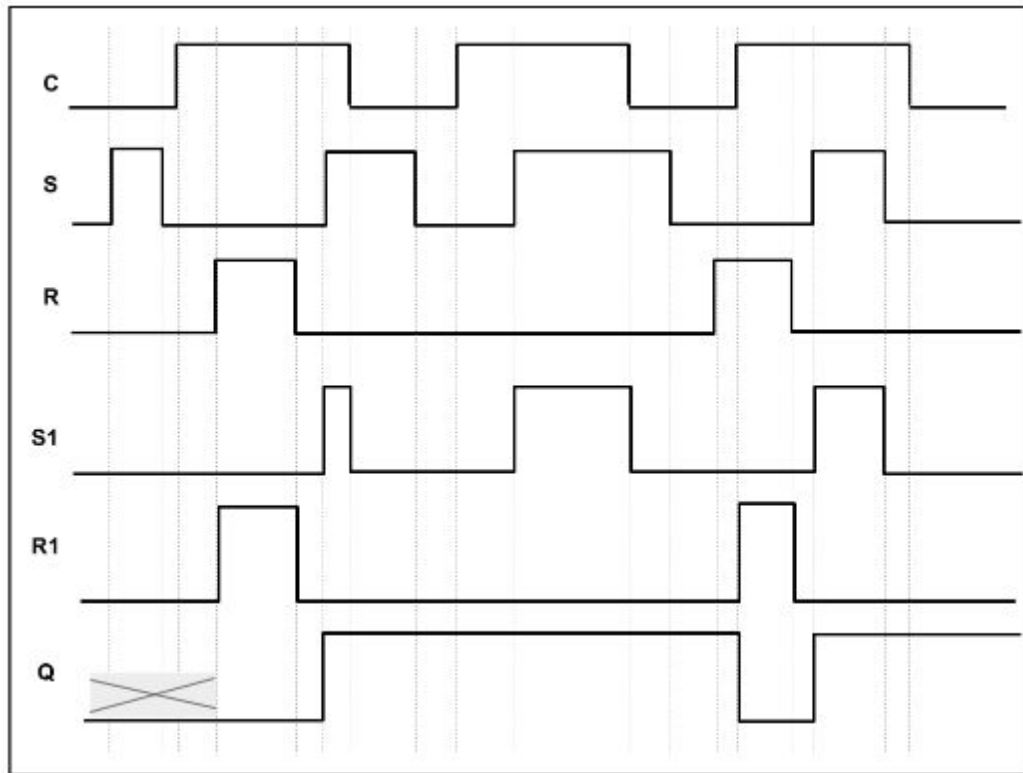


- a) Fill out the given table. Note that similar to NOR latch some of the input combinations should not be allowed, and you need to find them. In the status column you should write one of the followings: hold, set, reset, not allowed.

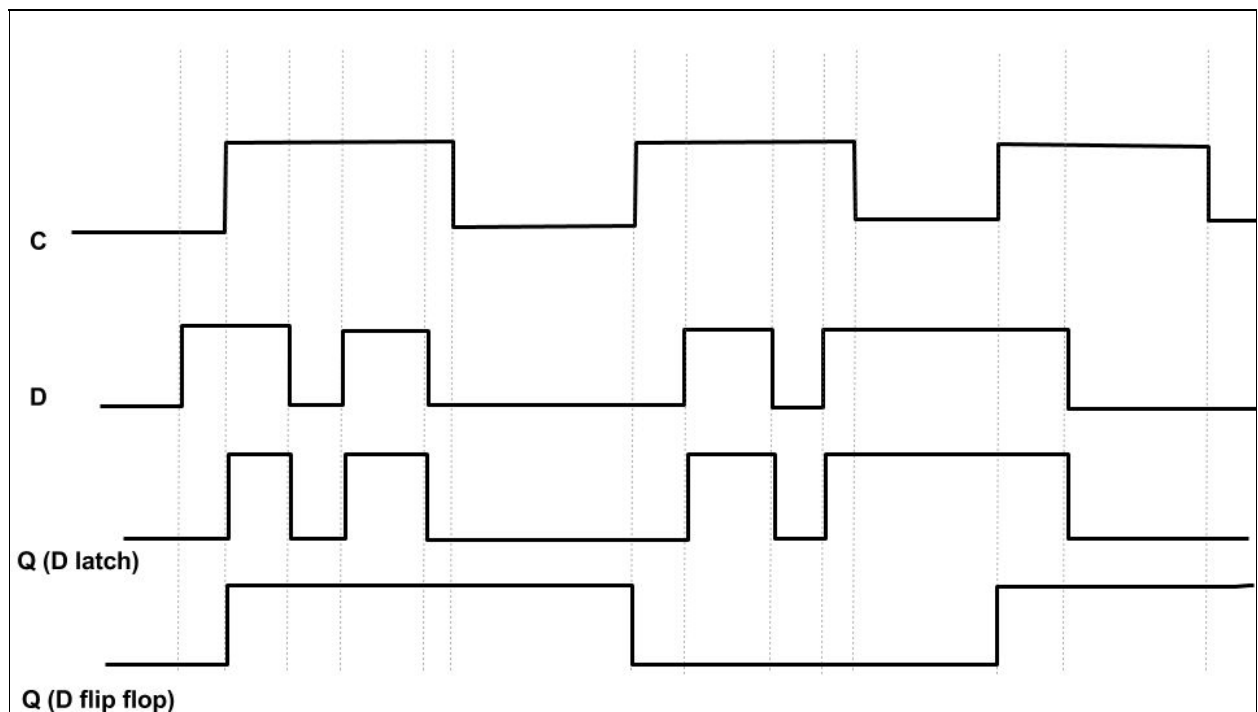
S	R	Q(t)	Q(t+ Δ)	Q'(t+ Δ)	Status
0	0	0	1	1	Not Allowed
0	0	1	1	1	Not Allowed
0	1	0	0	1	Reset
0	1	1	0	1	Reset
1	0	0	1	0	Set
1	0	1	1	0	Set
1	1	0	0	1	Hold
1	1	1	1	0	Hold

4) Timing diagrams

- a) Analyze the behavior of a level-sensitive SR latch and complete the timing diagram with the correct waveforms



- b) Compare the behavior of the D latch and D flip-flop by completing the timing diagram below. D is input, C is the clock. Assume that each device initially stores a 0. Provide a brief explanation of the behavior of each device.



- 5) Design a flip flop which has two control inputs called P and N. This PN flip flop performs the following operations:

if $(P,N) = (0,0)$, reset the output to zero;
 if $(P,N) = (0,1)$, hold the previous output;
 if $(P,N) = (1,0)$, invert the previous output;
 if $(P,N) = (1,1)$, set the output to one.

- Fill out a truth table for this circuit
- Provide the characteristic equation for the PN flip flop
- Show how we can implement D-FF using this PN-FF
- Show how we can implement PN-FF using D-FF and other gates, muxes, etc.

a) Truth table:

P	N	Q(t)	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

The table can be summarized as:

P	N	Q(t+1)
0	0	0
0	1	Q(t)
1	0	Q'(t)
1	1	1

b) We can drive the characteristic equations using K-maps:

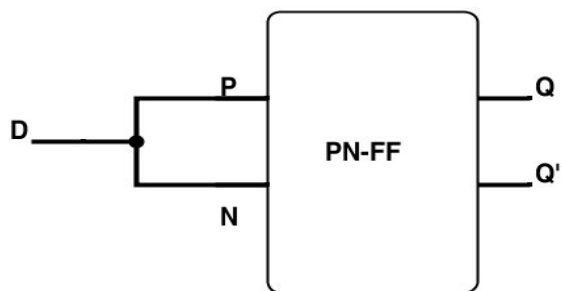
Q(t)\PN	00	01	11	10
0	0	0	1	1

1	0	1	1	0
---	---	---	---	---

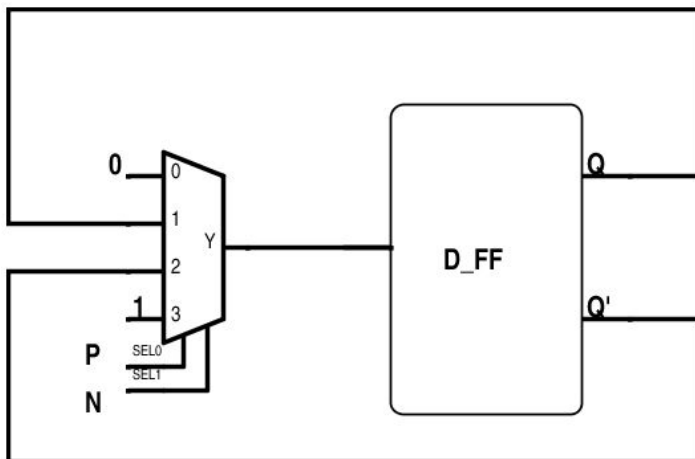
$$Q(t+1) = P Q'(t) + N Q(t)$$

c) By assigning $D=P=Q$:

- if $D=1$, then $Q(t+1) = 1 = D$
- if $D=0$ then $Q(t+1) = 0 = D$



d)



6) In this question, we want to design a ripple style 4bit comparator. The comparator receives two 4bit inputs A, B, and generates 3 single bit outputs EQ, GT, LT as follows:

- EQ is one if and only if A and B are equal.
 - GT is one if and only if A is greater than B.
 - LT is one if and only if A is less than B.
- a) Design a single bit comparator that receives single bit inputs a and b, and generate eq, gt and lt. Draw the truth table and circuit.
- b) Design a full comparator that can be used in ripple style. The full comparator receives single bit inputs a, b, gt_in, eq_in, lt_in and generates single bit outputs gt_out, eq_out, lt_out. Draw the truth table and circuit.

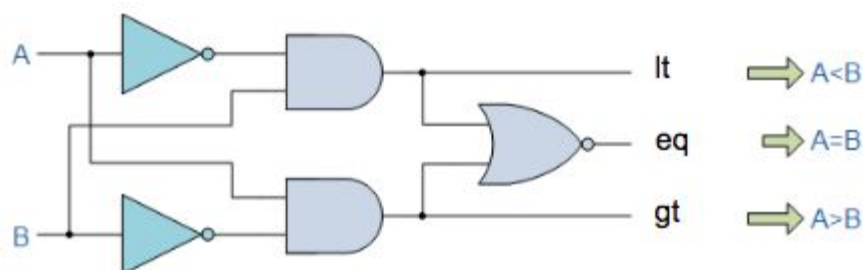
Note that at this point you need to decide if the 4 bit comparator you are designing ripples from msb bits to lsb bit or the other way around and design the full comparator accordingly. The circuit receives 5 inputs which mean the truth table has 32 lines! Note that there are many dont care conditions since only one of the inputs (GT, EQ, LT) can be one at the same time. You need to consider these dont care conditions to draw a small truth table and get the final equations.

c) Using the circuit from part b, design a 4 bit ripple comparator. You can use the full comparator from part b as a black box.

a) Truth table for single bit comparator.

A	B	GT	EQ	LT
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Circuit for single bit comparator



b) Truth table for full comparator

GTin	EQin	LTin	A	B	GTout	EQout	LTout
1	x	x	x	x	1	0	0

x	1	x	0	0	0	1	0
x	1	x	0	1	0	0	1
x	1	x	1	0	1	0	0
x	1	x	1	1	0	1	0
x	x	1	x	x	0	0	1

Equations:

$$GTout = GTin + EQin (AB')$$

$$EQout = EQin (AB + A'B')$$

$$LTout = LTin + EQin (A'B)$$

c)Ripple style four bit comparator

