Linear fn check : <https://gateoverflow.in/8294/gate-cse-2015-set-1-question-39?show=37120#c37120>

Emil post functional completeness : <https://gateoverflow.in/8294/gate-cse-2015-set-1-question-39?show=8532#a8532> [all the 5 steps]

Imp I always forget:-

1. IN S/R or J/K -> [TRICK : “When inputs are different, the Q (ouput) = whatever bit is in S/J”

If S or J = 0 & R or K =1 🡪 Q = S = 0 (RESET)

If S or J = 1 & R or K =0 🡪 Q = S = 1 (RESET)

------------------------------------------------------------

1. IN Booth’s Bit pair multiplier:- [TRICK: “Think 0 = +ve and 1 = -ve, so 01 = +1 and 10 = -ve”]

Initially take q-1 = 0

Then if bit pair = 0 1 🡪 +1 (ADD & ASR)

If Bit Pair = 1 0 🡪 -1 (SUB AND ASR)

Also remember the no.s are taken from LSB to MSB i.e q-1, q0, q1, q2 …, So while writing the recoded multiplier report answer from MSB to LSB (which is very natural)

------------------------------------------------------------------------------------------------

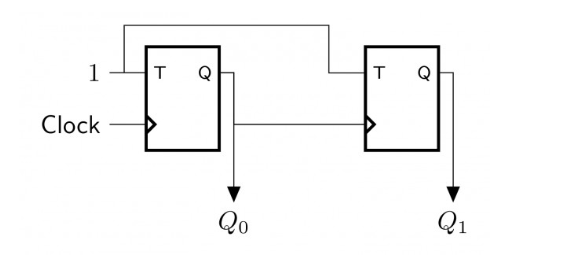
1. Overflow Expression : XYZ’ + X’Y’Z {VERY IMP} {X = MSB of 1st number, Y = MSB of 2nd number, Z = MSB of Result}

Note:-

* If opposite signed data added 🡪 Overflow Never Occurs
* If same signed data (X=Y) added and result (Z) is of different sign then definitely overflow occurred. Determine as:-
  + If X = Y = 1 & Z = 0 🡪 Means two negative no. added so result is negative. Hence take the carry and place it at the first and determine the result accordingly in 2’s complement form (Sign is negative, then convert entire no with 2’s complement method and output the result with sign)
  + If X = Y = 0 & Z = 1 🡪 Means two positive no. added so result is positive. No need to convert into 2’s completement form as it’s already +ve

----------------------------------------------------------------------------------------------------------------------------------

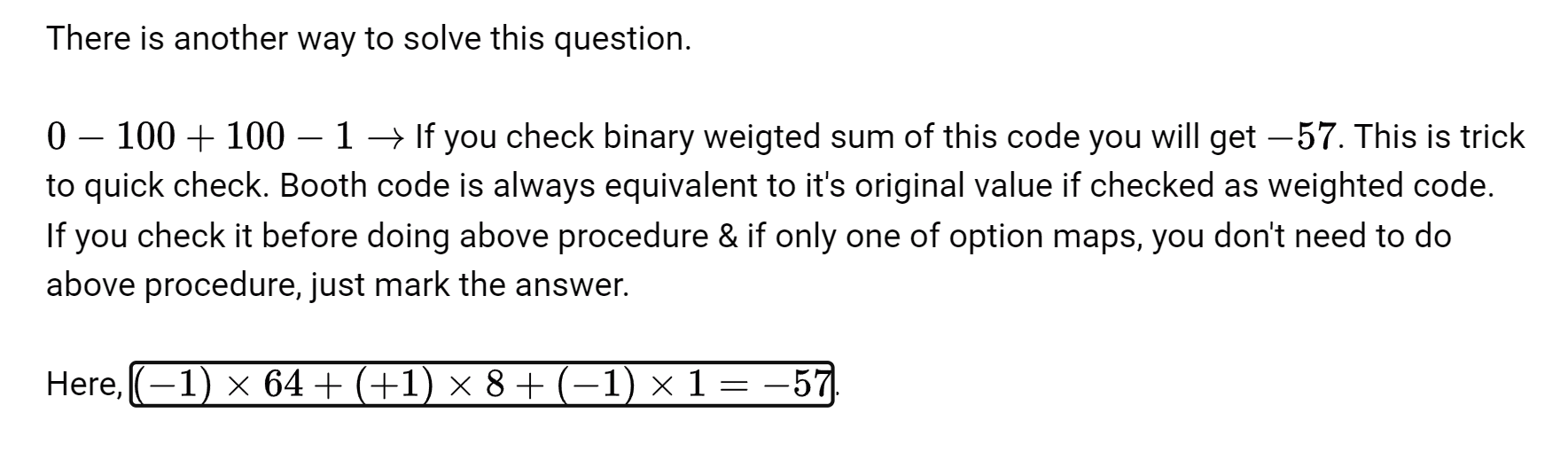
1. +Ve Edge Triggered vs -Ve Edge Triggered :-

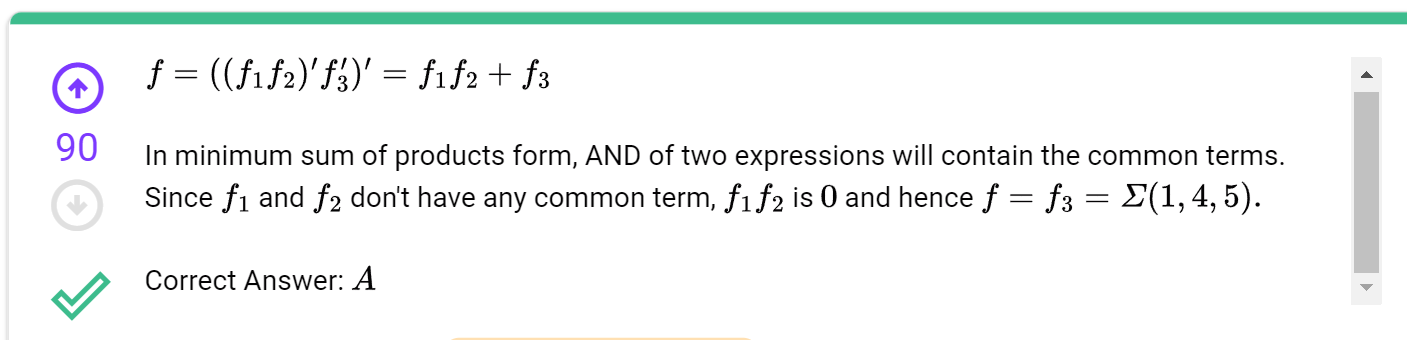
This is a +ve edge triggered counter b/c the arrow sign in Clock has no bubble. Also it is asynchronous b/c every clk depends on o/p of other flip flop. Thus we need to apply the concept of edge trigger clocking.

Remember : In +ve edge trigger, do like this (with ref to this diag):

Q1n = Q1’ [only when Q0 goes from 0 -> 1]

Q0n = Q0’ (b/c it’s independent of other ff clock)





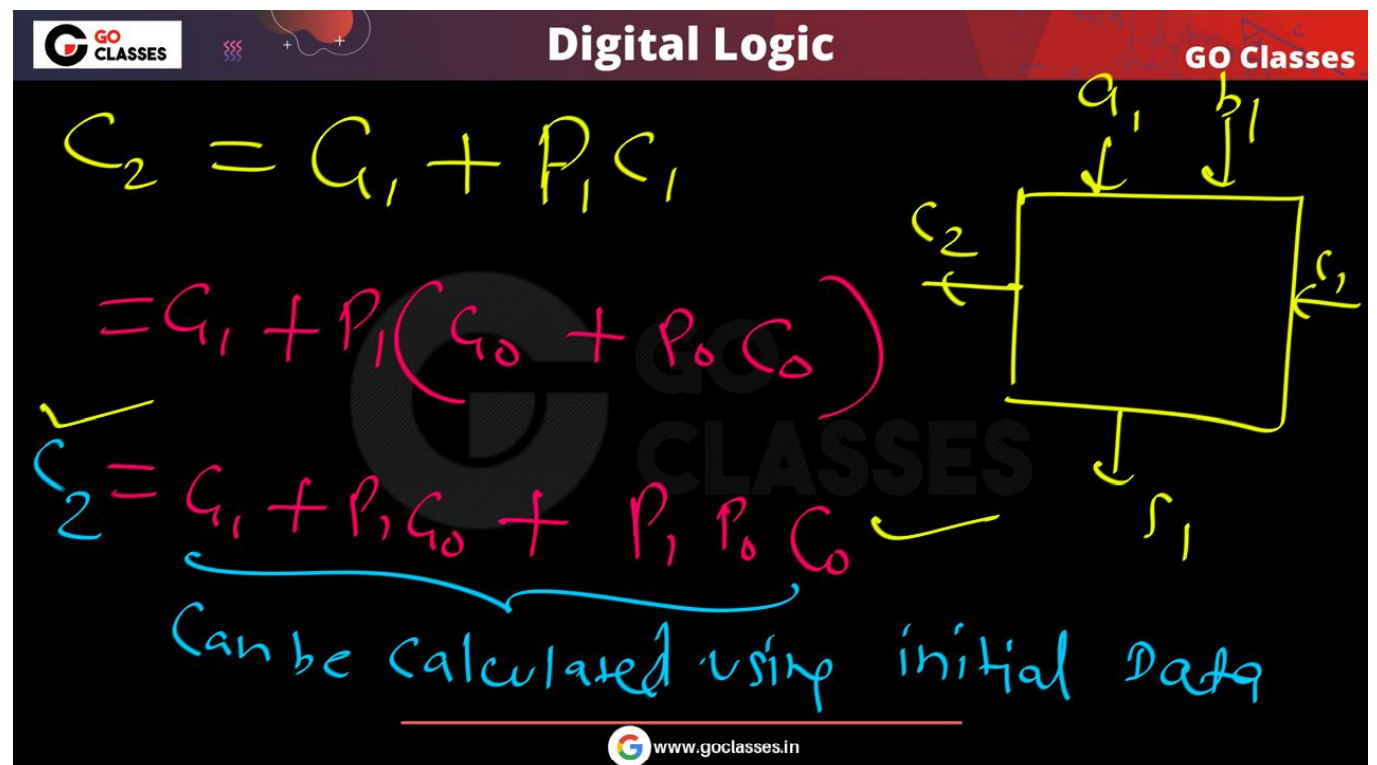
Must go through q: [Digital Logic: GATE CSE 2008 | Question: 8 (gateoverflow.in)](https://gateoverflow.in/406/gate-cse-2008-question-8)

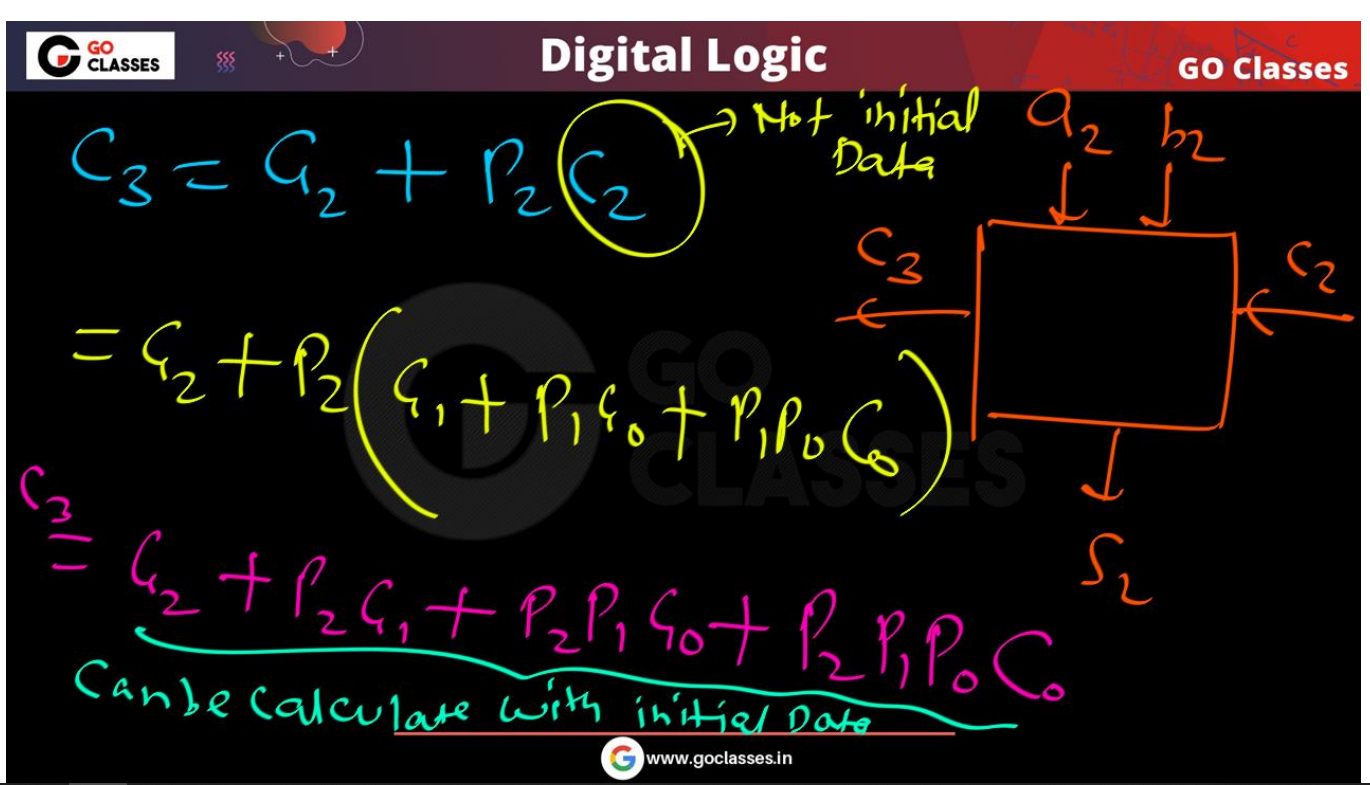
[Digital Logic: prime implicants (gateoverflow.in)](https://gateoverflow.in/66719/prime-implicants) [Very good question]

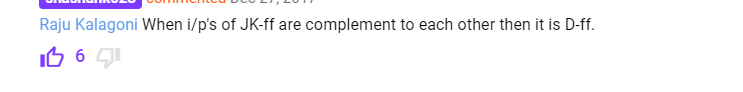
OR AND(POS) realization is implemented by NOR gates

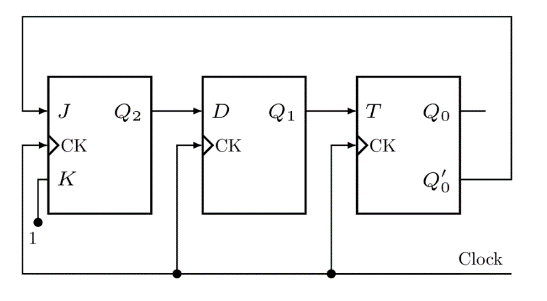
AND OR(SOP) is implemented by NAND

Do this: [Digital Logic: GATE CSE 2019 | Question: 50 (gateoverflow.in)](https://gateoverflow.in/302798/gate-cse-2019-question-50)







Important:-

The Equations will be as follows:-

Since K=1, thus Qn = 0 if J =0

Qn = Q’ if J = 1

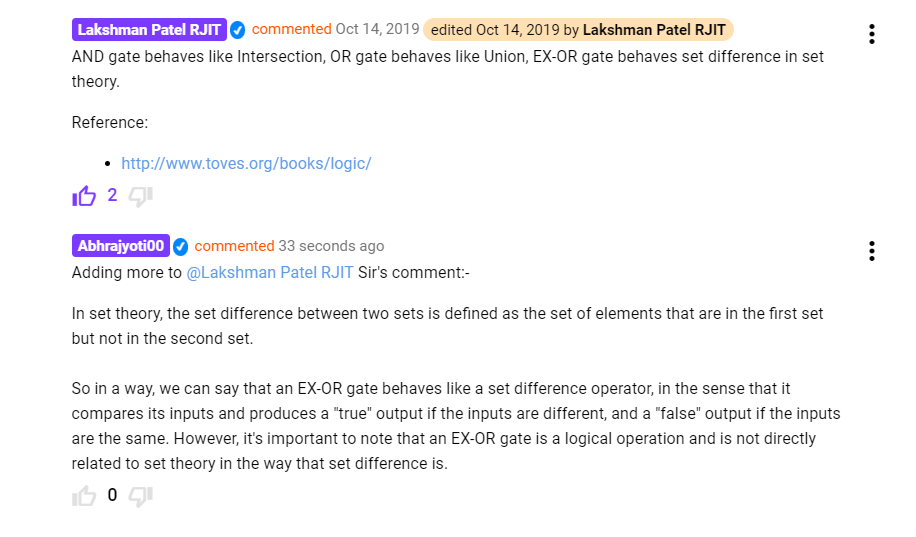
Now Q2n = Q2’ if Q0’ = J2 = 1; Which means Q2n = Q2’ if Q0 = 0

Q2n = 0 if Q0’ = J2 = 0; Which means Q2n = 0 if Q0 = 1

Q1n = Q2

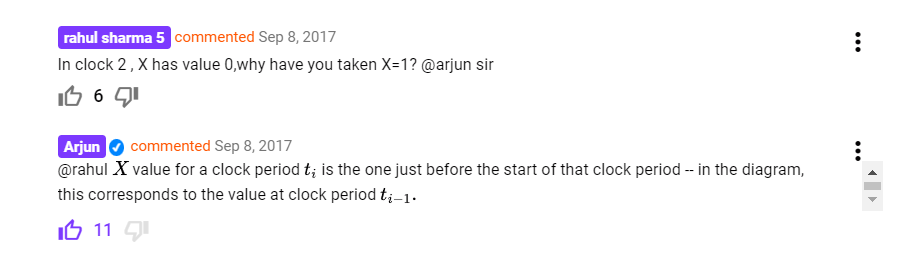
Q0n = Q0’ if Q1 =1

Q0n = Q0 if Q1 =0

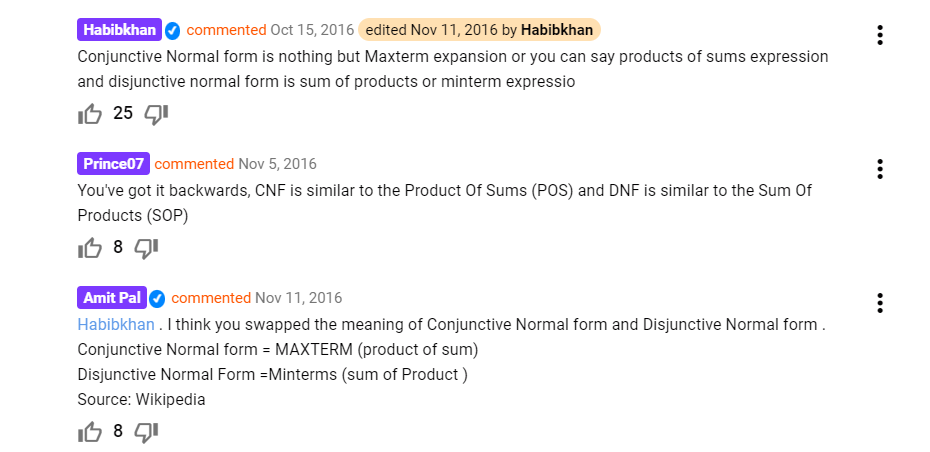


Imp Q: [Digital Logic: GATE CSE 2005 | Question: 62 (gateoverflow.in)](https://gateoverflow.in/264/gate-cse-2005-question-62)

Some points regarding this q:

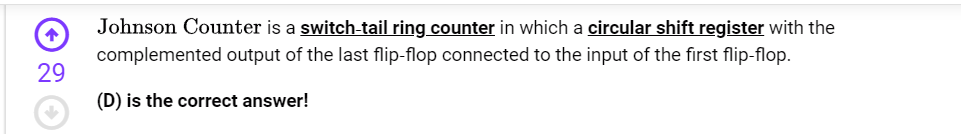


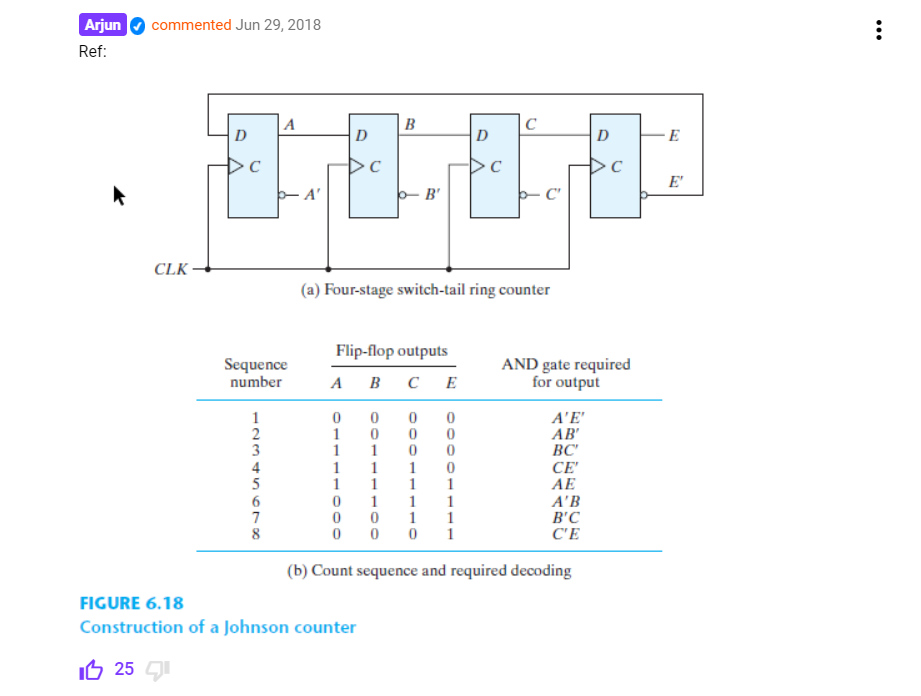
See now that I have conquered the art of typing without looking at the keyboard. I’m still now not very perfect but I’m trying to be perfect. 😊



This answer is lit: [Digital Logic: GATE CSE 2002 | Question: 7 (gateoverflow.in)](https://gateoverflow.in/860/gate-cse-2002-question-7)

One of the Best Q on timing: [Digital Logic: GATE CSE 2003 | Question: 47 (gateoverflow.in)](https://gateoverflow.in/29098/gate-cse-2003-question-47)





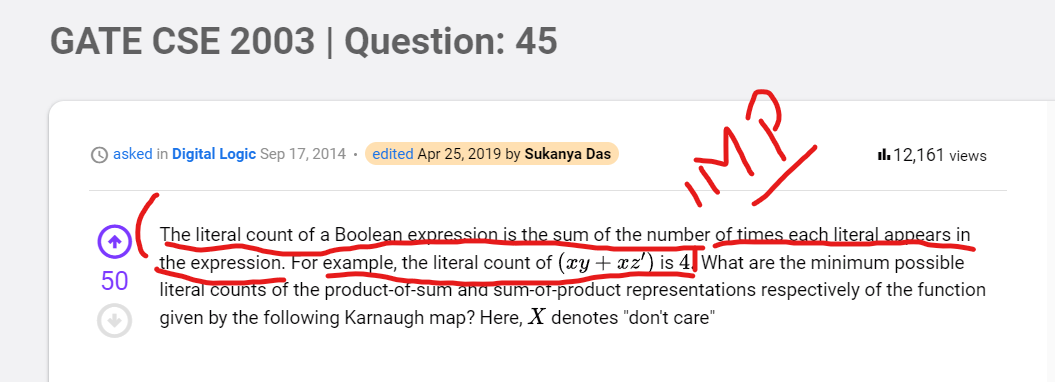
Converting fraction no.s in decimal system to binary : [Convert 0.239 to 32 Bit Single Precision IEEE 754 Binary Floating Point Standard, From a Base 10 Decimal Number = 0 - 0111 1100 - 111 0100 1011 1100 0110 1010 (base-conversion.ro)](https://binary-system.base-conversion.ro/real-number-converted-from-decimal-system-to-32bit-single-precision-IEEE754-binary-floating-point.php?decimal_number_base_ten=0.239) [Imp, GATE 05]

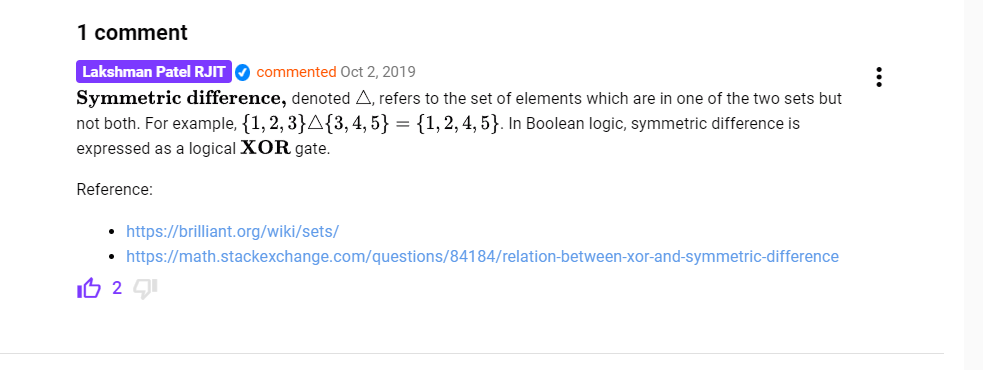
Redo: [Digital Logic: GATE CSE 2005 | Question: 85-b (gateoverflow.in)](https://gateoverflow.in/82139/gate-cse-2005-question-85-b)



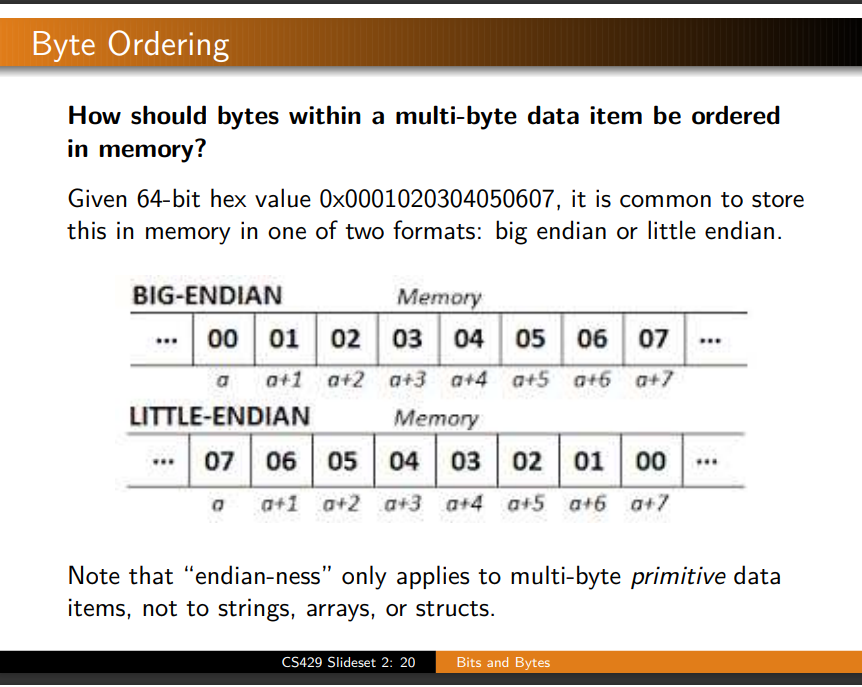
Ref: <http://steve.hollasch.net/cgindex/coding/ieeefloat.html>

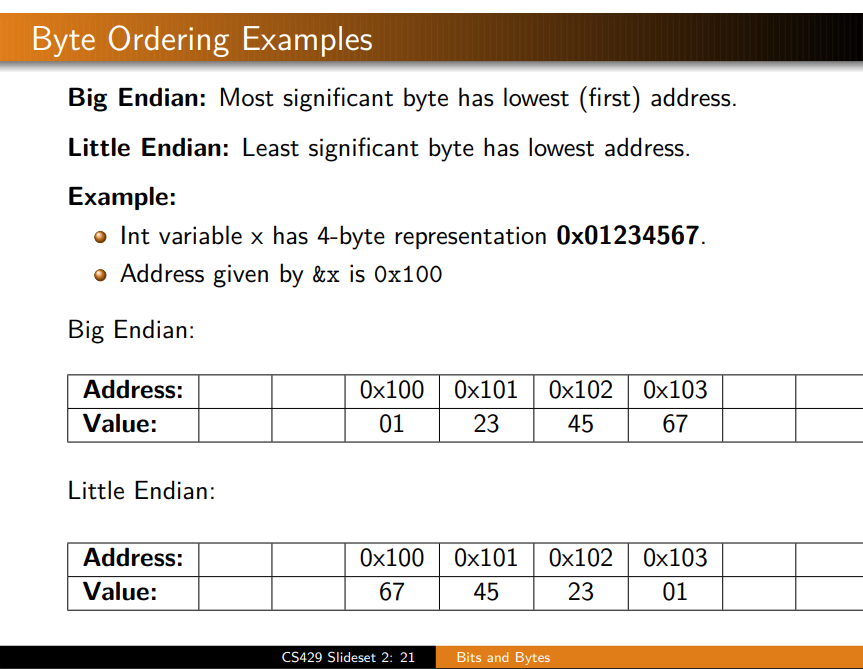
<https://gateoverflow.in/357536>

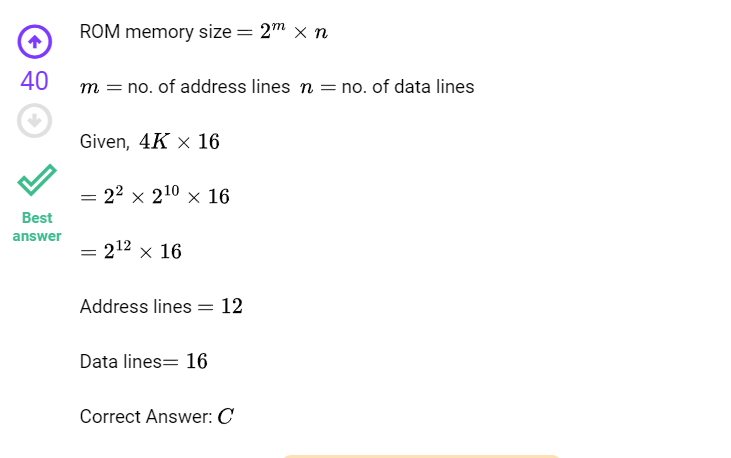


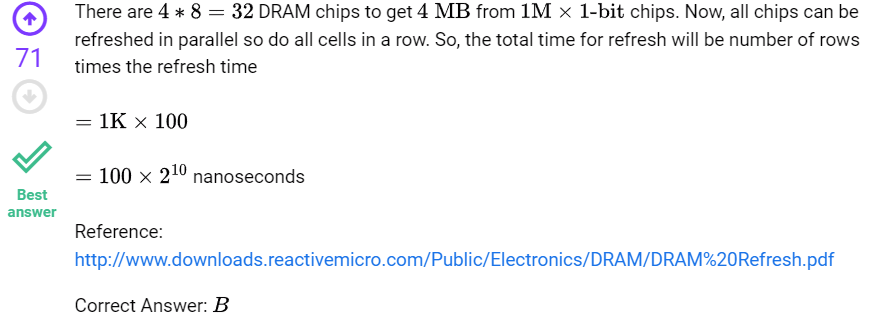


**About Little-Endian and Big-Endian**



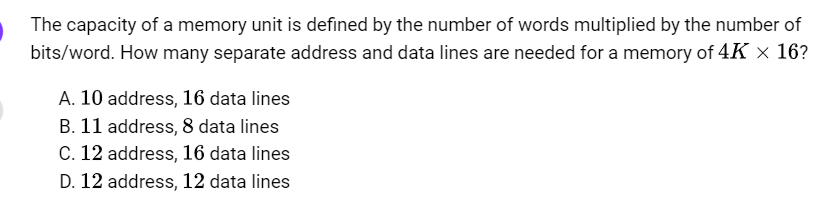






[Digital Logic: GATE IT 2005 | Question: 9 (gateoverflow.in)](https://gateoverflow.in/3754/gate-it-2005-question-9)

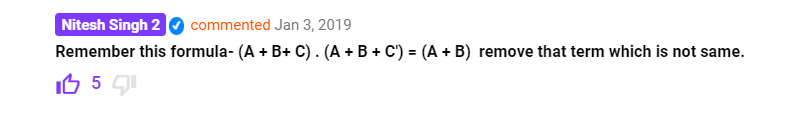
[Digital Logic: GATE CSE 1995 | Question: 2.2 (gateoverflow.in)](https://gateoverflow.in/2614/gate-cse-1995-question-2-2) Solution Technique:-



IMP concept: always remember #DATA LINES = # BITS/cell = 16 (here). This is because each data line carries one bit information. So if a cell has 16 bits, it will require those many data lines.

#ADDRESS LINES = log(#Cells) = log(4K) = 12

An awesome method designing circuits using min no of NAND/NOR GATES: <https://gateoverflow.in/1298/gate-cse-2009-question-6?show=327888#a327888>



[Digital Logic: GATE CSE 2018 | Question: 49 (gateoverflow.in)](https://gateoverflow.in/204124/gate-cse-2018-question-49) \*\*\* V.v.imp q

