EE 330 HW 11 Solutions

Spring 2024

Problem 1

$A_V = -g_m R_L = -3 \Rightarrow \mu C_{ox} \frac{W}{L} V_{EB} R_L = 3$

So $\frac{W}{L} = \frac{3}{\mu C_{ox} R_L V_{EB}}$ but $V_{EB} = +1.25 \Rightarrow \frac{W}{L} = \frac{3}{10^{-4}\, 10^{4}\, 1.}$

$\frac{W}{L} = 2.4$

$\therefore$ Let $L = 1\mu$, $W = 2.4\mu$

Problem 2 Observe $V_{GT} = .8V$, $I_{GT\text{-}MAX} = 200\mu A$, $V_{FON} \simeq .9V$

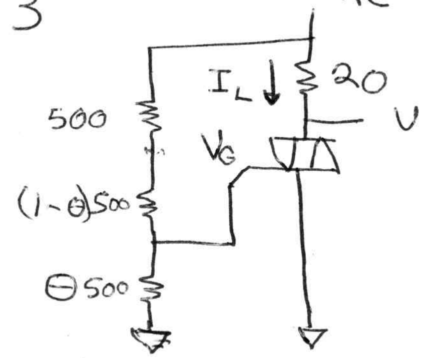
a) $\frac{12 - 0.8V}{R_{GG}} > 200\mu A \Rightarrow R < \frac{11.2V}{200\mu A} = 56K\Omega$

b) $I_F = \frac{50V - 0.9V}{40\Omega} = 1.23A \Rightarrow P = (1.23A)(.9V) = 1.11W$

c) $P_{GATE} \simeq (.8V)(200\mu A) = 160\mu W$

Problem 3

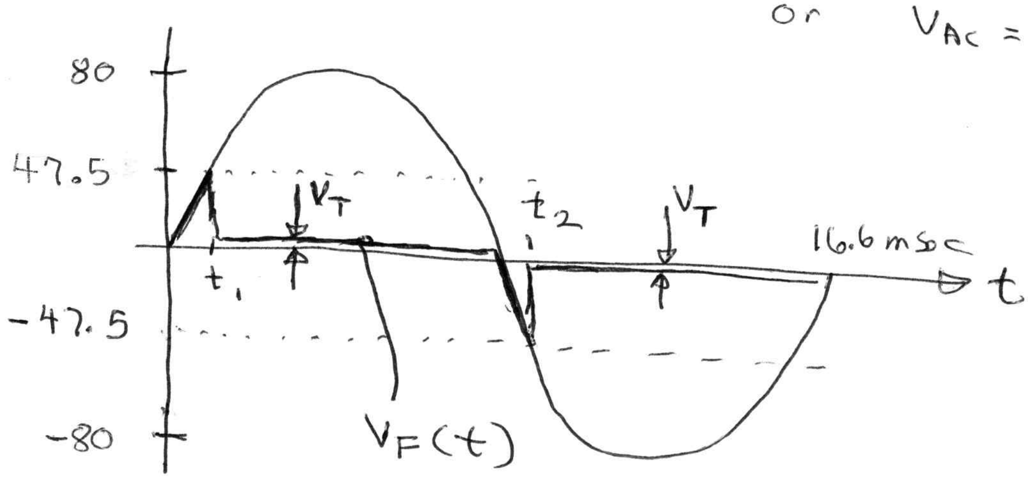
$V_{AC} = 80\sin(2\pi \cdot 60t)$, $V_{GT} = 2.5V$



a) $V_{G}\big|_{\theta=.1} = \dfrac{50}{450+500} V_{AC} = .053 V_{AC}$

will trigger when $V_G = V_{GT} \Rightarrow (.053)V_{AC} = 2.5$

or $V_{AC} = \dfrac{2.5}{.053} = 47.5V$



From figure in data sheet, since $|I_L| < \dfrac{80}{20\Omega} = 4A$,

$V_T \simeq 0.9V$

To find t_1, $80\sin(2\pi \cdot 60t_1) = 47.5V$

$(2\pi)\cdot 60t_1 = \sin^{-1}\left(\dfrac{47.5}{80}\right)$

$t_1 = 1.67 msec$

And $t_2 = t_1 + 8.3msec = 9.97msec$

b) $I_L \simeq \begin{cases} 0 & 0 \le t \le t_1 \\ \dfrac{V_{AC}}{20} & t_1 < t < 8.3msec \end{cases}$

will average power over ½ period which is 8.3 msec

$$P_{AVG} = \frac{1}{8.3\,msec}\int_0^{8.3\,msec}(V_F)(I_L)\,dt = \frac{1}{8.3\,msec}\left[\int_0^{1.67\,msec}(V_{AC}\cdot 0)\,dt + \int_{1.67\,msec}^{8.3\,msec} V_T\,\frac{V_{AC}}{20}\,dt\right]$$

$$= \frac{1}{8.3\,msec}\int_{1.67\,msec}^{8.3\,msec}(0.9V)\,4\sin(2\pi\cdot 60t)\,dt$$

$$= \frac{3.6}{8.3\,msec}\int_{1.67\,msec}^{8.3\,msec}\sin(2\pi\cdot 60t)\,dt = 2.1W$$

c) Quadrants 1 & 3

Problem 4 $V_{GT} = 2V$ want $\left.\frac{R_1}{R_1+10K}(170\sin\omega t)\right|_{t=\frac{T_S}{8}} = 2V$

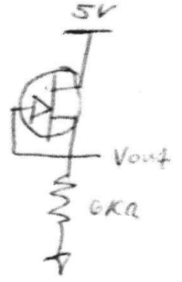
where $\omega = (2\pi)60$ and $T_S = \frac{1}{60}$

Thus $\frac{R_1}{R_1+10K}\,170\sin\left(2\pi(60)\frac{1}{(8)(60)}\right) = 2V$

$$\left(\frac{R_1}{R_1+10K}\right)(170)\sin(\pi/4) = 2V$$

solving obtain $R_1 = 169\Omega$

Problem 5



Assuming JFET is in saturation

$V_{GS} = 0$

$$I_D = I_{DSS}\cdot\left(1-\frac{V_{GS}}{V_P}\right)^2$$

$I_D = 100uA$

$V_{out} = I_D\cdot R = 100uA\cdot 6K\Omega = \boxed{0.6V}$

Problem 6

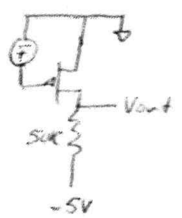
$$g_m = \frac{\partial I_D}{\partial V_{GS}} = -2 \cdot \frac{I_{DSS}?}{-V_P}\left(1 - \frac{V_{GS}}{V_P}\right)(1 - \lambda V_{DS})$$

$$g_m = 2 \cdot \frac{I_{DSSPO}}{V_P}\left(\frac{W}{L}\right)\left(1 - \frac{V_{GS}}{V_P}\right)\Bigg|_Q = \frac{2 I_{DQ}}{V_P\left(1 - \frac{V_{GSQ}}{V_P}\right)}$$

$$g_o = \frac{\partial I_D}{\partial V_{DS}} = \lambda \cdot I_{DSSPO}\left(\frac{W}{L}\right)\left(1 - \frac{V_{GS}}{V_P}\right)^2$$

7)

Problem 7



Observe $V_{GSQ} = 0V$

$$I_{DQ} = \frac{30u \cdot 10}{15} \cdot \left(1 - \frac{0}{1}\right)^2 = 20uA$$

$$I_{DQ} = \frac{V_{outQ} - (-5)}{50k\Omega} = 20uA \Rightarrow V_{out} = -4V$$

$$A_V = \frac{V_{out}}{V_{in}} = -g_m \cdot 50k\Omega = \boxed{-2\ \tfrac{V}{V}}$$

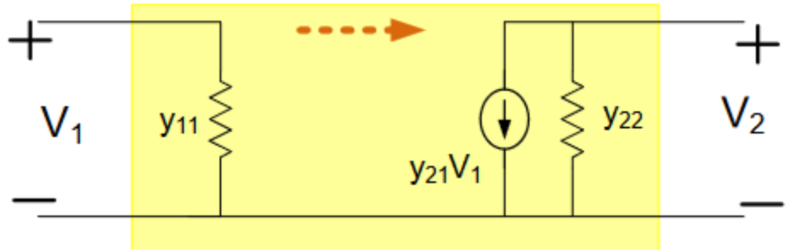
$$+g_m = \frac{2}{V_P} I_{DQ}$$

8)

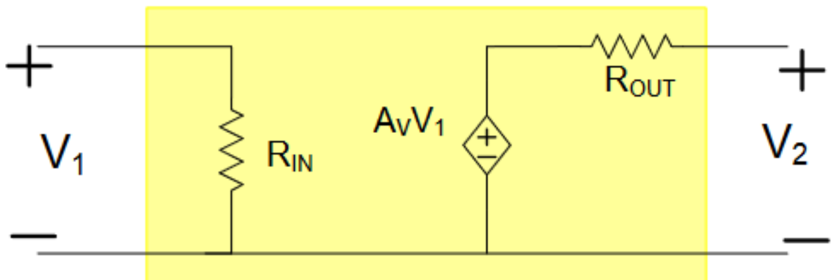
a) This amplifier is unilateral. This can be determined by the lack of a y_{12} parameter, which means that $A_{VR} = 0$. Having an A_{VR} (or A_V with relabeled ports) of 0 is a property of unilateral amplifiers.

b) This model can be developed using diagrams given in lecture slides

Unilateral amplifiers:



- Thevenin equivalent output port often more standard
- R_{IN}, A_V, and R_{OUT} often used to characterize the two-port of amplifiers

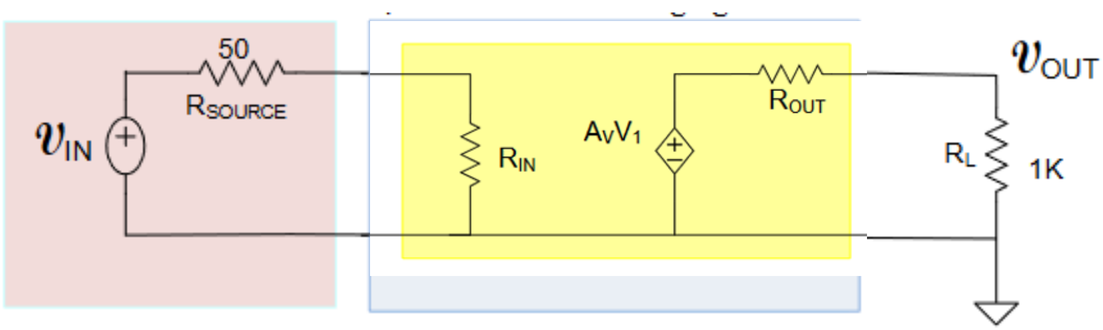


Unilateral amplifier in terms of “amplifier” parameters

$$R_{IN} = \frac{1}{y_{11}} \qquad A_V = -\frac{y_{21}}{y_{22}} \qquad R_{OUT} = \frac{1}{y_{22}}$$

From this description, $R_{IN} = \frac{1}{10^{-4}\ A/V} = 10k\Omega$, $A_V = -\frac{-10A/V}{0.1A/V} = 100$, $A_{VR} = \frac{0}{10^{-4}A/V} = 0$, and $R_{OUT} = \frac{1}{0.1A/V} = 10\Omega$.

c) While we found A_V in the previous step, this is not the actual gain of the given circuit. We'll perform some simple analysis to find the actual gain.

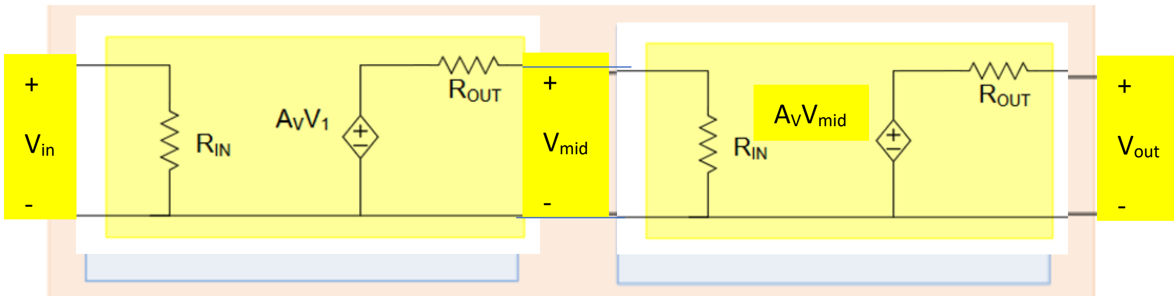


$$V_1 = V_{in}\left(\frac{R_{in}}{R_{source} + R_{in}}\right) = V_{in}\left(\frac{10000}{10050}\right) = .9950V_{in}$$

$$V_{out} = A_V V_1\left(\frac{R_l}{R_{out} + R_l}\right) = (100)(.9950)V_{in}\left(\frac{1000}{1010}\right) = 98.5173V_{in}$$

$$\frac{V_{out}}{V_{in}} = 98.5173$$

d) We'lll break these amplifiers up into stages and relate V_{IN} to V_{OUT} based on the intermediary node to find gain. R_{in} and R_{out} won't change for this amplifier



$$V_1 = V_{in}$$

$$V_{mid} = A_v V_1 \left(\frac{R_{in}}{R_{out} + R_{in}}\right) = (100)(V_{in})\left(\frac{10000}{10010}\right) = 99.9001 V_{in}$$

$$V_{out} = A_v V_{mid} \left(\frac{R_l}{R_{out} + R_l}\right) = (100)(99.9001 V_{in}) = 9990.0099 V_{in}$$

$$\frac{V_{out}}{V_{in}} = 9990.0099 = A_v$$

This leaves us with amplifier parameters $R_{in} = 10k\Omega, A_v = 9990.0099, A_{vr} = 0, and\ R_{out} = 10\Omega$

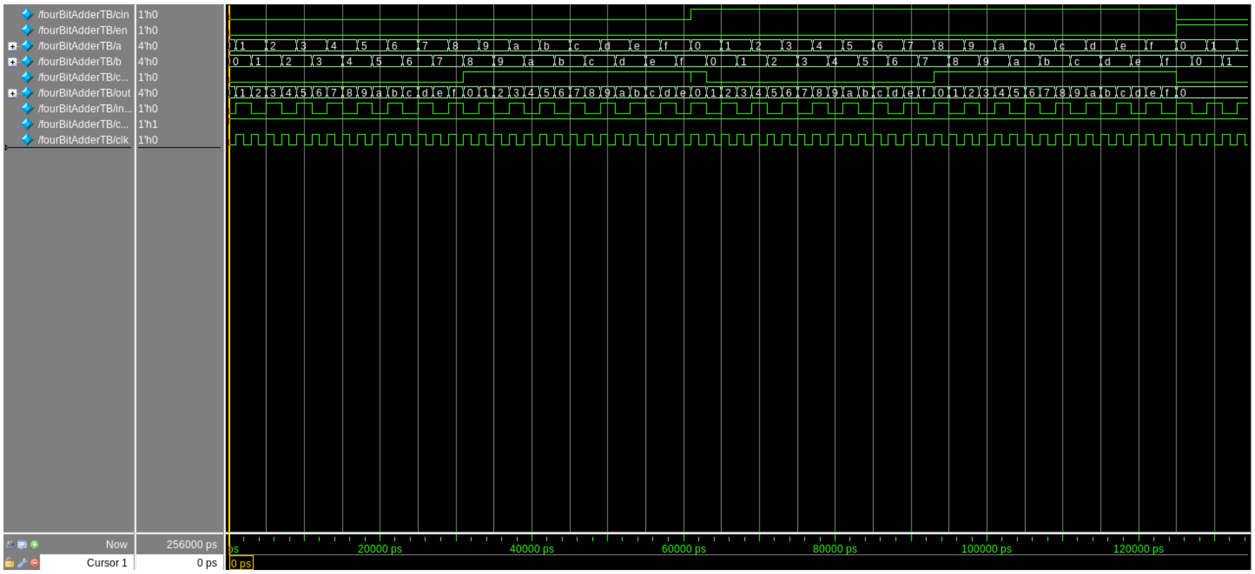
9&10) Four-Bit Adder Module Code

```
`timescale 1ns/1ps                                    //nice simulation timescale

module fourBitAdder(a, b, cin, en, out, cout);  //instantiate module
        input cin,en;                                 //instantiate 1 bit inputs
        input [3:0] a;                                //instantiate 4 bit inputs
        input [3:0] b;
        output cout;                                  //instantiate one bit output
        output [3:0] out;                             //instantiate four bit output
        reg [4:0] sum;                                //instantiate a sum register
        assign out = sum[3:0];                        //assign output to first four bits
        assign cout = sum[4];                         //assign carry-out to last bit
        always @ (*) begin                            //any time an input changes
                if (!en) begin                        //if device is enabled
                        sum <= a+b+cin;               //calculate sum
                end
                else begin                            //if device is disabled
                        sum <= 0;                     //output is zero
                end
        end
endmodule
```

Four-Bit Adder Testbench Code

```
`timescale 1ns/1ps                                          //nice simulation timescale

module fourBitAdderTB();                                    //instantiate testbench
        reg cin, en;                                        //create inputs
        reg [3:0] a;
        reg [3:0] b;
        wire cout;                                          //create outputs
        wire [3:0] out;
        reg incr, count_up, clk;                            //testbench-specific variables
        fourBitAdder DUT(.a(a),.b(b),.cin(cin),.en(en),.out(out),.cout(cout));
        initial clk = 0;                                    //give variables initial values
        initial a = 0;
        initial b = 0;
        initial cin = 0;
        initial incr = 0;
        initial en = 0;
        always #1 clk <= ~clk;                              //toggle clock every time unit
        initial count_up = 1;                               //create incrementing process
        always @ (posedge clk) begin                        //on rising clock edge
                if (!incr) begin                            //if a is to be incremented
                        if (a == 4'b1111) begin             //if a is maxed out
                                a = 4'b0000;                //reset to zero
                                incr = 1;                   //increment b on next clock cycle
                        end
                        else begin                          //if a is not maxed out
                                a <= a + count_up;          //increment a
                                incr = 1;                   //increment b on next clock cycle
                        end
                end
                else if (incr) begin                        //if b is to be incremented
                        if (b == 4'b1111) begin             //if b is maxed out
                                b = 4'b0000;                //reset to zero
                                incr = 0;                   //increment a on next clock cycle
                        end
                        else begin                          //if b is not maxed out
                                b <= b + count_up;          //increment b
                                incr = 0;                   //increment a on next clock cycle
                        end
                end
                if (a==0 && b==4'b1111) begin               //on one specific input combination
                        cin <= ~cin;                        //toggle carry-in bit
                end
                if (a==0 && b==4'b1111 && cin) begin        //on one specific input combination
                        en <= ~en;                          //toggle enable
                end
        end
endmodule
```

Four Bit Adder Waveforms



Proves function by

- Adds A+B+C_{IN}
- Output is sum and C_{OUT}
- Output is zero when en is high