คำนำ

รายงานนี้นำเสนอการออกแบบวงจร Op Amp แบบ Two-Stage Amplifier โดยให้ใช้โปรแกรม LTSpice ในการจำลองและทดสอบวงจรไฟฟ้า และใช้โมเดล CMOS แบบ BSIM3 เทคโนโลยี 0.18 nm ในการออกแบบวงจรดังกล่าว

ณัฐชนน จริยานุรัตน์

สารบัญ

บทที่ 1

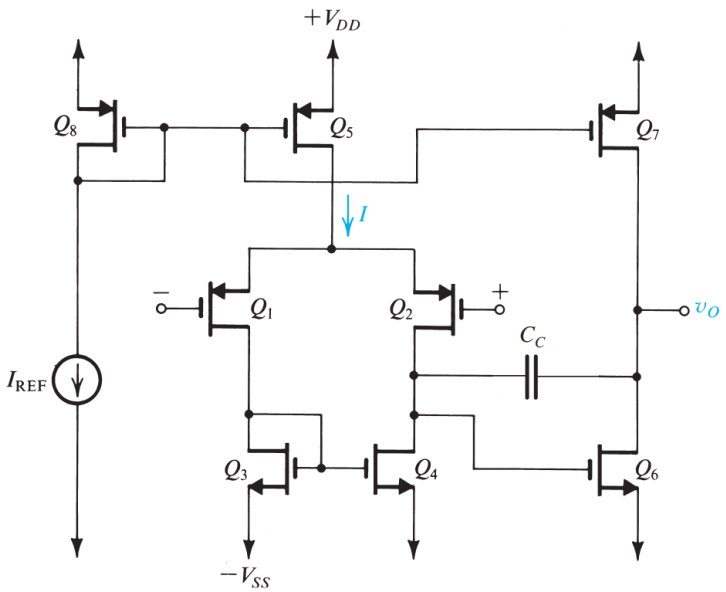
รายละเอียดงาน

1.1 การออกแบบวงจร Op Amp แบบ Two-Stage Amplifier

Op Amp เป็นส่วนประกอบที่ใช้บ่อยในวงจรอิเล็กทรอนิกส์ต่าง ๆ สำหรับ Op Amp ที่สามารถออกแบบได้ง่ายและเป็นพื้นฐานมากที่สุด คือวงจรแบบ Two-Stage Amplifier

1.1.1 ทฤษฎีที่เกี่ยวข้อง

โครงสร้างทั่วไปของวงจร Two-Stage Amplifier



รูปที่ 1.1: โครงสร้างทั่วไปของวงจร two-stage CMOS op-amp (จาก Sedra and Smith, Microelectronic Circuits, 7E [5])

จากรูปที่ 1.1 จะเห็นได้ว่า วงจร Two-Stage Amplifier จะประกอบไปด้วยสามส่วนหลัก ได้แก่

- วงจร Differential Amplifier ซึ่งประกอบไปด้วยทรานซิสเตอร์คือ Q1, Q2, Q3 และ Q4 ซึ่งทำหน้าที่ขยาย

สัญญาณในส่วนแรก โดยสัญญาณขาเข้าจะเข้ามาทางสัญลักษณ์ + และ - ในรูป

- หลังจากนั้น สัญญาณขาออกจากวงจร Differential Amplifier จะไปเข้าสู่วงจร Common-Source Amplifier ซึ่งประกอบไปด้วยทรานซิสเตอร์ Q6 ซึ่งทำหน้าที่ขยายสัญญาณในขั้นสุดท้าย
- วงจร Current Mirror ประกอบไปด้วยทรานซิสเตอร์ Q5, Q7 และ Q8 ซึ่งสร้างกระแสอ้างอิงให้กับทั้งวงจร Differential Amplifier และวงจร Common-Source Amplifier

เงื่อนไขของการไบแอสวงจร

เนื่องจาก drain-source voltage ของทรานซิสเตอร์ Q3, Q4 และ Q6 จะต้องเท่ากัน ดังนั้นกระแสที่ผ่านทานซิสเตอร์ทั้งสามตัวจะต้องเท่ากันด้วย เนื่องจากกระแสที่ผ่านทรานซิสเตอร์ Q3, Q4 จะเป็นครึ่งหนึ่งของกระแสที่ผ่านทรานซิสเตอร์ Q5 ดังนั้น อัตราส่วน W/L ของทรานซิสเตอร์ในรูปที่ 1.1 จึงต้องมีค่าเป็น

$$\frac{(W/L)_6}{(W/L)_4} = 2\frac{(W/L)_7}{(W/L)_5} \tag{1.1}$$

ช่วงของสัญญาณขาเข้า (Input Common-Mode Range) และสัญญาณขาออก (Output Swing)

สำหรับช่วงของสัญญาณขาเข้า (Input Common-Mode Range) และสัญญาณขาออก (Output Swing) ของวงจร two-stage CMOS op-amp เป็นดังนี้

ช่วงของสัญญาณขาเข้า (Input Common-Mode Range):

$$\text{lower limit} = -V_{SS} + V_{OV3} + V_{tn} - |V_{tp}| \tag{1.2a}$$

$$\text{upper limit} = +V_{DD} - |V_{tp}| - |V_{OV1}| - |V_{OV5}| \tag{1.2b}$$

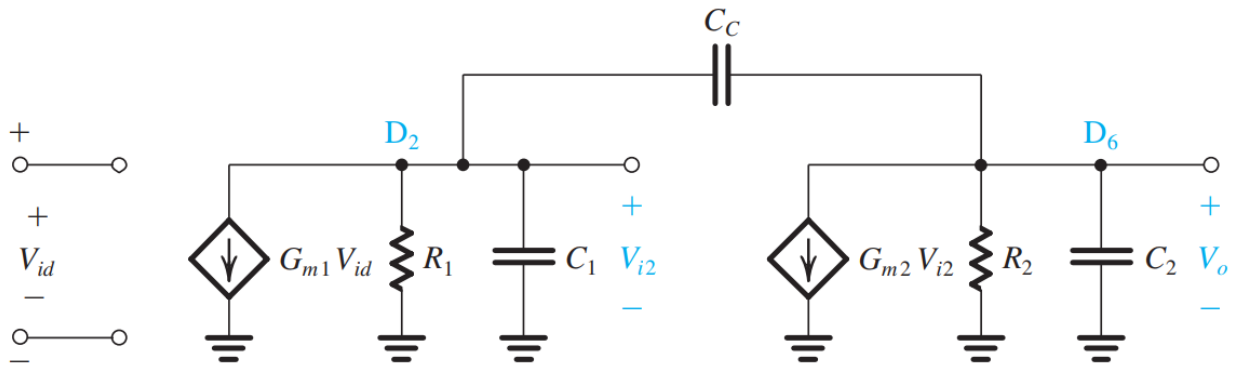
ช่วงของสัญญาณขาออก (Output Swing):

$$\text{lower limit} = -V_{SS} + V_{OV6} \tag{1.3a}$$

$$\text{upper limit} = +V_{DD} - |V_{OV7}| \tag{1.3b}$$

เพื่อทำให้ช่วงของสัญญาณขาเข้าและสัญญาณขาออกกว้างขึ้น การออกแบบวงจรควรพยายามที่จะทำให้กระแสไบแอสมีค่าต่ำ เพื่อทำให้ overdrive voltage ของทรานซิสเตอร์มีค่าต่ำลง

จากสมการ สังเกตว่าขอบบนของ Input Common-Mode Range นั้นถูกทำให้ลดลงอย่างมากจากขนาดของ $|V_{tp}|$ ที่คงที่



รูปที่ 1.2: โมเดลสัญญาณขนาดเล็ก (Small-Signal Model) ของวงจรในรูปที่ 1.1 (จาก Sedra and Smith, Microelectronic Circuits, 7E [5])

โมเดลสัญญาณขนาดเล็ก (Small-Signal Model) ของวงจร

จากการคำนวณต่อจากรูปที่ 1.1 จะได้ว่าโมเดลสัญญาณขนาดเล็กในรูปที่ 1.2 จะมีค่าของตัวแปรขององค์ประกอบต่าง ๆ ดังนี้ คือ

$$G_{m1} = g_{m1} \tag{1.4a}$$

$$G_{m2} = g_{m6} \tag{1.4b}$$

$$R_1 = (r_{o2}||r_{o4}) \tag{1.4c}$$

$$R_2 = (r_{o6}||r_{o7}) \tag{1.4d}$$

อัตราขยายของวงจร

จากโมเดลสัญญาณขนาดเล็กในรูปที่ 1.2 สามารถคำนวณหาอัตราขยายของ stage แรก คือ Differential Amplifier และ stage ที่สอง คือ Common-Source Amplifier ได้ดังนี้

อัตราขยายของ stage แรก:

$$A_1 = -g_{m1}(r_{o2}||r_{o4}) = -\frac{2}{|V_{OV1}|}/(\frac{1}{|V_{A2}|} + \frac{1}{|V_{A4}|}) \tag{1.5}$$

อัตราขยายของ stage ที่สอง

$$A_2 = -g_{m6}(r_{o6}||r_{o7}) = -\frac{2}{|V_{OV6}|}/(\frac{1}{|V_{A6}|} + \frac{1}{|V_{A7}|}) \tag{1.6}$$

หากต้องการอัตราขยายของวงจรให้มากขึ้น สามารถทำได้โดยการลดกระแสไบแอสของทรานซิสเตอร์ลง จะทำให้ overdrive voltage ของทรานซิสเตอร์มีค่าลดลงตาม และทำให้อัตราขยายมากขึ้น (แต่จะแลกมาด้วยความต้านทานของสัญญาณรบกวนที่แย่ลง) ในขณะเดียวกัน การเพิ่ม Early voltage ของทรานซิสเตอร์โดยการเพิ่มความยาว (L) ของทรานซิสเตอร์ก็สามารถทำให้อัตราขยายของวงจรเพิ่มขึ้นเช่นกัน (แต่จะแลกมาด้วยขนาดวงจรที่เพิ่มขึ้น และวงจร

จะใช้กระแสมากขึ้น ทำให้พลังงานที่ต้องใช้ในวงจรมากขึ้น)

การตอบสนองเชิงความถี่ของวงจร

จากรูปที่ 1.2 จะได้ว่า C_1 คือความจุไฟฟ้าทั้งหมดตั้งแต่ขาออกของวงจรขยาย stage แรก จนถึง ground ของวงจร ดังนั้นจึงได้ว่า

$$C_1 = C_{gd2} + C_{db2} + C_{gd4} + C_{db4} + C_{gs6} \tag{1.7}$$

เช่นเดียวกัน จะได้ว่า C_1 คือความจุไฟฟ้าทั้งหมดตั้งแต่ขาออกของวงจรขยาย stage ที่สอง จนถึง ground ของวงจร ดังนั้นจึงได้ว่า

$$C_2 = C_{db6} + C_{db7} + C_{gd7} + C_L \tag{1.8}$$

โดยตัวแปรของความจุไฟฟ้าต่าง ๆ ในสมการ 1.7 และ 1.8 มีความหมายดังนี้ คือ
C_L คือความจุไฟฟ้าของโหลดที่ต่อกับวงจรขาออก
C_{gd} คือความจุไฟฟ้าของทรานซิสเตอร์ระหว่างขา gate และขา drain
C_{gs} คือความจุไฟฟ้าของทรานซิสเตอร์ระหว่างขา gate และขา source
C_{sb} คือความจุไฟฟ้าของทรานซิสเตอร์ระหว่างขา source และขา body
C_{db} คือความจุไฟฟ้าของทรานซิสเตอร์ระหว่างขา drain และขา body
เลขหลัง subscript คือค่าของทรานซิสเตอร์ตามหมายเลขที่ระบุไว้ในรูปที่ 1.1
ตัวแปรความจุไฟฟ้าต่าง ๆ มีค่าเป็นดังนี้

$$C_{gd} = WL_{ov}C_{ox} \tag{1.9a}$$

$$C_{gs} = \frac{2}{3}WLC_{ox} + WL_{ov}C_{ox} \tag{1.9b}$$

$$C_{sb} = C_{sb0}/\sqrt{1 + \frac{|V_{SB}|}{V_0}} \tag{1.9c}$$

$$C_{db} = C_{db0}/\sqrt{1 + \frac{|V_{DB}|}{V_0}} \tag{1.9d}$$

โดย C_{ox} คือความจุไฟฟ้าที่เกิดจากชั้นออกไซด์, W และ L คือความกว้างและความยาวของทรานซิสเตอร์ ตามลำดับ, V_0 คือ junction built-in voltage ซึ่งปกติมีค่าอยู่ระหว่าง 0.6 - 0.8 V, $L_{ov} \approx 0.05L - 0.1L$ คือความยาวที่ซ้อนกันระหว่าง gate และ source/drain substrate, C_{sb0} และ C_{db0} คือค่าของ C_{sb} และ C_{db} เมื่อ body-source/drain bias มีค่าเป็น 0 V ตามลำดับ

อ้างอิงจาก Sedra and Smith [5] การวิเคราะห์วงจรแบบ node analysis ในรูปที่ 1.2 จะทำให้ได้ transfer function V_o/V_{id} ที่มีขั้ว (pole) ทั้งหมด 2 ตำแหน่ง และมีศูนย์ (zero) ที่อยู่บน right complex plane ทั้งหมดหนึ่ง

ตำแหน่ง ซึ่งมีค่าดังนี้

$$\omega_Z = \frac{G_{m2}}{C_C} \tag{1.10a}$$

$$\omega_{P1} = \frac{1}{R_1[C_1 + C_C(1 + G_{m2}R_2)] + R_2(C_2 + C_C)} \tag{1.10b}$$

$$\omega_{P2} = \frac{G_{m2}C_C}{C_1C_2 + C_C(C_1 + C_2)} \tag{1.10c}$$

โดยที่ขั้ว ω_{P1} จะมีค่าต่ำลงเนื่องจาก C_C ดังนั้น C_C จะช่วยเพิ่ม phase margin ของผลตอบสนองเชิงความถี่โดยการเลื่อน crossover frequency ให้ต่ำลงมาอยู่ที่ช่วงที่มี phase response สูง

การประมาณค่าต่าง ๆ ในสมการที่ผ่านมาจะทำให้ได้ค่าที่จัดการได้ง่ายขึ้น ดังนี้

$$\omega_Z = \frac{G_{m2}}{C_C} \tag{1.11a}$$

$$\omega_{P1} = \frac{1}{R_1C_CG_{m2}R_2} \tag{1.11b}$$

$$\omega_{P2} = \frac{G_{m2}}{C_2} \tag{1.11c}$$

ปัญหาของการลดลงของ phase response เนื่องจากการมีอยู่ของ zero ซึ่งอยู่บน right complex plane สามารถแก้ไขได้โดยการต่อความต้านทาน R แบบอนุกรมกับตัวเก็บประจุ C_C ซึ่งจะทำให้ ω_Z มีค่าที่เปลี่ยนไปเป็น

$$\omega_Z = \frac{1}{C_C(\frac{1}{G_{m2}} - R)} \tag{1.12}$$

หากทำให้ค่า R มีค่าสูงขึ้นมากพอ จะทำให้ ω_Z เปลี่ยนไปอยู่ในฝั่งของ left complex plane แทน และทำให้ผลตอบสนองของ phase response เปลี่ยนเป็นการมีค่ามากขึ้นที่ตำแหน่งศูนย์ดังกล่าว

ไม่ว่าอย่างไรก็ตาม การต่อ R แบบอนุกรมกับตัวเก็บประจุ C_C ทำให้เกิด pole ขึ้นมาใหม่หนึ่งตำแหน่งในช่วงความถี่สูง ซึ่ง pole ใหม่นี้จะมีค่าเป็น

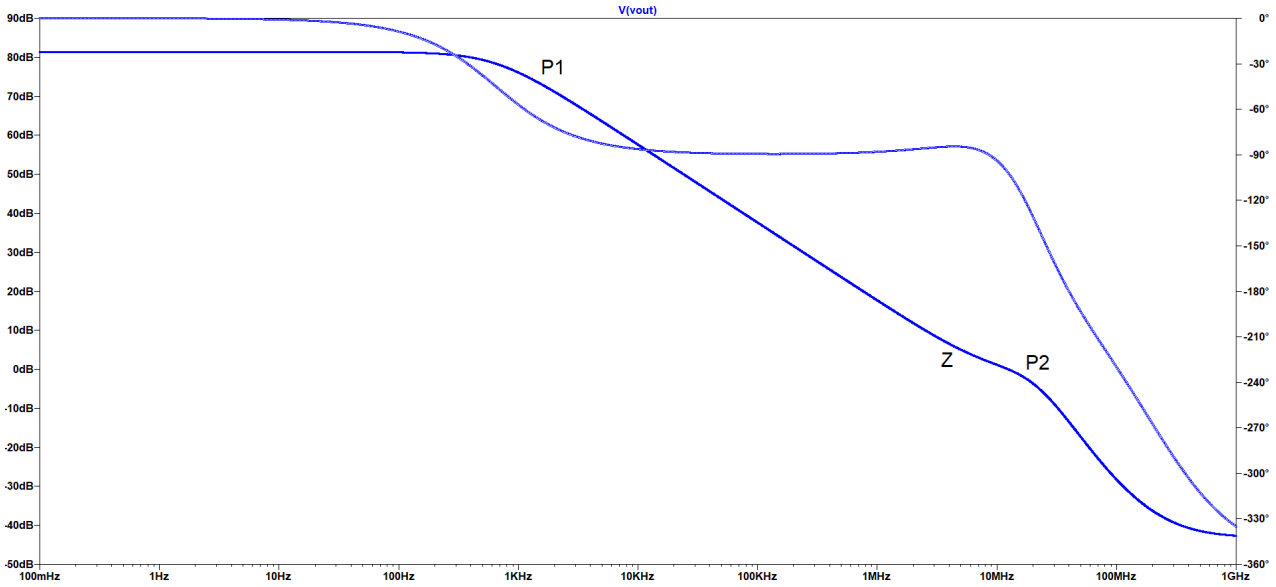
$$\omega_{P3} = \frac{1}{RC_1} \tag{1.13}$$

Slew Rate

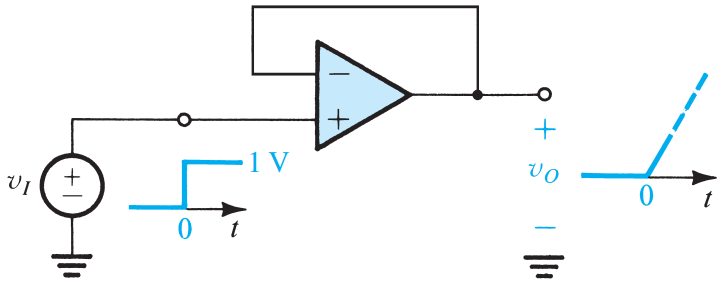
หากสัญญาณขาเข้าของวงจร op amp เปลี่ยนไปอย่างกะทันหัน สัญญาณขาออกของวงจร op amp จะไม่เปลี่ยนแปลงทันทีทันใด แต่จะค่อย ๆ เปลี่ยนจนกระทั่งถึงจุดคงตัว (steady-state) ความเร็วในช่วงระยะเวลาของการเปลี่ยนแปลงนั้นเรียกว่า Slew Rate

สำหรับการแปลงวงจร two-stage amplifier ให้อยู่ในรูปอย่างง่ายพอที่จะสามารถคำนวณ slew rate ได้โดยคร่าว ได้แสดงในรูปที่ 1.5 ซึ่งเป็นการประมาณให้ stage ที่สองอยู่ในรูปของ integrator

เมื่อสัญญาณขาเข้าด้านหนึ่งของวงจรเปลี่ยนแปลงไป อีกด้านของ differential amplifier จะถูกปิดลง จึงไม่มี



รูปที่ 1.3: ผลตอบสนองเชิงความถี่ของ two-stage opamp ในกรณีที่ศูนย์ของผลตอบสนองเลื่อนมาอยู่ที่ left complex plane แล้ว โดยสังเกตเห็นว่าศูนย์ (ω_Z) ดังกล่าวจะทำให้ phase response ของวงจรเพิ่มขึ้น ก่อนที่จะลดลงอย่างมากเมื่อผ่านความถี่ของขั้วตำแหน่งที่สอง (ω_{P2})



รูปที่ 1.4: แสดงนิยามโดยคร่าวของ Slew Rate (จาก Sedra and Smith, Microelectronic Circuits, 7E [5])

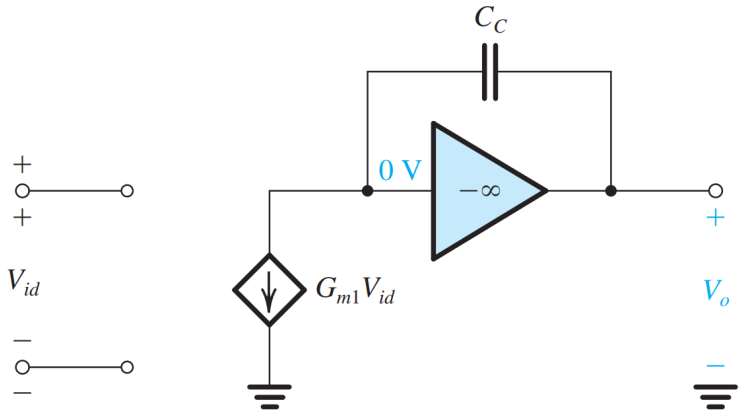
กระแสที่ไหลผ่านทรานซิสเตอร์ Q_2 หลังจากนั้น current mirror จะดึงกระแสจาก C_C เข้ามาที่ทรานซิสเตอร์ Q_2 แทน จากวงจร integrator อย่างง่ายในรูปที่ 1.5 จึงสามารถคำนวณได้ว่า

$$v_o(t) = (I/C_C)t \tag{1.14}$$

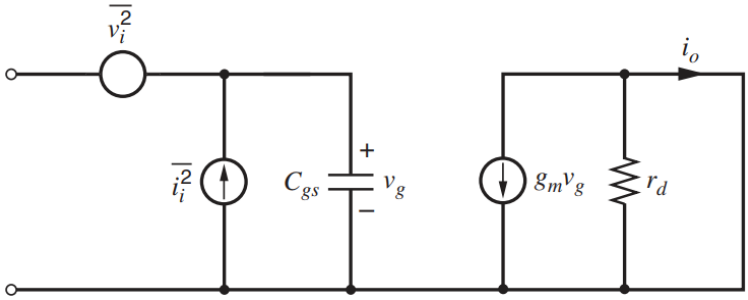
โดยที่ I คือกระแสจากทรานซิสเตอร์ Q_4 ดังนั้น slew rate หรือ SR จะมีค่าเป็น

$$SR = I/C_C \tag{1.15}$$

จากสมการดังกล่าวจึงได้ว่า ค่าของ C_C ไม่ควรมีค่ามาก เนื่องจากจะทำให้ slew rate ลดลง



รูปที่ 1.5: 2-stage op amp ที่ทำให้อยู่ในรูปอย่างง่าย



รูปที่ 1.6: โมเดลสัญญาณขนาดเล็กของ MOSFET ที่พิจารณาถึงสัญญาณรบกวนอ้างอิงขาเข้าแบบแรงดัน (voltage) และกระแส (current) ร่วมด้วย (จาก Gray et. al., Analysis and Design of Analog Integrated Circuits, 5E [2])

สัญญาณรบกวนอ้างอิงขาเข้า (Input-Refer Noise) ของ MOSFET

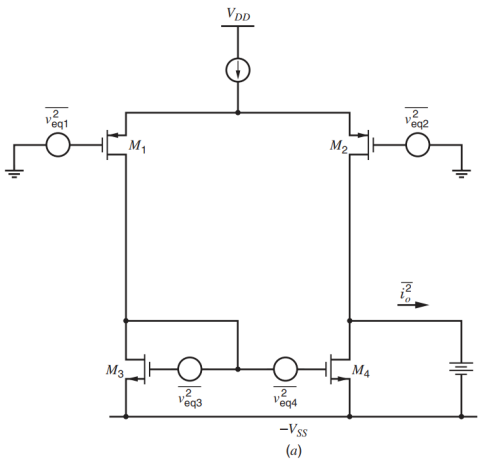
จากรูปที่ 1.6, Gray et. al. [2] ได้คำนวณแล้วว่าสัญญาณรบกวนอ้างอิงขาเข้าเชิงแรงดัน (voltage) จะมีค่าเป็นดังนี้

$$\frac{\overline{v_i^2}}{\Delta f} = \frac{8kT}{3g_m} + \frac{K_f}{WLC_{ox}f} \tag{1.16}$$

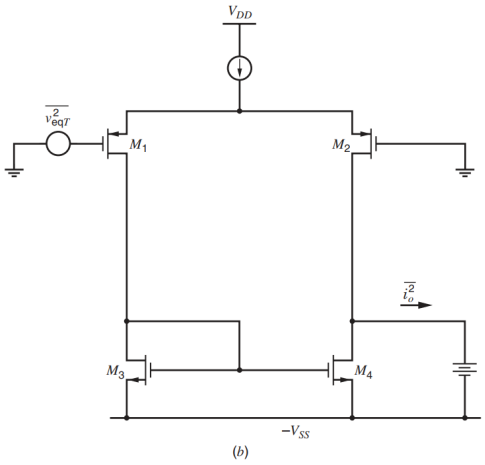
โดยที่ k คือค่าคงที่โบลทซ์มานน์ (Boltzmann constant), และ K_f ค่าคงที่ของ flicker noise ซึ่งโดยทั่วไปจะมีค่าประมาณ 3×10^{-12} V^2 pF

ในสมการที่ 1.16 พจน์แรกคือ thermal noise และพจน์ที่สองคือ flicker noise (หรือ 1/f noise) โดยสัญญาณรบกวนในสมการดังกล่าวถูกเขียนให้อยู่ในรูปของค่า root-mean square ต่อช่วงความถี่

เนื่องจากใน amplifier แบบ MOSFET ในอุดมคติจะมีค่าความต้านทานขาเข้าเป็นอนันต์ ดังนั้นจึงอาจไม่จำเป็นที่จะคำนึงถึงสัญญาณรบกวนอ้างอิงขาเข้าเชิงกระแส (current) ก็ได้



รูปที่ 1.7: สัญญาณรบกวนของทรานซิสเตอร์แต่ละตัวใน Differential Amplifier [2]



รูปที่ 1.8: วงจรสมมูลของสัญญาณรบกวนใน Differential Amplifier โดยรวมให้อยู่ในแหล่งจ่ายเดียว [2]

สัญญาณรบกวนอ้างอิงขาเข้า (Input-Refer Noise) ของ Differential Amplifier

จากรูปที่ 1.7 สัญญาณรบกวนจากทรานซิสเตอร์ตัวต่าง ๆ ใน Differential Amplifier สามารถนำมารวมกันให้อยู่ในแหล่งจากแรงดันแหล่งเดียวได้ดังในรูปที่ 1.8 โดยมีสมการดังนี้ คือ [2]

$$\overline{v_{eqT}^2} = \overline{v_{eq1}^2} + \overline{v_{eq2}^2} + (g_{m3}/g_{m1})^2(\overline{v_{eq3}^2} + \overline{v_{eq4}^2}) \tag{1.17}$$

จากสัญญาณรบกวนอ้างอิงขาเข้าของ MOSFET (สมการที่ 1.16) นำสัญญาณรบกวนแบบ flicker noise ไปแทนในสมการที่ 1.17 เพื่อให้ได้สัญญาณรบกวน flicker noise สมมูลแบบแหล่งจ่ายแรงดันแหล่งเดียว คือ

$$\overline{v_{1/f}^2} = \frac{2K_p}{fW_1L_1C_{ox}}\left(1 + \frac{K_n\mu_nL_1^2}{K_p\mu_pL_3^2}\right)\Delta f \tag{1.18}$$

โดยที่ K_p และ K_n คือ flicker noise constant ของ PMOS และ NMOS ตามลำดับ

เช่นเดียวกันคือ จากสัญญาณรบกวนอ้างอิงขาเข้าของ MOSFET (สมการที่ 1.16) นำสัญญาณรบกวนแบบ thermal

noise ไปแทนในสมการที่ 1.17 เพื่อให้ได้สัญญาณรบกวน thermal noise สมมูลแบบแหล่งจ่ายแรงดันแหล่งเดียว คือ

$$\bar{v_{1/f}^2} = 4kT\frac{4}{3\sqrt{2\mu_p C_{ox}(W/L)_1 I_D}}(1+\sqrt{\frac{\mu_n(W/L)_3}{\mu_p(W/L)_1}})\Delta f \tag{1.19}$$

จากสมการที่ 1.18 การเพิ่ม W และ L ของทรานซิสเตอร์ Q_1 และ Q_2 จะทำให้สัญญาณรบกวนลดลงได้ แต่จะทำให้ความจุไฟฟ้าแบบต่าง ๆ ใน MOS เพิ่มขึ้น และทำให้ผลตอบสนองเชิงความถี่ของวงจร op amp นี้มี crossover frequency ที่ต่ำลง

ในขณะที่จากสมการที่ 1.19 การเพิ่มอัตราส่วน W/L ของทรานซิสเตอร์ Q_1 และ Q_2 สามารถลดขนาดของสัญญาณรบกวนในวงจรได้เช่นกัน นอกจากนั้นการเพิ่มกระแสไบแอสในทรานซิสเตอร์ก็สามารถทำให้ลดสัญญาณรบกวนได้ แต่จะทำให้ช่วงสัญญาณขาเข้า (input common-mode range) และช่วงสัญญาณขาออก (output swing) มีค่าต่ำลงเนื่องจาก overdrive voltage ของทรานซิสเตอร์เพิ่มขึ้น นอกจากนั้นยังทำให้วงจรใช้พลังงานมากขึ้นด้วย

การต่อ body ของทรานซิสเตอร์ต่าง ๆ เข้าไปที่ขาของ power supply โดยตรงจะทำให้สัญญาณรบกวนของวงจร op amp ต่ำลงอย่างมีนัยสำคัญ

Total Harmonic Distortion

การคำนวณ Total Harmonic Distortion หรือ THD สามารถทำได้โดยการนำขนาดของ amplitude ของสัญญาณที่มี Harmonic สูงขึ้นไปเป็น 2,3,4,... เท่า เปรียบเทียบกับสัญญาณที่มี Harmonic พื้นฐานที่ต้องการจากวงจร ซึ่งใช้สมการดังนี้ในการคำนวณ

$$THD = \frac{\sqrt{V_2^2+V_3^2+V_4^2+...}}{V_1} \tag{1.20}$$

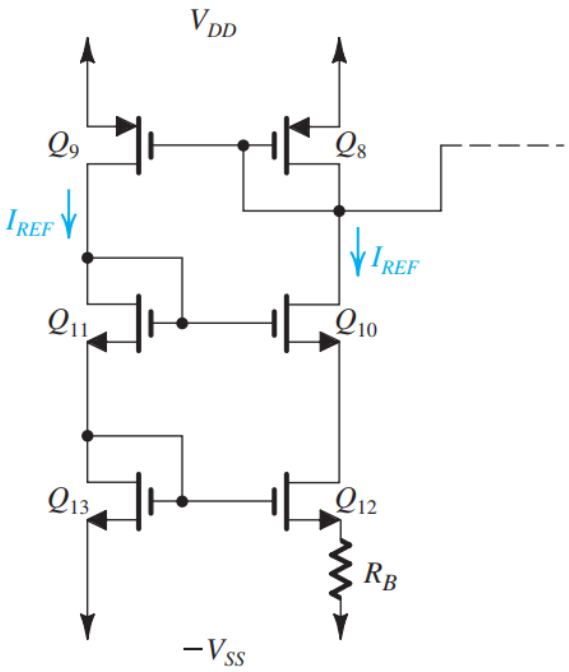
ในโปรแกรม LTSpice นั้น การคำนวณ Total Harmonic Distortion สามารถทำได้โดยใช้คำสั่ง `.four` โดยสำหรับค่าเริ่มต้นในโปรแกรม จำนวน Harmonic ทั้งหมดที่โปรแกรมจะรวมให้ในการคำนวณ THD คือ 10 Harmonics

แหล่งจ่ายกระแสอ้างอิง

สำหรับแหล่งจ่ายกระแสอ้างอิง สามารถออกแบบเพื่อให้กระแสไม่เปลี่ยนแปลงไปเมื่อแรงดันจาก power supply เปลี่ยนแปลง การออกแบบดังกล่าวสามารถมีรูปแบบได้ดังรูปที่ 1.9

จากภาพ จะเห็นได้ว่า Q11 และ Q10 จะเป็น current mirror ให้กันและกัน นอกจากนั้น Q9 และ Q8 ต่างก็เป็น current mirror ซึ่งกันและกันเช่นกัน นั่นแสดงว่ากระแสไฟฟ้าในรางซ้ายจะถูกอ้างอิงกับกระแสไฟฟ้าในรางขวาเสมอ โดยขนาดของกระแสนั้นสามารถถูกกำหนดได้โดย Q13 และ Q12 ที่มีอัตราส่วนของ W/L ไม่เท่ากัน และมีตัวต้านทาน R_B เพื่อเพิ่มข้อจำกัดของกระแสอ้างอิงนั้น

จากการคำนวณของ Sedra and Smith [5] จะได้ว่าขนาดของ R_B ในวงจรรูปที่ 1.9 สามารถออกแบบให้ได้ I_{REF}



รูปที่ 1.9: แหล่งจากอ้างอิงสำหรับ CMOS op amp โดยที่ Q_8 เป็นทรานซิสเตอร์ตัวเดียวกันกับ Q_8 ในวงจรของรูปที่ 1.1 (จาก Sedra and Smith, Microelectronic Circuits, 7E [5])

ตามต้องการ โดยใช้สมการ

$$R_B = \frac{2}{\sqrt{2\mu_n C_{ox}(W/L)_{12} I_{REF}}}\left(\sqrt{\frac{(W/L)_{12}}{(W/L)_{13}}} - 1\right) \tag{1.21}$$

ซึ่งจะเห็นได้ว่าขนาดของ I_{REF} เป็นอิสระจากแรงดันของ power supply และขนาดของ threshold voltage ของทรานซิสเตอร์ต่าง ๆ ในวงจร

1.1.2 เงื่อนไขของการออกแบบวงจร

สำหรับเงื่อนไข (specifications) ของวงจร op amp ซึ่งเป็นโจทย์ที่ทางบริษัทให้มา มีรายละเอียดดังนี้

1. ออกแบบวงจร two-stage op amp โดยใช้เทคโนโลยี CMOS 0.18 nm ซึ่งมีโมเดลของ BSIM3 ให้มา
2. Supply Voltage ของ op amp นี้จะมีค่าอยู่ระหว่าง 1.1 V ถึง 1.8 V (single supply)
3. วงจร op amp ดังกล่าวต้องขับโหลดซึ่งเป็นตัวเก็บประจุขนาด C_L = 100 fF
4. การใช้พลังงานของวงจรจะต้องไม่เกิน 6 µW
5. DC open-loop gain $\geq$ 60 dB
6. ช่วงของสัญญาณขาออก (Output swing) $\geq V_{DD}$ − 0.2 V
   (โดยที่ DC open-loop gain $\geq$ 60 dB)
7. ช่วงของสัญญาณขาเข้า (Input common-mode voltage range) $\geq$ 50% ของช่วง $(V_{DD} - V_{SS})$
   (โดยที่ DC open-loop gain $\geq$ 60 dB)

8. สัญญาณรบกวนอ้างอิงขาเข้ารวมของวงจร ตั้งแต่ความถี่ 1 Hz จนถึง 1 GHz เมื่อต่อวงจร op amp เป็นรูปแบบ unity-gain feedback ต้องมีค่าไม่เกิน 220 μVrms
9. 1% settling time ของผลตอบสนองแบบ step input ขนาด 0.5 V ต้องมีค่าไม่เกิน 2 μs เมื่อต่อวงจร op amp เป็นรูปแบบ unity-gain feedback
10. total harmonic distortion ของสัญญาณขาออกเมื่อมีขนาดแบบขึ้นสุด ลงสุด จะต้องมีค่าไม่เกิน 0.2%
11. เนื่องจากการผลิตวงจร IC อาจมีความเบี่ยงเบนของค่า parameters ต่าง ๆ ของทรานซิสเตอร์จำนวนมากในวงจร ดังนั้นการจำลองวงจรควรที่จะคำนึงถึงค่า parameters ของทรานซิสเตอร์ต่าง ๆ ในกรณีที่มีโอกาสเบี่ยงเบนไปมากที่สุด ซึ่งเรียกโมเดลของทรานซิสเตอร์ที่มีค่าเบี่ยงเบนของ parameters ต่าง ๆ อย่างสุดขอบว่าเป็น process corner โดยสำหรับในการออกแบบวงจร op amp นี้ ให้ทดสอบในโมเดลแบบ TT, FF และ SS (ตัวอักษร T, F, S ย่อมาจาก typical, fast และ slow ตามลำดับ โดยตัวอักษรตัวแรกแทน process corner ของ NMOS และตัวอักษรตัวที่สองแทน process corner ของ NMOS)
12. ในการจำลองวงจรในหัวข้อต่าง ๆ ให้จำลองในช่วงอุณหภูมิที่แตกต่างกันกันด้วย คือช่วงอุณหภูมิตั้งแต่ 0°C. ถึง 70°C.

1.1.3 การออกแบบวงจรตามเงื่อนไข

การจำลองเพื่อหาคุณลักษณะของโมเดล CMOS

ก่อนที่จะเริ่มออกแบบวงจร op amp ตามเงื่อนไขของโจทย์ ควรที่จะหา characteristics โดยคร่าวของโมเดล CMOS ก่อน ซึ่งได้แก่ threshold voltage, k'_n หรือ k'_p และ Early voltage $|V'_A|$

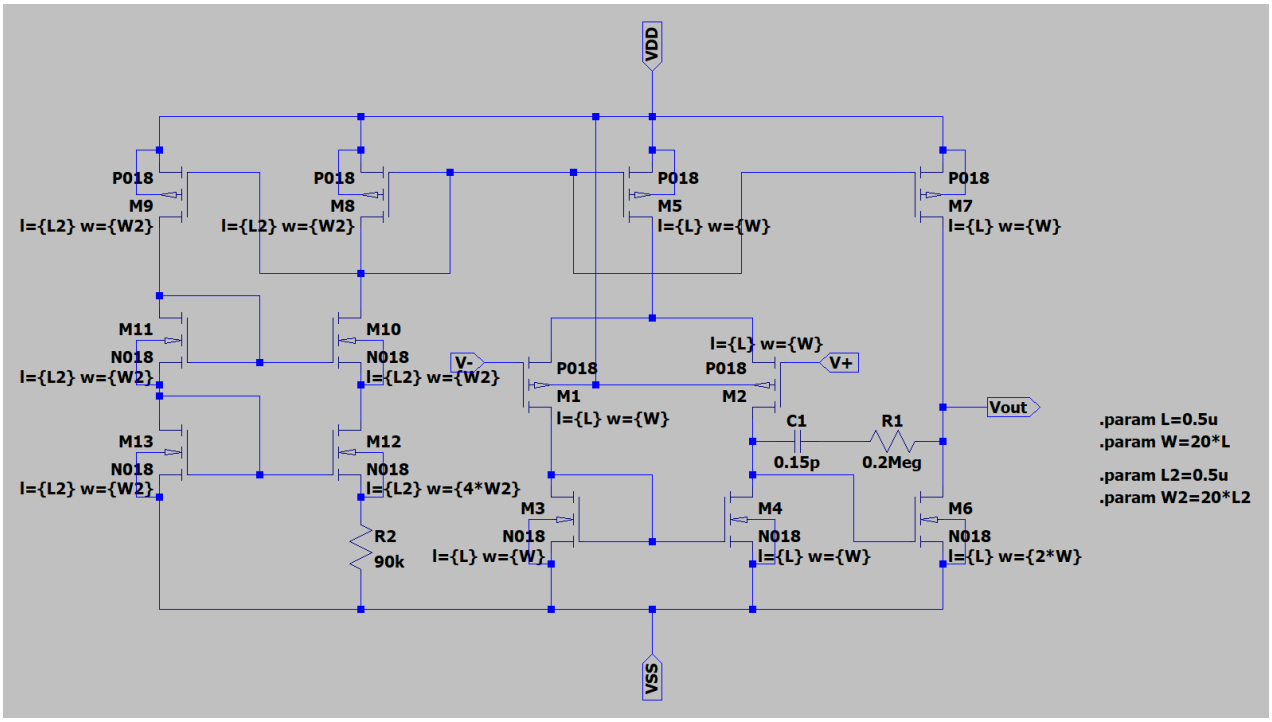
สำหรับโมเดล NMOS:

- Threshold Voltage $|V_{tn}|$: 0.391 V
- k'_n: 392 μA/V^2
- $|V'_A|$: 20 V/μm

สำหรับโมเดล PMOS:

- Threshold Voltage $|V_{tp}|$: 0.405 V
- k'_p: 89.0 μA/V^2
- $|V'_A|$: 28 V/μm

สำหรับ V_{tn} และ $|V_{tp}|$ ถูกคำนวณในช่วง $|V_{gs}|$ ระหว่าง 0.39 V ถึง 0.70 V และ $|V'_A|$ ถูกคำนวณในช่วง $|V_{ds}|$ ระหว่าง 0.60 V ถึง 0.70 V โดยที่ $|V'_A|$ เป็นเพียงค่าประมาณเฉลี่ยเนื่องจากค่า $|V'_A|$ มีความไม่เป็นอิสระจาก I_D ที่เปลี่ยนแปลงไปอีกด้วย



รูปที่ 1.10: วงจร op amp ของผู้จัดทำ

โครงสร้างโดยคร่าวของ Op Amp

การจัดการพลังงาน

เนื่องจากวงจร op amp ที่ออกแบบไม่สามารถใช้พลังงานได้เกิน 6 μW และเนื่องจากแรงดันจาก power supply สามารถมีค่าได้มากที่สุดคือ 1.8 V จากสมการ $P = VI$ ได้ว่ากระแสที่จ่ายมาจาก power supply จะต้องมีค่าไม่เกิน 3.3 μA

จากรูปที่ 1.10 เนื่องจากมีรางของกระแสทั้งหมด 4 ราง ดังนั้น I_{REF} จึงควรมีค่าไม่เกิน $3.3/4 = 0.83$ μA ดังนั้นเพื่อความเผื่อเกิน จึงจะออกแบบวงจรเพื่อให้ I_{REF} มีค่าเป็น 0.5 μA

การออบแบบวงจรไบแอสกระแสอ้างอิง

จากสมการที่ 1.21 เมื่อให้ $\frac{(W/L)_{12}}{(W/L)_{13}} = 4$ โดยที่ $(W/L)_{13}$ เท่ากับ 20, $I_{REF} = 0.5 \times 10^{-6}$, $k'_n = 392 \times 10^{-6}$ ค่าของ R_B ที่คำนวณได้จะมีค่าเป็น 11.3 kΩ

ไม่ว่าอย่างไรก็ตาม ค่าของกระแส I_{REF} กลับมีค่ามากกว่า 0.5 μA มาก เกินอะไรขึ้น?

การจำลองเพื่อหาคุณลักษณะของโมเดล CMOS ตอนที่สอง

ความผิดพลาดของการหาคุณลักษณะของโมเดล CMOS ในตอนแรกคือการหาคุณลักษณะในช่วงการทำงานที่ไม่ใช่ช่วงการทำงานที่ใช้จริง ดังนั้นจึงควรหาคุณลักษณะใหม่อีกครั้ง โดยเปลี่ยนช่วงการจำลองของ $|V_{gs}|$ เป็นช่วง 0.20 V ถึง 0.36 V สำหรับ NMOS และ 0.25 V ถึง 0.45 V สำหรับ PMOS (ซึ่งเป็นช่วงที่ค่ากระแส $I_D = 0.5 \times 10^{-6}$) ผลลัพธ์ของคุณลักษณะของโมเดล CMOS จะเปลี่ยนเป็น

สำหรับโมเดล NMOS:

- Threshold Voltage: 0.235 V
- k'_n: 19.7 μA/V^2

สำหรับโมเดล PMOS:

- Threshold Voltage: 0.299 V
- k'_p: 15.2 μA/V^2

เนื่องจากช่วงการทำงานที่ใช้จริง กระแส I_D มีค่าต่ำมาก ๆ จนได้ว่า CMOS จะทำงานอยู่ในช่วงของ subthreshold

การออบแบบวงจรไบแอสกระแสอ้างอิง ตอนที่สอง

จากคุณลักษณะของโมเดล CMOS ที่หาได้ใหม่จากหัวข้อที่แล้ว จะได้ว่า R_B ที่คำนวณได้ใหม่จะมีค่าเป็น 50.4 kΩ ไม่ว่าอย่างไรก็ตาม ค่าของ I_{REF} ที่จำลองได้จริงก็ยังคงมีค่ามากกว่าที่ต้องการอยู่ การปรับละเอียด (fine-tuning) จึงยังคงจำเป็น และได้ว่าค่า R_B ที่ต้องการในตอนสุดท้าย คือ 90 kΩ ซึ่งทำให้ได้กระแสอ้างอิง I_{REF} ที่มีค่าเป็น 0.657 μA[1]

ค่าของ R_B ไม่ควรสูงเกินไปเนื่องจากจะทำให้พื้นที่ที่ใช้ในวงจรรวมเพิ่มขึ้น

การออกแบบวงจรขยายทั้งสอง stage

เมื่อเลือกอัตราส่วนของ W/L ของ MOS ทุกตัวเป็น 20 (ยกเว้น Q_6 ที่จะต้องมีค่า W/L เท่ากับ 40 เนื่องด้วยสมการที่ 1.1.) และให้ L ของ MOS ทุกตัวเป็น 0.5 μm

เมื่อใช้สมการที่ 1.5 และสมการที่ 1.6 แทนค่าคุณลักษณะต่าง ๆ ของ CMOS ที่เกี่ยวข้องทั้งหมดเข้าไปในสมการจะได้ค่าของ $A_1 = A_2 = 571$ ดังนั้น DC open-loop gain ของวงจรที่ออกแบบได้ในเชิงทฤษฎีจะมีค่าเป็น $A_1A_2 = 8.16 \times 10^4$ = 98.2 dB

สำหรับในการจำลองวงจร ได้ว่าค่าของ DC open-loop gain จะมีค่าเพียงประมาณ 80 dB แต่ก็ยังเกินจากเงื่อนไขที่กำหนดไว้ในโจทย์

ค่าของ DC open-loop gain สามารถทำให้สูงขึ้นได้หากเพิ่มความยาว L ของ MOS แต่ crossover frequency ของผลตอบสนองเชิงความถี่จะลดลงเนื่องจาก ω_{P2} ที่ลดลง. นอกจากนั้นการลด I_D จะสามารถเพิ่ม DC open-loop gain ได้ แต่ขนาดของสัญญาณรบกวนอ้างอิงขาเข้าก็จะเพิ่มขึ้นตามไป

การออกแบบผลตอบสนองเชิงความถี่ของอัตราขยายแบบวงเปิด (open-loop gain) ของวงจร

จากตำแหน่งของ dominant pole ω_{P1} ในสมการที่ 1.11 และขนาดของ DC open-loop gain ในสมการที่ 1.5 และสมการที่ 1.6 เราสามารถคำนวณหาตำแหน่งของ crossover frequency ได้เป็น

$$\omega_t = G_{m1}/C_C$$

[1]โดยใช้โมเดลแบบ TT CMOS และจำลองที่อุณหภูมิ 35°C

โดยค่าของ G_{m1} สามารถคำนวณได้จาก $g_m = \sqrt{2k'_n(W/L)I_D}$ และถ้าหากเลือกค่า C_C = 0.15 pF, จะได้ว่าค่าของ crossover frequency หรือ ω_t จะมีค่าเป็น 17.1 MHz.

จากการจำลองวงจร ได้ว่า ω_t อยู่ในช่วงระหว่าง 9 ถึง 13 MHz ซึ่งยังเกินจากเงื่อนไขที่กำหนดไว้ในโจทย์

การออกแบบ crossover frequency ω_t ไม่ควรจะมีค่ามากกว่านี้อีก เนื่องจากขนาดของ phase margin จะมีค่าลดลงมากเนื่องจากตำแหน่งของขั้วที่สอง: ω_{P2}.

ปัญหาที่แก้ได้ยากของการออกแบบผลตอบสนองเชิงความถี่ของอัตราขยายแบบวงเปิด เกิดจากตำแหน่งของขั้วที่สอง, ω_{P2}, ซึ่งมีค่าเป็น $\omega_{P2} = G_{m2}/C_2$ เนื่องจาก C_2 เป็นค่าที่กำหนดมาจาก C_L และความจุไฟฟ้าตามคุณลักษณะของ MOS ตัวต่าง ๆ และค่าของ G_{m2} ก็สามารถเปลี่ยนได้ยากเนื่องจากการเปลี่ยนค่าดังกล่าวจะทำให้ค่าคุณลักษณะของวงจร op amp ที่ออกแบบไว้แล้วเปลี่ยนแปลงไปด้วย ดังนั้น ขั้วที่สองดังกล่างจึงดูเหมือนว่าถูกกำหนดไว้แล้วและเปลี่ยนแปลงได้ยาก ซึ่งหากเราต้องการเพิ่ม phase margin ของผลตอบสนองดังกล่าว เราจะต้องออกแบบ crossover frequency หรือ ω_t ให้ต่ำพอ โดยอาจต่ำกว่าค่าของ ω_{P2} ประมาณ 1 decade ลงไป ไม่ว่าอย่างไรก็ตามการทำเช่นนั้นจะทำให้ค่าของ crossover frequency ต่ำเกินไป

ค่าของ ω_{P2} สามารถคำนวณได้ยากเนื่องจากขึ้นอยู่กับ C_2 ที่ประกอบด้วยความจุไฟฟ้าตามคุณลักษณะของ MOS ตัวต่าง ๆ ดังนั้นการนำค่าจากผลการจำลองมาใช้เลยน่าจะเป็นทางเลือกที่ดีกว่า ซึ่งการประมาณค่าของ ω_{P2} (โดยใช้การลากเส้นโดยคร่าวบนรูปที่ 1.3) สามารถหาได้ว่าอยู่ที่ประมาณ 20 MHz.

การแก้ปัญหาที่เกิดจาก ω_{P2} ได้กล่าวไว้แล้วในสมการที่ 1.12 โดยการออกแบบวงจรจะต้องต่อ R อนุกรมกับ C_C เพื่อเปลี่ยนตำแหน่งของ zero ให้อยู่ที่ left complex plane แทน โดยหากเลือกค่าของ ω_Z อย่างระมัดระวังให้มีค่าอยู่ต่ำกว่าตำแหน่งของ ω_{P2} เล็กน้อย จทำให้ขั้ว ω_{P2} ดังกล่าวถูกยกเลิก (cancelled) ไป

โดยจากสมการ 1.12 เมื่อเลือกค่า R = 200 kΩ จะทำให้ได้ค่าของ ω_Z อยู่ที่ประมาณ 6.2 MHz ซึ่งตรงกับผลการจำลองของวงจรในโปรแกรม

Phase margin ของ open-loop gain จากการคำนวณควรจะมีค่าน้อยกว่า 90 องศาเล็กน้อย เนื่องจากขั้วแรกและขั้วที่สองที่อยู่ใกล้กับ crossover frequency

Slew Rate

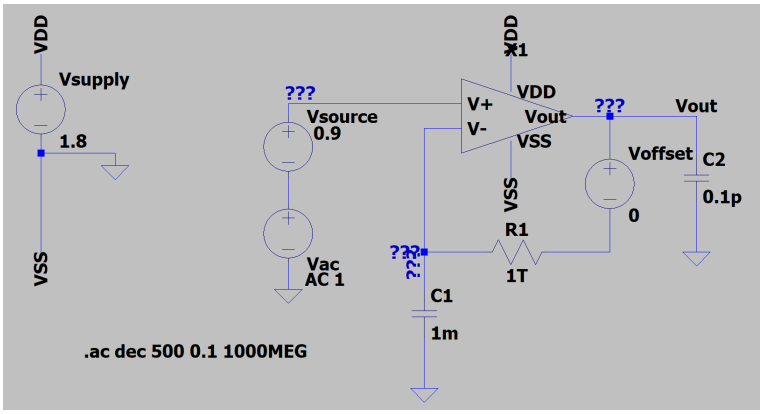
จากเงื่อนไขของการออกแบบในโจทย์ ค่าของสัญญาณขาออกจะต้องตาม 0.5-V step input ในเวลาไม่เกิน 2 μs ดังนั้น slew rate ของสัญญาณขาออกจะต้องมีค่าไม่ต่ำกว่า $0.5/2 \times 10^{-6} = 2.5 \times 10^5$ V/s.

จากสมการที่ 1.15 การคำนวณ slew rate ได้ค่าออกมาเป็น 2.19 × 10^6 V/s ซึ่งมีค่ามากกว่าที่โจทย์ได้กำหนดเอาไว้มาก

การออกแบบเพื่อลดสัญญาณรบกวน

เนื่องจากการคำนวณสัญญาณรบกวนของวงจรค่อนข้างมีความซับซ้อนมาก การจำลองวงจรน่าจะทำให้การออกแบบเป็นไปได้ง่ายกว่า

การลดสัญญาณรบกวนของวงจรมีวิธีทั่วไปสองแบบ คือการเพิ่มอัตราส่วน W/L ของทรานซิสเตอร์ที่เกี่ยวข้องกับการขยายสัญญาณ และการต่อ body ของทรานซิสเตอร์เข้าไปที่แหล่ง power supply โดยตรง



รูปที่ 1.11: วงจรเพื่อจำลองผลตอบสนองเชิงความถี่ของอัตราขยายแบบวงเปิด

ช่วงของสัญญาณขาเข้า (Input Common-Mode Range) และสัญญาณขาออก (Output Swing)

ขนาดของ overdrive voltage ของแต่ละทรานซิสเตอร์สามารถคำนวณได้จาก

$$\frac{1}{2}k'_n(W/L)V_{ov}^2 = I_D$$

จากสมการดังกล่าวจะได้ว่า $|V_{ov6}| = |V_{ov3}| = 0.041\text{V}$, $|V_{ov5}| = |V_{ov7}| = 0.066\text{V}$, and $|V_{ov1}| = 0.046\text{V}$. โดยเมื่อพิจารณาร่วมกับข้อมูล $|V_{tn}|$ และ $|V_{tp}|$ และจากสมการที่ 1.2 จะได้ว่า

- ช่วงของสัญญาณขาเข้า (input common-mode range) คือตั้งแต่ -0.023 V ถึง v_{DD} - 0.411 V
- ช่วงของสัญญาณขาออก (output swing) คือตั้งแต่ 0.041 V ถึง v_{DD} - 0.066 V

ยิ่งช่วงของสัญญาณขาออก (output swing) มีความกว้างมากขึ้นเท่าไหร่ ขนาดของ total-harmonic-distortion (THD) ของสัญญาณขาออกก็จะยิ่งลดลงเท่านั้น

1.1.4 การจำลองวงจรและผลการทดลอง

ผลตอบสนองเชิงความถี่ของอัตราขยายแบบวงเปิด: DC Gain, Crossover Frequency และ Phase Margin

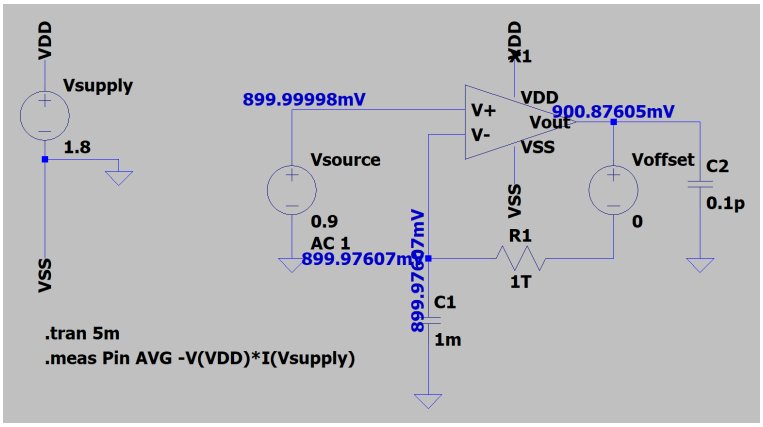
สำหรับการจำลองของตอนนี้จะมีทั้งหมด 171 การทดลอง ดังนี้

- ทดสอบบน 3 process corners (FF, TT, SS).
- แต่ละ process corner ใช้แรงดันของแหล่งจ่ายเป็น 1.1 V และ 1.8 V
- กรณีที่เป็น 1.1 V:
  - เปลี่ยนค่าของแรงดันขาเข้า: 0.1, 0.3, 0.5, 0.7, 0.9 V
  - เปลี่ยนค่าของแรงดันขาออก: 0.1, 0.55, 1.0 V
- กรณีที่เป็น 1.8 V:
  - เปลี่ยนค่าของแรงดันขาเข้า: 0.1, 0.4, 0.7, 1.0, 1.3, 1.6 V
  - เปลี่ยนค่าของแรงดันขาออก: 0.1, 0.5, 0.9, 1.3, 1.7 V
- แต่ละการทดลอง ทำที่อุณหภูมิ 0, 35, 70 °C.

ผลการทดลองแบบสรุป

- ค่าของ DC gain $\geq$ 60 dB สำหรับทั้ง 171 การทดลอง
- ค่าของ Phase Margin $\geq$ 60 degrees สำหรับทั้ง 171 การทดลอง
- ค่าของ Crossover frequency $\geq$ 7 MHz สำหรับทั้ง 171 การทดลอง **ยกเว้น**ในการทดลองต่อไปนี้
  - V_{in} = 1.6 V เมื่อ V_{supply} = 1.8 V.
  - V_{in} = 0.7 V และ 0.9 V เมื่อ V_{supply} = 1.1 V.

การใช้พลังงาน



รูปที่ 1.12: วงจรเพื่อทดสอบการใช้พลังงานของ op amp

การวัดการใช้พลังงานของวงจร op amp อย่างง่ายที่สุดคือการวัดค่าแรงดันและกระแสของแหล่งจ่ายไฟ แล้วนำมาคำนวณต่อเป็นกำลังไฟฟ้าอีกทีหนึ่ง

สำหรับการจำลองของตอนนี้จะมีทั้งหมด 18 การทดลอง ดังนี้

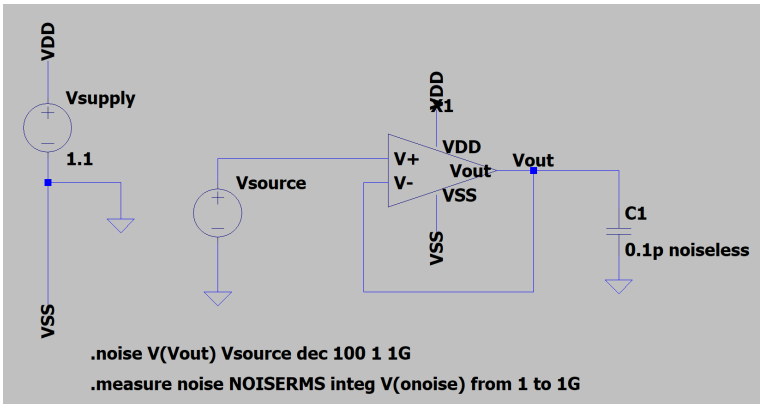
- ทดสอบบน 3 process corners (FF, TT, SS).
- แต่ละ process corner ใช้แรงดันของแหล่งจ่ายเป็น 1.1 V และ 1.8 V
- แต่ละการทดลอง ทำที่อุณหภูมิ 0, 35, 70 °C.

ผลการทดลองแบบสรุป

- สำหรับ power supply ค่าแรงดัน 1.8 V,
  ได้ว่า การใช้กำลังไฟฟ้าอยู่ในช่วง 4.11 ถึง 5.06 µW
- สำหรับ power supply ค่าแรงดัน 1.1 V,
  ได้ว่า การใช้กำลังไฟฟ้าอยู่ในช่วง 1.74 ถึง 2.22 µW

การทดสอบสัญญาณรบกวน

สำหรับการจำลองของตอนนี้จะมีทั้งหมด 18 การทดลอง ดังนี้

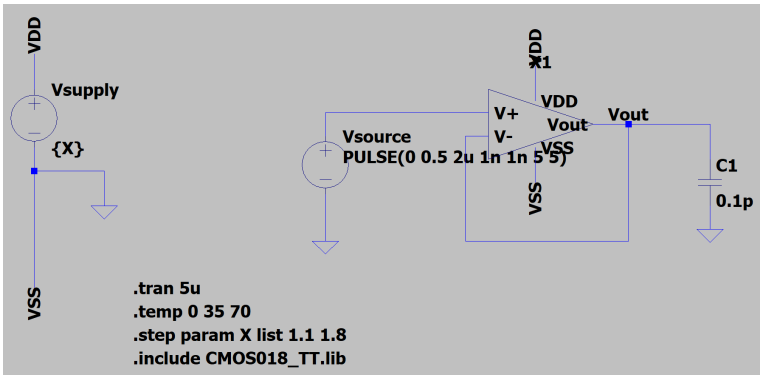


รูปที่ 1.13: วงจรที่ใช้สำหรับวัดค่าสัญญาณรบกวนอ้างอิงขาเข้า ซึ่งวัดจาก V_{out}.

- ทดสอบบน 3 process corners (FF, TT, SS).
- แต่ละ process corner ใช้แรงดันของแหล่งจ่ายเป็น 1.1 V และ 1.8 V
- แต่ละการทดลอง ทำที่อุณหภูมิ 0, 35, 70 °C.

ผลการทดลองแบบสรุป ขนาดของสัญญาณรบกวนอ้างอิงขาเข้ารวมแบบ root-mean square รวมตั้งแต่ความที่ 1 Hz ถึง 1 GHz ในทุกการทดลอง มีค่าอยู่ในช่วง 120 to 180 µVrms.

การทดสอบ Settling Time



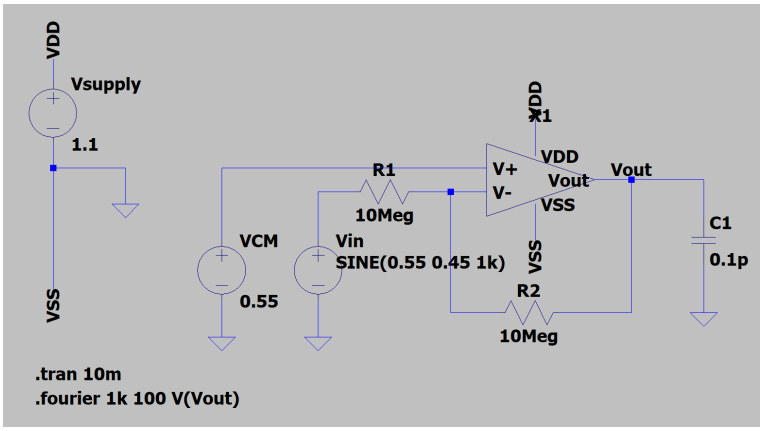
รูปที่ 1.14: วงจรที่ใช้สำหรับวัด Settling Time

สำหรับการจำลองของตอนนี้จะมีทั้งหมด 18 การทดลอง ดังนี้

- ทดสอบบน 3 process corners (FF, TT, SS).
- แต่ละ process corner ใช้แรงดันของแหล่งจ่ายเป็น 1.1 V และ 1.8 V
- แต่ละการทดลอง ทำที่อุณหภูมิ 0, 35, 70 °C.

ผลการทดลองแบบสรุป ช่วงของ 1% settling time ในทุกการทดลองอยู่ระหว่าง 0.35 ถึง 0.57 µs

การทดสอบ Total Harmonic Distortion



รูปที่ 1.15: วงจรที่ใช้สำหรับทดสอบ Settling Time

THD วัดโดยใช้ output swing แบบ sine wave ที่ขึ้นสุดลงสุดในช่วง V_{DD} - 0.2 V.

สำหรับการจำลองของตอนนี้จะมีทั้งหมด 18 การทดลอง ดังนี้

- ทดสอบบน 3 process corners (FF, TT, SS).
- แต่ละ process corner ใช้แรงดันของแหล่งจ่ายเป็น 1.1 V และ 1.8 V
- แต่ละการทดลอง ทำที่อุณหภูมิ 0, 35, 70 °C.

ผลการทดลองแบบสรุป ช่วงของ 1% settling time ในทุกการทดลองอยู่ระหว่าง 0.35 ถึง 0.57 µs

การเขียนโค้ดเพื่อให้สามารถรันการทดลองบน LTSpice แบบอัตโนมัติ

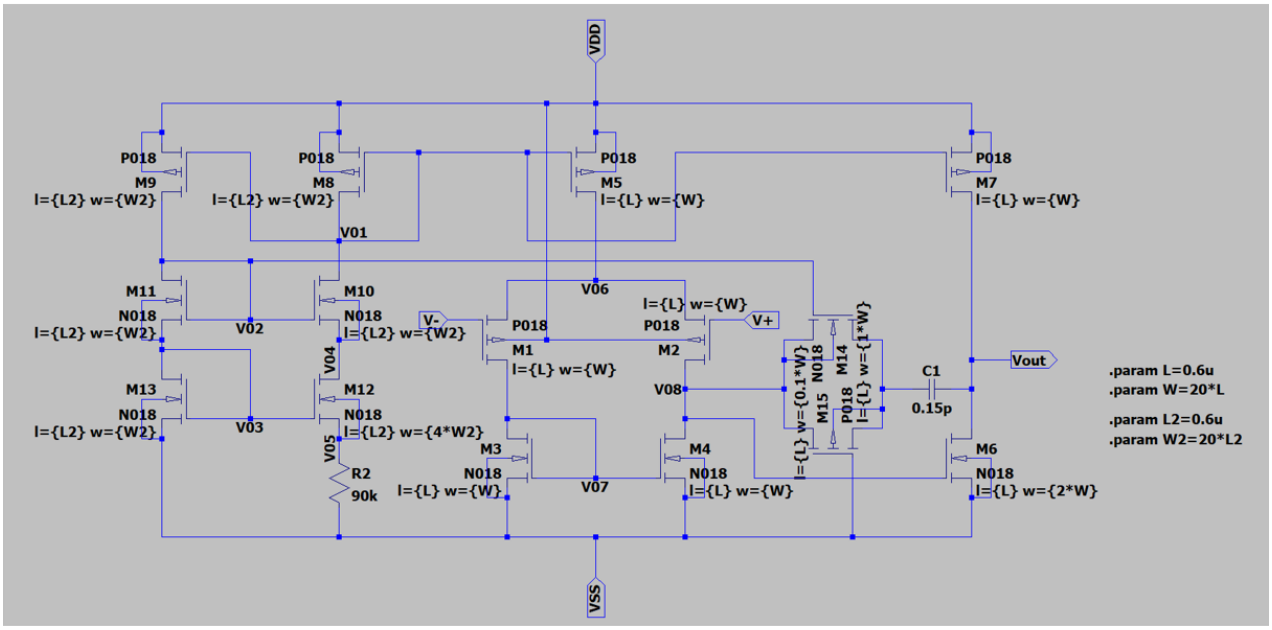
- ใช้ LTSpice ในการจำลองพฤติกรรมของวงจรไฟฟ้า
- การรันอัตโนมัติสั่งผ่านโค้ดของ Python
- ใช้ PyLTSpice Python library จาก Nuno Brum ซึ่งสามารถรันไฟล์ของ LTSpice ได้โดยตรง, สามารถเปลี่ยนค่าขององค์ประกอบในวงจร, อ่านไฟล์ `.raw` ซึ่งมีผลของการจำลองวงจรได้จากโค้ดของ Python โดยตรง [1]

1.1.5 การพัฒนาวงจรที่ออกแบบ

เนื่องจากในวงจร op amp ที่ออกแบบในรูปที่ 1.10 ยังมีตัวต้านทานขนาดใหญ่ถึงสองตัว ซึ่งอาจจะทำให้วงจรกินพื้นที่ของ IC ค่อนข้างมาก จึงสามารถพัฒนาต่อได้อีก ดังนี้

การแทนที่ตัวต้านทานที่ต่ออนุกรมกับ C_C ด้วย transmission gate

เนื่องจาก MOSFET สามารถทำหน้าที่เป็นตัวต้านทานได้เช่นกัน โดยการไบแอสแรงดันที่ขา gate ของทรานซิสเตอร์เพื่อให้ได้ช่วงในการทำงานอยู่ในช่วง triode region โดยเราสามารถไบแอสแรงดันได้โดยใช้วงจรสร้างกระแสที่ได้สร้าง



รูปที่ 1.16: การแทนที่ตัวต้านทานที่ต่ออนุกรมกับ C_C ด้วย transmission gate

ไว้แล้ว และเนื่องจากวงจรสร้างกระแสดังกล่าวสามารถให้แรงดันที่ค่อนข้างคงที่ ดังนั้นจึงได้ว่าแรงดันดังกล่าวมีความเหมาะสมที่จะใช้ในการไบแอสแรงดันที่ขา gate นั้น

สำหรับในรูปที่ 1.16 ตัว MOSFET ที่ทำหน้าที่ในการเป็นตัวต้านทานคือ Q14 และ Q15 โดยเหตุผลที่ใช้ทั้ง PMOS และ NMOS พร้อมกันเนื่องจากเพื่อทำให้การทำงานของ resistor นั้นกว้างขึ้น กล่าวคือ ไม่ถูก cutoff พร้อมกันทั้งสองทรานซิสเตอร์ [3]

สำหรับสมการของความต้านทานที่สร้างจาก PMOS และ NMOS ดังกล่าว สามารถเขียนได้ดังนี้ [3]

$$R = (\mu_n C_{ox}(W/L)_n(V_{GS} - V_{tn}) + \mu_p C_{ox}(W/L)_p(V_{SG} - |V_{tp}|))^{-1}$$

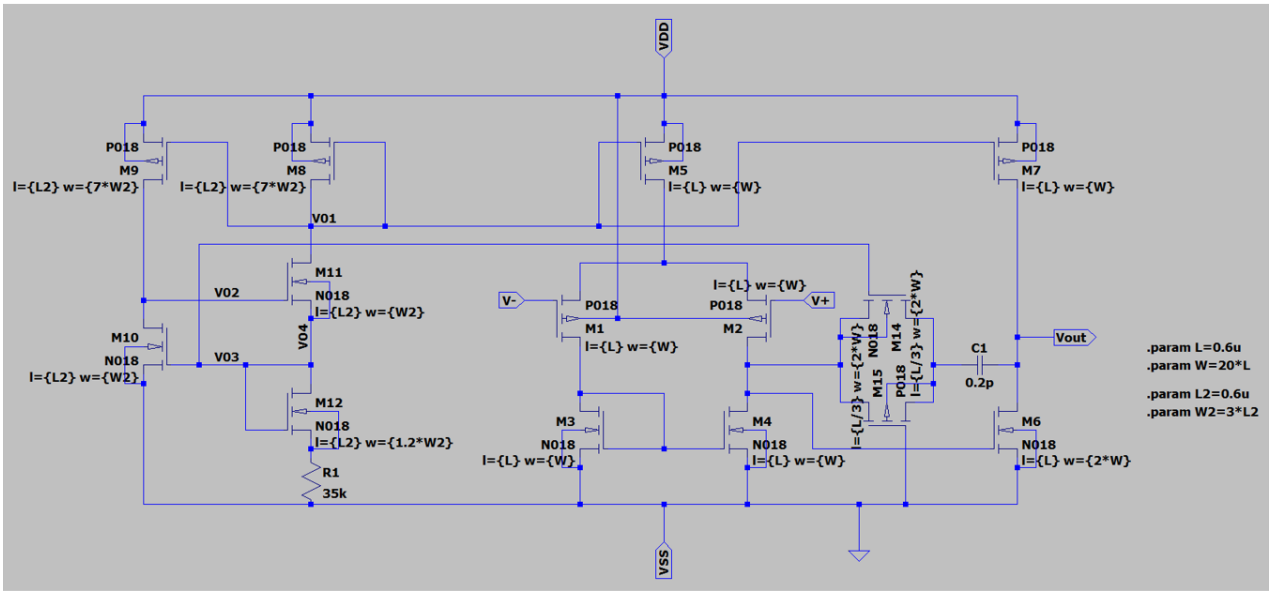
เราสามารถปรับอัตราส่วนของ W/L ของทรานซิสเตอร์ Q14 และ Q15 แบบ fine-tuning จนกระทั่งได้ผลตอบสนองเชิงความถี่ของอัตราขยายวงเปิดที่ใกล้เคียงกับแบบเดิม

สำหรับผลการทดลองที่ได้จากการจำลองวงจรในรูปที่ 1.16 เหมือนกับที่ได้จากวงจรในรูปที่ 1.10 ในทุกการทดลองที่ทำในหัวข้อที่ 1.1.4

การเปลี่ยนวงจรสร้างกระแสอ้างอิงเพื่อให้มีขนาดของตัวต้านทานที่ลดลง

ในวงจรเดิม (จากรูปที่ 1.10) การไบแอสกระแสที่มีค่าต่ำมาก (ประมาณ 0.5 µA) จำเป็นต้องใช้ตัวต้านทานที่ค่าใหญ่ การแก้ปัญหาเบื้องต้นคือการแทนที่ตัวต้านทานนั้นด้วย diode-connected MOSFET ไม่ว่าอย่างไรก็ตาม การทำเช่นนั้นจะทำให้กระแสอ้างอิงถูกรบกวนอย่างมากด้วย PVT variation หรือความเบี่ยงเบนจากค่าปกติที่อาจเกิดขึ้นกับกระบวนการผลิต, แรงดันจากแหล่งจ่าย และอุณหภูมิของวงจร เนื่องจากความต้านทานของ diode-connected MOSFET ขึ้นอยู่กับการเปลี่ยนแปลงดังกล่าวอย่างมาก

การแก้ปัญหาต่อมา คือการเปลี่ยนรูปแบบของวงจรอ้างอิงกระแสให้เป็นแบบ threshold voltage reference



รูปที่ 1.17: การเปลี่ยนวงจรสร้างกระแสอ้างอิงเพื่อให้มีขนาดของตัวต้านทานที่ลดลง

แทน ซึ่งได้แสดงในรูปที่ 1.18 [2] เพื่อให้กระแสอ้างอิงเป็นอิสระจากแรงดันจากแหล่งจ่ายและอุณหภูมิของวงจร จากรูปดังกล่าว เมื่อเปรียบเทียบกับรูปที่ 1.17 นั่นคือ ทรานซิสเตอร์ T1 และ T2 คือทรานซิสเตอร์ M10 และ M11 ตามลำดับ

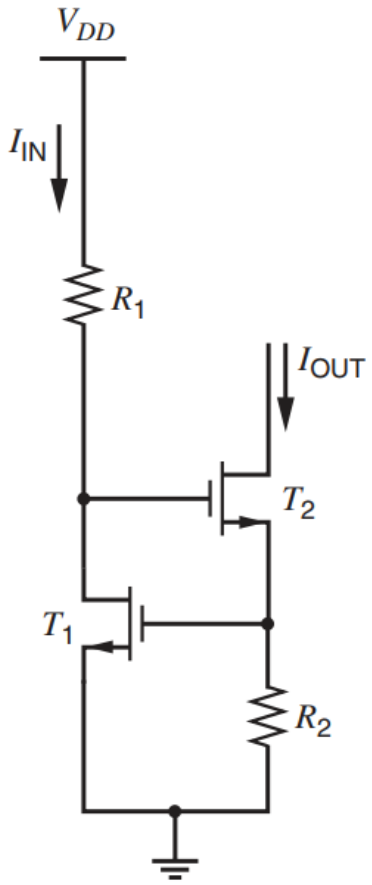
และตัวต้านทาน R2 ในรูปที่ 1.18 ก็คือแทนที่เป็นตัวต้านทาน R1 ต่ออนุกรมกับ diode-connected MOSFET ที่ใช้ M12 ในรูปที่ 1.17

นอกจากนั้น ตัวสร้างกระแสขาเข้า I_{IN} ในรูปที่ 1.18 สามารถสร้างแทนได้ด้วย current mirror ซึ่งใช้ทรานซิสเตอร์ M8 และ M9 ในรูปที่ 1.17 เพื่อให้กระแสของทั้งสองรางถูกผูกไว้อีกทีหนึ่ง

สมการความสัมพันธ์ระหว่าง I_{IN} และ I_{OUT} ของรูปที่ 1.18 สามารถเขียนได้ดังนี้ [2]

$$I_{OUT} = \frac{V_t + \sqrt{\frac{2I_{IN}}{k'(W/L)_1}}}{R_2}$$

ไม่ว่าอย่างไรก็ตาม การออกแบบวงจรในรูปที่ 1.17 ก็ยังมีประสิทธิภาพที่แย่ว่า 1.10 ซึ่งไม่คุ้มค่ากับการแลกมาของพื้นทีของวงจรที่ใช้บน IC ที่น้อยลง



รูปที่ 1.18: วงจรสร้างกระแสอ้างอิงแบบ threshold voltage reference

บทที่ 2

สรุป

2.1 ปัญหา อุปสรรค และข้อเสนอแนะ

เนื่องจากการทำ Analog Design จำเป็นต้องใช้ความรู้ทางด้านอิเล็กทรอนิกส์อีกมาก ดังนั้นจึงจะต้องอ่านหนังสือเพิ่อค้นคว้าด้วยตัวเอง ซึ่งหนังสือที่แนะนำ เช่น ของ Sedra และ Smith [5] และจะมีเนื้อหาที่ค่อนข้างลึกบางส่วนที่ในหนังสือ Sedra และ Smith ไม่ได้พูดถึงอีกมาก เช่น สัญญาณรบกวน (Noise) และการออกแบบวงจรสร้างกระแสอ้างอิงที่มีคุณสมบติเพิ่มเติม ฯลฯ ซึ่งแนะนำให้อ่านต่อไปในหนังสือของ Gray [2] และหนังสือการออกแบบวงจร CMOS ของ Razavi [4]

ภาคผนวก

ผลการทดลองทั้งหมด: คลิกที่นี่

วงจรและโค้ดทั้งหมดที่ใช้ในการจำลองวงจร: คลิกที่นี่

Code สำหรับการวัดค่าที่เกี่ยวกับผลตอบสนองเชิงความถี่ของอัตราขยายแบบวงเปิด ได้แก่ DC Gain, Crossover Frequency และ Phase Margin

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  6 15:19:30 2022

@author: nutchanonj
"""

# import PyLTSpice
from PyLTSpice.LTSpiceBatch import SimCommander
from PyLTSpice import LTSpice_RawRead

# import math function
from math import log10
from math import pi
from cmath import phase

# import
import numpy as np
from matplotlib import pyplot as plt

import time

# function to change complex number to abs
def to_dB(x):
    return 20*log10(abs(x))

# function to change complex number to phase
def to_degree(x):
    out = phase(x)/pi*180
    if 0 < out < 180:
        out = out - 360
    return out

# function to receive raw data and generate result
def run_result(exp_name):

    # read data from raw file
    LTR = LTSpice_RawRead.RawRead(f"{exp_name}.raw")
```

```
    print("Read Success")
    freq = np.array(LTR.get_trace('frequency'), dtype=np.longdouble)
    vout = np.array(LTR.get_trace('V(vout)'), dtype=np.clongdouble)
    vout_abs = np.fromiter(map(lambda x: to_dB(x), vout), dtype=np.longdouble)
    vout_phase = np.fromiter(map(lambda x: to_degree(x), vout), dtype=np.longdouble)

    # # plot the result
    # figure = plt.figure()
    # ax = figure.add_subplot(111)
    # plt.xscale("log")
    # ax.plot(freq, vout_abs)
    # figure.savefig(f'{exp_name}.png', dpi=300)

    # find dc gain
    dc_gain = vout_abs[0]

    # find crossover frequency and phase margin
    zero_crossing = int(np.where(np.diff(np.sign(vout_abs)))[0])
    cross_freq = freq[zero_crossing]/1000000 # unit in MHz
    phase_margin = vout_phase[zero_crossing] + 180

    return dc_gain, cross_freq, phase_margin

# export result
def export_result(export_name, dc_gain_data, cross_freq_data, phase_margin_data, rows,
columns):
    # export dc gain
    dc_gain_exp = np.reshape(dc_gain_data, (rows, columns))
    np.savetxt(f"{export_name}_dc_gain.csv", dc_gain_exp, delimiter=",", fmt='%.2f')
    # export crossover freq
    cross_freq_exp = np.reshape(cross_freq_data, (rows, columns))
    np.savetxt(f"{export_name}_cross_freq.csv", cross_freq_exp, delimiter=",", fmt='%.2f
')
    # export phase margin
    phase_margin_exp = np.reshape(phase_margin_data, (rows, columns))
    np.savetxt(f"{export_name}_phase_margin.csv", phase_margin_exp, delimiter=",", fmt='
%.2f')


# params
corners = ["SS","TT","FF"]
temps = [0,35,70]

LTC = SimCommander("SIM_ac_sweep.asc", parallel_sims=8)

dc_gain_data = []
cross_freq_data = []
phase_margin_data = []

for corner in corners:
    # run the simulation
    LTC.add_instruction(f".include cmos018_{corner}.lib")

    # change the power supply to 1.1 V
    Vsupply = 1.1
    LTC.set_component_value('Vsupply', Vsupply)

    for Vin in [0.1, 0.3, 0.5, 0.7, 0.9]:
        LTC.set_component_value('Vsource', Vin)
        LTC.set_component_value('Voffset', 0)
        for temp in temps:
            LTC.add_instruction(f".temp {temp}")
            exp_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}_{Vin:.1f}_{temp:.1f}"
```

```
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.remove_instruction(f".temp {temp}")

    for Voffset in [-0.45, 0, 0.45]:
        Vin = 0.55
        Vout = Vin + Voffset
        LTC.set_component_value('Vsource', Vin)
        LTC.set_component_value('Voffset', Voffset)
        for temp in temps:
            LTC.add_instruction(f".temp {temp}")
            exp_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}_{Vout:.1f}_{temp:.1f}"
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.remove_instruction(f".temp {temp}")

    # change the power supply to 1.8 V
    Vsupply = 1.8
    LTC.set_component_value('Vsupply', Vsupply)

    for Vin in [0.1, 0.4, 0.7, 1.0, 1.3, 1.6]:
        LTC.set_component_value('Vsource', Vin)
        LTC.set_component_value('Voffset', 0)
        for temp in temps:
            LTC.add_instruction(f".temp {temp}")
            exp_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}_{Vin:.1f}_{temp:.1f}"
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.remove_instruction(f".temp {temp}")

    for Voffset in [-0.8, -0.4, 0, 0.4, 0.8]:
        Vin = 0.9
        Vout = Vin + Voffset
        LTC.set_component_value('Vsource', Vin)
        LTC.set_component_value('Voffset', Voffset)
        for temp in temps:
            LTC.add_instruction(f".temp {temp}")
            exp_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}_{Vout:.1f}_{temp:.1f}"
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.remove_instruction(f".temp {temp}")

    # remove old instruction
    LTC.remove_instruction(f".include cmos018_{corner}.lib")

LTC.wait_completion()

for corner in corners:

    # change the power supply to 1.1 V
    Vsupply = 1.1
    for Vin in [0.1, 0.3, 0.5, 0.7, 0.9]:
        for temp in temps:
            exp_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}_{Vin:.1f}_{temp:.1f}"
            dc_gain, cross_freq, phase_margin = run_result(exp_name)
            dc_gain_data.append(dc_gain)
            cross_freq_data.append(cross_freq)
            phase_margin_data.append(phase_margin)

    rows = 5;
    columns = 3;
    export_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}"
    export_result(export_name, dc_gain_data, cross_freq_data, phase_margin_data, rows,
columns)
    dc_gain_data = []; cross_freq_data = []; phase_margin_data = [];

    for Voffset in [-0.45, 0, 0.45]:
```

```
        Vin = 0.55
        Vout = Vin + Voffset
        for temp in temps:
            exp_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}_{Vout:.1f}_{temp:.1f}"
            dc_gain, cross_freq, phase_margin = run_result(exp_name)
            dc_gain_data.append(dc_gain)
            cross_freq_data.append(cross_freq)
            phase_margin_data.append(phase_margin)

    rows = 3;
    columns = 3;
    export_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}"
    export_result(export_name, dc_gain_data, cross_freq_data, phase_margin_data, rows,
columns)
    dc_gain_data = []; cross_freq_data = []; phase_margin_data = [];

    # change the power supply to 1.8 V
    Vsupply = 1.8
    LTC.set_component_value('Vsupply', Vsupply)

    for Vin in [0.1, 0.4, 0.7, 1.0, 1.3, 1.6]:
        for temp in temps:
            exp_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}_{Vin:.1f}_{temp:.1f}"
            dc_gain, cross_freq, phase_margin = run_result(exp_name)
            dc_gain_data.append(dc_gain)
            cross_freq_data.append(cross_freq)
            phase_margin_data.append(phase_margin)

    rows = 6;
    columns = 3;
    export_name = f"SIM_ac_sweep_Vin_{corner}_{Vsupply:.1f}"
    export_result(export_name, dc_gain_data, cross_freq_data, phase_margin_data, rows,
columns)
    dc_gain_data = []; cross_freq_data = []; phase_margin_data = [];

    for Voffset in [-0.8, -0.4, 0, 0.4, 0.8]:
        Vin = 0.9
        Vout = Vin + Voffset
        for temp in temps:
            exp_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}_{Vout:.1f}_{temp:.1f}"
            dc_gain, cross_freq, phase_margin = run_result(exp_name)
            dc_gain_data.append(dc_gain)
            cross_freq_data.append(cross_freq)
            phase_margin_data.append(phase_margin)

    rows = 5;
    columns = 3;
    export_name = f"SIM_ac_sweep_Vout_{corner}_{Vsupply:.1f}"
    export_result(export_name, dc_gain_data, cross_freq_data, phase_margin_data, rows,
columns)
    dc_gain_data = []; cross_freq_data = []; phase_margin_data = [];
```

Code สำหรับการวัดการใช้พลังงานของวงจร

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  6 15:19:30 2022

@author: nutchanonj
"""
import re
```

```

# import PyLTSpice
from PyLTSpice.LTSpiceBatch import SimCommander
from PyLTSpice import LTSpice_RawRead

# import numpy
import numpy as np

LTC = SimCommander("SIM_oper_point.asc")

# params
corners = ["SS", "TT", "FF"]
temps = [0,35,70]

# array to store data
power_data = []

for corner in corners:
    LTC.add_instruction(f".include cmos018_{corner}.lib")

    for Vsupply in [1.8, 1.1]:
        LTC.set_component_value('Vsupply', Vsupply)

        for temp in temps:
            exp_name = "SIM_oper_point"

            LTC.add_instruction(f".temp {temp}")
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.wait_completion()

            file = open(f'{exp_name}.log', 'r')
            Lines = file.readlines()

            for line in Lines:
                if (line.find('pin:') != -1):
                    power_data.append( float(re.findall(r"[-+]?(?:\d*\.\d+|\d+)", line)
[0]) )

            LTC.remove_instruction(f".temp {temp}")

    # remove old instruction
    LTC.remove_instruction(f".include cmos018_{corner}.lib")

rows = 3;
columns = 6;

power_data = np.reshape(power_data, (rows, columns)).T
np.savetxt("SIM_Power.csv", power_data, delimiter=",", fmt='%.2f')
```

Code สำหรับการทดสอบสัญญาณรบกวนของวงจร

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  6 15:19:30 2022

@author: nutchanonj
"""
import re

# import PyLTSpice
from PyLTSpice.LTSpiceBatch import SimCommander
```

```
from PyLTSpice import LTSpice_RawRead

# import numpy
import numpy as np

LTC = SimCommander("SIM_noise.asc")

# params
corners = ["SS", "TT", "FF"]
temps = [0,35,70]

# array to store data
noise_data = []

for corner in corners:
    LTC.add_instruction(f".include cmos018_{corner}.lib")

    for Vsupply in [1.8, 1.1]:
        LTC.set_component_value('Vsupply', Vsupply)

        for temp in temps:
            exp_name = "SIM_noise"

            LTC.add_instruction(f".temp {temp}")
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.wait_completion()

            file = open(f'{exp_name}.log', 'r')
            Lines = file.readlines()

            for line in Lines:
                if (line.find('noiserms:') != -1):
                    noise_data.append( float(re.findall(r"[-+]?(?:\d*\.\d+|\d+)", line)
[0]) * 1000000 )

            LTC.remove_instruction(f".temp {temp}")

    # remove old instruction
    LTC.remove_instruction(f".include cmos018_{corner}.lib")

rows = 3;
columns = 6;

noise_data = np.reshape(noise_data, (rows, columns)).T
np.savetxt("SIM_noise.csv", noise_data, delimiter=",", fmt='%.2f')
```

Code สำหรับการทดสอบ Settling Time

```
# import PyLTSpice
from PyLTSpice.LTSpiceBatch import SimCommander
from PyLTSpice import LTSpice_RawRead

import pandas as pd
import numpy as np

corners = ["SS","TT","FF"]
temps = [0,35,70]
Vsupply_list = [1.8, 1.1]

LTC = SimCommander("SIM_transient.asc")

```

```
# array to store data
transient_data = []

for corner in corners:
    LTC.add_instruction(f".include cmos018_{corner}.lib")

    # change the power supply to 1.1 V
    for Vsupply in Vsupply_list:
        LTC.set_component_value('Vsupply', Vsupply)

        for temp in temps:
            LTC.add_instruction(f".temp {temp}")
            exp_name = f"SIM_transient_{corner}_{Vsupply:.1f}_{temp:.1f}"
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.wait_completion()
            LTR = LTSpice_RawRead.RawRead(f"{exp_name}.raw")
            print("Read Success")

            # read time data. (I don't know why time from raw file is sometimes negative
, so abs is needed.)
            time = np.abs(np.array(LTR.get_trace('time'), dtype=np.longdouble))# I don't
            # read magnitude data.
            mag = np.array(LTR.get_trace('V(vout)'), dtype=np.longdouble)
            # data to export.
            settling = []

            zero_crossings = np.where(np.diff(np.sign(mag - 0.505)))
            # Step value is 0.5 V. Wait until the signal drops to within 1%.

            transient_data.append((time[zero_crossings[0][1]] - 2e-6)/1e-6)
            # The start time is at 2us. the time unit exported is microsec.

            LTC.remove_instruction(f".temp {temp}")

    # remove old instruction
    LTC.remove_instruction(f".include cmos018_{corner}.lib")

rows = 3;
columns = 6;

transient_data = np.reshape(transient_data, (rows, columns)).T
np.savetxt("SIM_transient.csv", transient_data, delimiter=",", fmt='%.3f')
```

Code สำหรับการทดสอบ Total Harmonic Distortion

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  6 15:19:30 2022

@author: nutchanonj
"""
import re

# import PyLTSpice
from PyLTSpice.LTSpiceBatch import SimCommander
from PyLTSpice import LTSpice_RawRead

# import numpy
import numpy as np

LTC = SimCommander("SIM_fourier.asc")
```

```

# params
corners = ["SS", "TT", "FF"]
temps = [0,35,70]

# array to store data
THD_data = []

for corner in corners:
    LTC.add_instruction(f".include cmos018_{corner}.lib")

    for Vsupply in [1.8, 1.1]:
        LTC.set_component_value('Vsupply', Vsupply)
        if Vsupply == 1.8:
            LTC.set_component_value('VCM', 0.9)
            LTC.set_element_model('Vin', "SINE(0.9 0.8 1k)")
        else:
            LTC.set_component_value('VCM', 0.55)
            LTC.set_element_model('Vin', "SINE(0.55 0.45 1k)")

        for temp in temps:
            exp_name = "SIM_fourier"

            LTC.add_instruction(f".temp {temp}")
            LTC.run(run_filename=f"{exp_name}.net")
            LTC.wait_completion()

            file = open(f'{exp_name}.log', 'r')
            Lines = file.readlines()

            for line in Lines:
                if (line.find('Total Harmonic Distortion:') != -1):
                    THD_data.append( float(re.findall(r"[-+]?(?:\d*\.\d+|\d+)", line)
[0]) )

            LTC.remove_instruction(f".temp {temp}")

    # remove old instruction
    LTC.remove_instruction(f".include cmos018_{corner}.lib")

rows = 3;
columns = 6;

THD_data = np.reshape(THD_data, (rows, columns)).T
np.savetxt("SIM_fourier.csv", THD_data, delimiter=",", fmt='%.3f')
```

เอกสารอ้างอิง

[1] Nuno Brum. Pyltspice - LTSpice automation tools with Python. `https://pyltspice.readthedocs.io/en/latest/`, 2022.

[2] Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits*. Wiley, 5 edition, 2009.

[3] Sourav Nath, Naushad Laskar, Saurav Chanda, and Krishna Baishnab. A high gain, low power two stage opamp using self cascoding technique for low frequency application. 03 2017.

[4] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Education, 2 edition, 2017.

[5] A. Sedra and K. Smith. *Microelectronic Circuits*. Oxford University Press, 7 edition, 2015.