Lecture 1

1. Тооллын систем Лекц 1

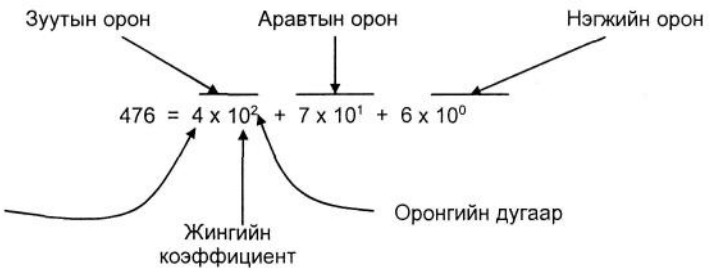
1.1 Тооллын систем. Тоо тоолол гэдэг бол бидэнд байнга хэрэглэгддэг чухал хэрэглэгдэхүүн юм. Тоо, тоололгүйгээр хүний амьдралыг төсөөлөх аргагүй юм. Бидний амьдралд хамгийн ойр тооллын систем бол *аравтын тооллын систем* юм. Аравтын тооллын системээс гадна *хоёртын, наймтын, арван зургаатын* гэх мэт өргөн хэрэглэгддэг тооллын системүүд байдаг. Эдгээр нь тоон технологид өргөн хэрэглэгддэг системүүд юм.

Тооллын системүүд хоорондоо ямар ялгаатай байгааг сонирхож үзье. Аравтын тооллын системийн хувьд тоог тэмдэглэхэд оролцдог 0,1,2,3,4,5,6,7,8,9 гэсэн 10 ширхэг тэмдэгттэй (цифр), орон бүр нь арав тоолоод дараагийн оронд шилждэг зэргээс нь үзэхэд яагаад *аравтын* гэж нэрлэх болсон нь тодорхой байна. Эндээс уг тооллын системийн үндэс буюу суурь тоо нь 10 болох нь харагдаж байна. Уг суурь тоог *жингийн коэффициент* гэж нэрлэдэг.Тооллын системүүдийн гол ялгагдах зүйл нь энэхүү *жингийн коэффициент* юм.

1.2 Аравтын тооллын систем. Аравтын тооллын системийн жингийн коэффициент нь 10, тоог тэмдэглэхэд 0-с 9 хүртэлх 10 ширхэг цифрийг ашигладаг. Тооллын системийн дурын тоог жингийн коэффициент,цифр болон оронгоор нь илэрхийлэн задлаж болно. Жишээ нь 476 гэсэн тоог доорхи байдлаар задлаж болно.

$$476 = 4 \times 10^2 + 7 \times 10^1 + 6 \times 10^0$$

Дээрхээс харахад 10 гэсэн жингийн коэффициентийг уг тоooны тухайн оронгийн дугаараар зэрэг дэвшүүлээд, гарсан үр дүнгээр тухайн оронд орших цифрийг үржүүлнэ. Энэ үйлдлийг орон бүрт хийгээд хооронд нь нэмбэл уг тоо гарч байна. Эндээс харахад тооллын системийн үндсэн параметрүүд нь *жингийн коэффициент, орон, цифр* болж байна.



Оронгийн утга баруунаас зүүн тийш шилжих тусам нэгээр нэмэгдэх буюу зүүн талын орон бүрийн жингийн коэффициентээр илэрхийлэгдэх үржигчийн утга баруун талынхаасаа 10 дахин илүү байна. Эсрэгээр баруун талынх нь зүүн талынхаасаа 10 дахин бага байна. Харин нэгжийн оронгоос цааш бутархай оронгуудыг ашиглаж байгаа үед оронгийн дугаар хасах тэмдэгтэйгээр тодоройлогдоно. Жишээ нь 957,28 гэсэн тоог задлая

$$957{,}28 = 9 \times 10^2 + 5 \times 10^1 + 7 \times 10^0 + 2 * 10^{-1} + 7 * 10^{-2}$$

1.3 Хоёртын тооллын систем. Хоёртын тооллын системийн хувьд жингийн коэффициент нь 2 бөгөөд тоог тэмдэглэхэд оролцдог цифрүүд нь ердөө л 0 болон 1 юм. Жишээ нь 10110 тоо нь аравтын тооллын системд арван мянга зуун арав гэсэн утгыг илэрхийлнэ. Харин хоёртын тооллын системийн утгыг нь аравтын тооллын системд хөрвүүлвэл 22 гэсэн утга гарч байна.

$$10110\,(2 = 22\,(10$$

Хоёртын тооллын системийн нэг оронг *бит* гэж нэрлэнэ. Тухайн тооны зүүн талынхыг ахлах бит, баруун талынхыг бага бит гэнэ. Хоёртын тооллын системийн тооны аравтын тооллын систем дэх утга нь дараахь байдлаар тодорхойлогдоно.

$$10110\,(2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 =$$

$$= 16 + 0 + 4 + 2 + 0 = 22\,(10$$

Энэ нь дээр аравтын тооллын системд тухайн тоог оронгийн жин, жингийн коэффициентээр илэрхийлж байсантай төстэй байна. Хоёртын тооллын системийн жингийн коэффициентийг (2-ийг) тухайн битийн дугаараар зэрэг дэвшүүлээд гарсан үр дүнгээр уг битийг үржүүлж байна. Энэ үйлдлийг бит тус бүрд хийгээд үржвэрүүдийг хооронд нь нэмэхэд аравтын тооллын систем дэх утга гарч байна.

1.4 Наймтын тооллын систем. Наймтын тооллын системийн хувьд тоог тоолоход оролцдог цифрүүд нь 0,1,2,3,4,5,6,7 гэсэн 8 ширхэг цифр байхаас гадна уг тооллын системийн жингийн коэффициент нь 8 байна. Наймтын тооллын системд байгаа тоог дээр өгүүлсэн байдлаар аравтын тооллын систем рүү хөрвүүлж болно.

$$472_{(8} = 4 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 = 256 + 56 + 2 = 314_{(10}$$

Наймтын тооллын системд 472 гэж бичигдэж байгаа тоо нь аравтын тооллын системд 314 гэсэн утгатай байна.

1.5 Арван зургаатын тооллын систем. Арван-зургаатын тооллын системд 16 ширхэг цифр ашиглан тоог тэмдэглэнэ: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F . Уг тооллын системийн жингийн коэффициент нь 16 байна.

$$13F_{(16} = 1 \times 16^2 + 3 \times 16^1 + 15 \times 16^0 = 256 + 48 + 15 = 319_{(10}$$

Арван зургаатын тооллын системд 13F тооны аравтын тооллын систем дэх бичиглэл нь 319 болж байна.

1.6 Тооллын системүүдэд арифметик үйлдэл хийх. Аравтын тооллын системээс бусад тооллын системүүдэд арифметик үйлдэл хийх нь зарчмын хувьд аравтын тооллын системд арифметик үйлдэл хийхтэй адил. Арифметик үйлдлийг хийж байх үед хэдийд орон шилжих вэ гэдгийг анхаарах хэрэгтэй. Нэмэх үйлдлийг авч үзье.

Хоёртын тооллын системийн хувьд нэмэх үйлдлийг хүснэгтлэн үзүүлбэл:

X	Y	X + Y
0	0	0
0	1	1
1	0	1
1	1	10

Хүснэгт 1.

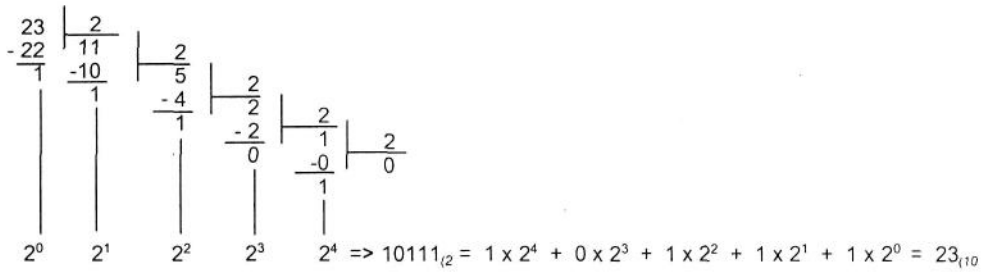
Учир нь 1 Хүснэгтээс харахад хоёр нэгийг хооронд нь нэмэхэд 10 гарч байна. нь хоёртын тооллын системийн тоолоход оролцдог хамгийн их цифр бөгөөд түүн дээр нэгийг нэмэхэд уг орон 0 болж дараагийн оронд шилжинэ.

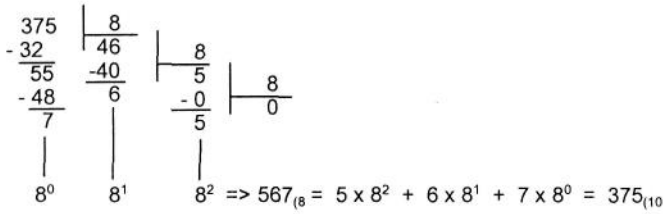
1.7 Тооллын системүүдийн харилцан хувиргалт. Бид эхний хэсэгт бусад тооллын системүүдээс аравтын тооллын систем рүү хувиргалт хийж байсан. Нэгэнт хоёрт, наймт, арван зургаатын тооллын системүүдээс аравтын тооллын системд хувиргалт хийх аргыг мэдэж байгаа тул энэ удаад аравтын тооллын системээс бусад тооллын системд хувиргалт хийхийг авч үзье. Аравтын тооллын системээс тухайн тооллын систем рүү хувиргалт хийхдээ доорхи дарааллыг баримтлана:

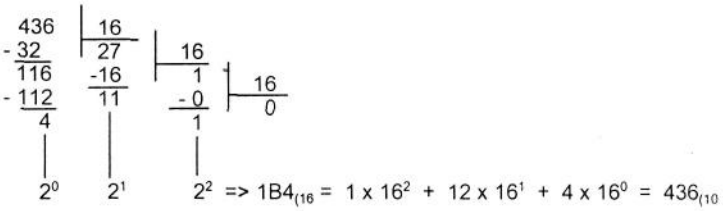
<

1. Тухайн тоог шилжүүлэх гэж байгаа тооллын системийн жингийн коэффициентод үлдэгдэлтэй хуваана.
2. Гарсан ноогдворыг ахин жингийн коэффициентод хуваана.
3. Энэ мэтчилэн ноогдворыг 0 болтол хуваана хуваана.
4. Эцэст нь хамгийн сүүлийн хуваах үйлдлийн үлдэгдлээс иэхлэн үлдэгдлүүдийг цувуулан бичинэ. Үр дүнд нь тухайн тооллын системд хувиргагдсан тоо гарна.

Жишээ болгож аравтын тооллын системийн 23-г хоёртsy тооллын систем рүү, 375-г наймтын тооллын систем рүү, 436-г арван зургаатын тооллын систем рүү хувиргая.







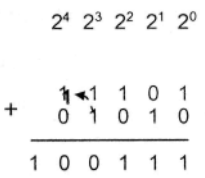
Хувиргалт бүрийн дараа эргүүлээд аравтын тооллын систем рүү хувиргаж шалгахад анхны тоо гарч, хувиргалт зөв хийгдсэнийг батлаж байна. Энэ нь зөвхөн бүхэл тооны хувьд хийгдэх үйлдэл юм. Харин бутархай тооны хувьд яах вэ? Энэ үед дараахь аргыг баримтална.

1. Бутархай тоог тухайн шилжүүлэх гэж байгаа тооллын системийн жингийн коэффициентээр үржүүлнэ.
2. Гарсан үржвэрийн бутархай хэсгийг мөн жингийн коэффициентээр үржүүлэх гэх мэтээр үргэлжлүүлнэ.
3. Дээр өгүүлсэн үйлдлийг үржвэрийн утга "О" болтол үргэлжлүүлнэ.
4. Үржвэрийн утга "О" болсоны дараа үржвэрүүдийн бүхэл хэсгийг нь салган авч, хамгийн эхний үржвэрийн бүхэл хэсгээс эхлэн цувуулан бичинэ.

Аравтын тооллын системийн 0,6875 тоог роёртын тооллын систем рүү, 0.546875 тоог наймт болон арван зургаатын тооллын систем рүү хөрвүүлээд дараа нь эргүүлж хувирган шалгаж үзье.

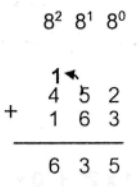
зүүн тийш нэг орон шилжиж байна (аравтын тооллын системийн хамгийн их цифр 9 дээр мөн 1-ийг нэмэхэд уг орон 0 болж ураш нэг орон шилждэгийг сана).

Хоёртын тооллын систем 11101 тоон дээр 1010 тоог нэмье.

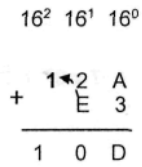


Нэмэх үйлдлийн явцал гуравдугаар бит дээр хоёр нэгийг хооронд нь нэмж байна. 1 + 1 = 10 тул нэг орон шилжинэ . Үр дүнд нь дөрөвдүгээр бит дээр ахин нэг 1 нэмэгдэж 1+1=10 болсоноор 11101 + 1010 = 100111 болжээ. Энэ үйлдлийг үнэн зев хийгдсэн эсэхийг аравтын тооллын системд харьцуулж үзье: $11101_{(2} = 29_{(10}$; $1010_{(2} = 10_{(10}$; $100111_{(2} = 39_{(10}$; $39_{(10} + 10_{(10} = 39_{(10}$

Наймтын тооллын системийн хувьд нэмэх үйлдпийн жишээ авъя.

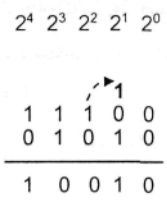


Нэгдүгээр оронд 5 болон 6 хоёрыг хооронд нь нэмэхэд орон шилжинэ: 5 + 6 = 13. Аравтын тооллын системд харцуулж үзвэл: $452_{(8} = 298_{(10}$; $163_{(8} = 115_{(10}$; $635_{(8} = 413_{(10}$; $298_{(10} + 115_{(10} = 413_{(10}$ Арван зургаатын тооллын системийн хувьд 2A + E3 үйлдлийг авч үзье.



Үр дүнг аравтын тооллын системд шалгаж үзвэл: $2A_{(16} = 42_{(10}$; $E3_{(16} - 227_{(10}$; $10D_{(16} = 269_{(10}$; $42_{(10} + 227_{(10} = 269_{(10}$

Харин хасах үйлдлийн хувьд урд талын хасагч тооны тухайн орон хасагдагч тооны тухайн оронгоос их байвал зүүн талын оронгоос нэгийг зээлнэ.



Дээрхи үйлдлийг аравтын тооллын системд шалгаж үзвэл: $11100_{(2} = 28_{(10}$; $1010_{(2} = 10_{(10}$; $10010_{(2} = 18_{(10}$; $28_{(10} + 10_{(10} = 28_{(10}$

Энэ мэтчилэн бусад тооллын системүүдэд хасах үйлдлийг хийж болно. Үржих болон хуваах үйлдлүүдийг аравтын тооллын системд хийгддэгийн адилаар бусад тооллын системүүдэд хийх боломжтой.

Ахлах бит

2 x 0,6875 = 1,375
2 x 0,375 = 0,750
2 x 0,750 = 1,500
2 x 0,500 = 1,000

Бага бит

$0,6875_{(10} = 0,1011_{(2} = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} = 0,5 + 0 + 0,125 + 0,0625 = 0.6875_{(10}$

8 x 0.546875 = 4,375
8 x 0,375000 = 3,000

$0,546875_{(10} = 0,43_{(8} = 4 \times 8^{-1} + 3 \times 8^{-2} = 0,5 + 0,046875 = 0,546875_{(10}$

16 x 0,546875 = 8,75
16 x 0,750000 = 12,0

$0,546875_{(10} = 0,8C_{(16} = 8 \times 16^{-1} + 12 \times 16^{-2} = 0,5 + 0.046875 = 0,546875_{(10}$

Дээр бид Аравтын тооллын системийг хоёрт болон наймт, арван зургаатын тооллын системүүдтэй хэрхэн харилцан хөрвех талаар авч үзлээ.

Харин хоёртын тооллын систем, наймт болон арван зургаатын тооллын системүүдийн тоог хооронд нь харилцан хөрвүүлэх тохиолдолд яах вэ? Доор эдгээр гурван тооллын системүүдийн натурал тоон дарааллын харгалзуулсан хүснэгтийг үзүүлжээ.

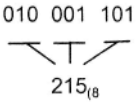
2-тын тооллын систем	8-тын тооллын систем	16-тын тооллын систем
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	10	8
1001	11	9
1010	12	A
1011	13	B
1100	14	C
1101	15	D
1110	16	E
1111	17	F
010 000	20	10
010 001	21	11
....	....	....
....	....	....
111 110	76	3E
111 111	77	3F
001 000 000	100	40
001 000 001	101	41
....	....	....
....	....	....
1111 1110	376	FE
1111 1111	377	FF
0001 0000 0000	400	100
0001 0000 0001	401	101
....	....	....
....	....	....

Хүснэгт 2.

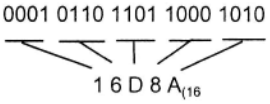
Хүснэгтээс арван зургаатын тооллын систем, наймтын тооллын системүүд нь хоёртын тооллын системтэй ямар уялдаа холбоотой болох онцлогийг харж болно. Наймтын тооллын системийн нэг оронд хоёртын тооллын системийн гурван орон харгалзаж байна. Мен арван зургаатын тооллын системийн нэг оронд хоёртын тооллын системийн дөрвөн орон харгалзана. Энэ онцлог нь наймтын тооллын системийн тоог хоёртын тооллын системийн тоотой, эсвэл арван зургаатын системийн тоог хоёртын тооллын системийн тоотой харилцан хөрвүүлэхэд маш хялбар болгож өгдөг. Харин аравтын тооллын системийн хувьд хоёртын тооллын системтэй харилцан хөрвүүлэлт хийхэд нийлээд төвөгтэй байдгийг бид мэдэх билээ.

Жишээ нь хүснэгтээс наймтын тооллын системийн 76 тоог хоёртын тооллын систем рүү хувиргая. Түүнд харгалзах хоёртын тооллын системийн утга нь 111 110 болох нь хүснэгтээс харагдаж байна. Наймтын тооллын системийн 76 тоо нь 7 болон 6 цифрээс тогтож байна. Тэгвэл эдгээр цифрүүд хоёртын тооллын системийн ямар утгатай харгалзаж байгааг хүснэгтээс харвал 111 болон 110 гасан утгуудад харгалзаж байна. Иймд эдгээр утгуудыг байршлын хувьд өөрчлөлгүй сольж тавсанаар хөрвүүлэлт маш хялбар хийгдэж байна.

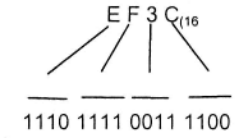
Мөн энэ шинж чанараар нь хоёртын тооллын системийн 10001101 тоог наймтын тооллын систем рүү хөрвүүлье. Үүний тулд уг тоог хамгийн бага битээс нь эхлэн гурав гурван битээр таслана. Дараа нь таслагдсан хэсгүүдэд харгалзах наймтын тооллын системийн цифрийг нөхөж тавьсанаар хөрвүүлэлт дуусна.



Дээрхи хөрвүүлэлтийг арван зургаатын тооллын систем болон хоёртын тооллын системийн хойронд хийж болно. Гол ялгаа нь наймтын тооллын системийн нэг орон хоёртын тооллйн системийн гурван оронтой харгалзаж байсан бол арван зургаатын системийн нэг орон хоёртын тооллйн системийн дөрвөн оронтой харгалзана. Хоёртын тооллын системээс 10110110110001010 тоог арван зургаатын тооллын систем рүү хөрвүүлье.



Арван зургаатын тооллын системээс хоёртын тооллын систем рүү EF3C тоог хөрвүүлье.



Тоон технологийн үндсэн тооллын систем болох хоёртын тооллын систем нь урт бичиглэлтэй, аравтын тооллын системтэй хөрвөхөд төвөгтэй байдаг зэргээсээ болоод хүн шууд ашиглахад нилээд хүндрэлтэй байдаг. Харин аравтын тооллын системийг бодвол арван зургаатын тооллын систем нь хоёртын тооллын системтэй харилцан хөрвехдөе амархан, мөн хоёртын тооллын системийг бодвол хүний ухамсар хүлээн авч боловсруулалт хийхэд арай дөхүү байдаг тул өргөн ашиглагдах болсон байна.

1.8 Тооллын системүүдийн гүйцээлтүүд*.Ихэнх* компьютерийн системүүдэд хасах үйлдлийг нэмэх үйлдлээр орлуулж гүйцэтгэдэг. Жишээ нь: X - Y үйлдлийг гүйцэтгэхийн тулд хасагч болох Y-ийн *гүйцээлтийг* олж түүнийгээ X дээр нэмнэ. Гэрсэн нийлбэр нь X - Y үйлдлийн ялгавартай тэнцэж байх ёстой. Иймд бидэнд энэхүү *гүйцээлт* хэмээгдээд байгаа утгыг хэрхэн гаргаж авах вэ? гэсэн асуудал гарч ирнэ.

1.9 Аравтын тооллын системийн гүйцээлтүүд*.hравтын* тооллын систедмд *гүйцээлтийг* хэрхэн олж ашиглаж байгааг авч үзье. Аравтын тооллын системд *9-н гүйцээлтийг* өргөн ашигладаг. Тухай тооны 9-н гүйцээлтийг олохдоо уг тооны орон бүрийг 9-с хасна: Жишээ нь 147 тооны 9-н гүйцээлтийг олъё. 999 - 147 = 852 . Тэгэхээр 147 тооны 9-н гүйцээлт нь 852 болж байна. 9-н гүйцээлтиийг олсон учраас гүйцээлтээ хасах үздлийг нэмэх үйлдэл болгоход хэрхэн ашигладаг талаар сонирхоё.

X – У ялгаварын утгыг нэмэх үйлдэл ашиглан олохдоо доорхи дарааллыг баримдална.

1. *Y-н (хасагчийн) 9-н гуйцээлтийг олно.*
2. *9-н гүйцээлтийг X (хасагдагч) дээр нэмнэ.*
3. *Гарсан нийлбэрийн урагш шилжсэн хамгийн ахлах орон болох орон шилжсэн 1-ийг* дарна.*
4. *Гарсан үр дүнгийн хамгийн бага орон дээр 1-ийг нэмнэ.*

Дарааллын дагуу зарим хасах үйлдлүүдийг нэмэх үйлдлээр олъё (гүйцээлт ашиглан).

а) 762 – 143 = 619 б) 501 – 85 = 416 в) 385,16 – 72,9 = 312,26

а) 143-н 9-н гүйцээлтийг олно. 999 - 143 = 856 . 9-н гүйцээлт нь 856 болж байна. Хасагдагч дээр 9-н гүйцээлтийг нэмнэ. Урагш шилжсэн хамгийн ахлах оронгийн 1-г дарна. Үр дүнд нь гарсан 3 оронтой тооны хамгийн бага орон дээр 1-г нэмнэ.

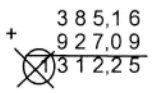
$$\begin{array}{r} 762 \\ +\ 856 \\ \hline \cancel{1}618 \end{array} \qquad \begin{array}{r} 618 \\ +\ 1 \\ \hline 619 \end{array}$$

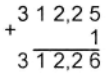
б) Хасагч 85 - н 9-н гүйцээлтийг олно. Хасагч болон хасагдагч хоёул ижил тооны оронтой байх ёстой. Иймд хасагчийн оронг 085 буюу 3 оронтой болгоно. 999 - 085 = 914. 9-н гүйцээлтийг хасагдагч дээр нэмнэ. Нийлбэрийн хамгийн ахлах орон болох орон шилжсэн 1-дарна. Энэ үйлдлийн дүнд гарсан тооны хамгийн бага (хамгийн ар талын) орон дээр 1-г нэмнэ.

$$\begin{array}{r} 501 \\ +\ 914 \\ \hline \cancel{1}415 \end{array} \qquad \begin{array}{r} 415 \\ +\ 1 \\ \hline 416 \end{array}$$

в) хасагч 72,9-ийн 9-н гүйцээлтийг олно. 999,99 - 72,9 = 927,09. Хасагдагч дээр 9-н гүйцээлтийг нэмнэ. Хамгийн

ахлах орон 1-г дарна. Хамгийн бага орон дээр 1-г нэмнэ.

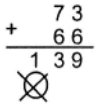




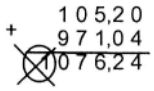
Дээр хийгдсэн бүх үйлдлүүдээс харахад нэмэх үйлдлийн үр дүнд гарсан алдааг засаж байна (дараа нь хамгийн бага орон дээр 1-г нэмэх замаар). Энэ байдлаас гарахын тулд 9-н гүйцээлтийн хамгийн бага орон дээр урьдчилаад 1-г нэмээд егч болдог. Үүнийг *10-н гүйцээлт* гэдэг. Жишээ болгож доорхи үйлдлүүдийг *аравтын гүйцээлт* ашиглан нэмэх үйлдлээр бодъё.

а) 73 – 34 = 39 б) 105,2 – 28,96 = 76,24

а) Эхлээд хасагчаас аравтын гүйцээлтийг олно. 99 - 34 = 65 => 65 + 1 = 66. Хасагдагч дээр олсон аравтын гүйцээлтээ нэмнэ. Гарсан нийлбэрийн хамгийн ахлах орон буюу орон шилжсэн 1-г дарна.



б) Хасагчаас аравтын гүйцээлтийг олно. 999,99 - 28,96 = 971,03 => 971,03 + 0.01 = 971,04 . Хасагдагч дээр аравтын гүйцээлтээ нэмнэ. Орон шилжсэн 1 болох хамгийн ахлах оронг дарна.



1.1 Хоёртын тооллын системийн гүйцээлтүүд. *Хоёртын* тооллын системд аравтын тооллын системийн нэгэн адилаар хамгийн их хэмжээтэй цифрээрээ нэрлэгдэх гүйцээлт бий. Энэ нь 1-н гүйцээлт юм. Хоёртын тооллын системийн хамгийн их цифр нь 1 учраас 1-н гүйцээлт гэх болсон. Мөн аравтын тооллын системийн нэгэн адилаар олно. Жишээ нь 101101,10 тооны 1-н гүйцээлтийг олъё. 111111,11-101101,10 =010010,01

Эндээс харахад хоёртын тооллын системд гүйцээлт олох үйлдэл нь маш хялбар, тухайн битийн цифрийг урвуугаар нь бичихэд л хангалттай байна. 1011101 тооны 1-н гүйцээлт нь 0100010 болох нь шууд харагдаж байна.

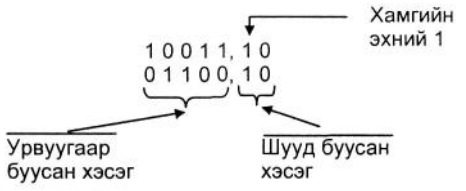
Хоёртын тооллын системд гүйцээлтийг ашиглан хасах үйлдлийг хийх нь аравтын тооллын системд хийдэгтэй яг ижил. Жишээ болгож дараахь ялгаврыг нэгийн гүйцээлт ашиглан олъё.

```
11010
01101
01101
```

```
  11010
+ 10010
  01100
+     1
  01101
```

Эндээс 1-н гүйцээлтээс *2-й* гүйцээлт гаргаж авч болох нь харагдаж байна. Аравтын тооллын системд хийдэгийн адилаар 1-н гүйцээлтийн хамгийн бага орон дээр нэгийг нэмж *2-й* гүйцээлтийг гаргаж авч болно. Мөн хоёртын тооллын системд 1-н гүйцээлтийг олоход хялбар байсан шиг 2-н гүйцээлтийг олох хялбар арга байдаг.

Хоёртын тооллын системд 2-н гүйцээлтийг олохын тулд тухайн тоог хамгийн бага оронгоос нь эхлээд шууд буулгаж эхлэх (1-н гүйцээлтийг олдогийн адилаар урвууг нь буулгаж эхлэхгүй) ба хамгийн эхний 1 цифрийг бууснаас хойш урвууг нь буулгаж эхэлнэ. Үүний дүнд 2-н гүйцээлт маш хялбархан олдоно. 10011,10 тооны 1-н гүйцээлт нь 01100,01. Харин 2-н гүйцээлтийг олбол 01100,01 + 0,01 = 01100,10. Одоо дээр өгүүлсэн хялбар аргаар 2-н гүйцээлтигй ахин олж тулгая.



2-н гүйцээлтийг ашиглан ялгаврыг нэмж олох үед урагш шилжсэн 1 болох хамгийн ахлах бит 1-г устгаж жинхэнэ хариуг гаргаж авна. Жишээ нь:

1.11 Наймт болон арван зургаатын тооллын системийн гүйцээлтүүд. Эдгээр тооллын системүүд хоёртын тооллын системтэй харилцан хөрвөхдөө хялбар байдаг. Харин хоёртын тооллын системд гүйцээлтүүдийг олоход амархан байдаг. Эдгээр шинж чанаруудыг харгаллзан үзэж наймт болон арванзургаатын системийн гүйцээлтүүдийг олохдоо хоёртын тооллын систем рүү хөрвүүлээд, гүйцээлтийг нь олсоны дараа эргүүлээд тухайн тооллын системд хувиргаж болох юм. Мөн өмнө өгүүлсэн аргаар чолж болно. Наймтын тоолын системд 7-н болон 8-н гүйцээлт гарч ирэх бөгөөд харин арван зургаатын системд 15-н болон 16-н гүйцээлтүүп гарч ирнэ. Эдгээрийг тусламжтайгаар нэмэх үйлдэлийг ашиглан ялгаврыг олох арга нь өмнө өгүүлсэн аравтын болон хоёртын тооллын системүүдийнхтэй ижил байх болно.

1.12 Тэмдэгт тоо. Вид гүйцээлтүүд тусламжтайгаар нэмэх үйлдэл ашиглан ялгаврыг олохдоо үргэлж хасагч нь хасагдагчаас модулиараа бага утгатай жишээг авч байсан. Өөрөөр хэлбэл ялгаврын утга үргэлж эерэг тоо байх жишээнүүдийг авч байв. Тэгвэл хасагч нь хасагдагчаас их буюу ялгавар нь сөрөг тоо тарах тохиолдолд яах вэ? Энэ асуудлыг шийдэхэд тулгарч байгаа гол бэрхшээл нь нэмэх тэмдэг (эерэг тоог илэрхийлэх) болон хасах тэмтгийг (сөрөг тоог илэрхийлэх) хэрхэн илэрхийлэх вэ? гэдэгт оршиж байна. Үүнийг шийдэхдээ тухайн тооны хамгийн ахлах битээр тогтоож өгдөг*. Өөрөөр хэлбэл хамгийн ахлах бит "О" бол тухайн тоо нь нэмэх тэмдэгтэй буюу эерэг тоо, харин хамгийн ахлах бит "1" бол тухайн тоо хасах тэмдэгтэй буюу сөрөг тоо болно. Ингэж тэмдэглэх нь тооцоолох техникт арифметик бодолт хийх замыг хялбарчилж өгдөг. 8 битийн компьютерүүдэд ихэвчлэн 8 битээр өгөгдөл боловсруулдаг тул 8 битийн өгөгдлийн хувьд 7-р бит (хамгийн ахлах бит) тэмдгийг илэрхийлж, бусад бага 7 бит нь өгөгдлийн утгыг илэрхийлнэ. 7-р бит буюу тэмдэгт битийг sign bit гэж тэмдэглэдэг. Аравтын тооллын системийн +75 тоог хоёртын тооллын системд илэрхийлбэл тэмдэгт бит буюу 7-р бит нь 0 байна.

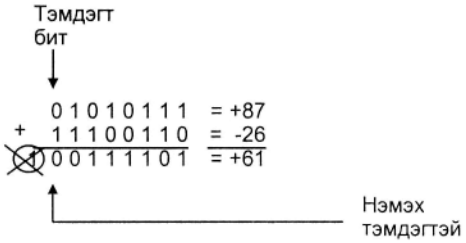
$+75_{(10} = 01001011_{(2}$ ← Тэмдэгт бит (+)

Нөгөө талаас хасах тоог илэрхийлэхдээ 7-р бит нь 1 байх ба үлдсэн 7 бит нь илэрхийлэх гэж байгаа тооны модулийн гүйцээлт байна (1-гүйцээлт юм уу, 2-н гүйцээлт байна). -52 тоог 1-н гүйцээлт ашиглан илэрхийлье.

$-52_{(10} = 11001011_{(2}$ ← Тэмдэгт бит (-)

Дээр бичигдсэн тооны бага 7 бит 1001011 $_{(2}$ нь $52_{(10}$ ($0110100_{(2}$) тооны 1-н гүйцээлт юм. Дээрхи тоонууд нь эерэг болон сөрөгөөр тэмдэглэгдсэн тоонууд. Харин эдгээр тоонуудыг тэмдэглэгдээгүй тоо гэж үзвэл 11001011(2 = $203_{(10}$ гэсэн өөр утга гарч байна. Компьютерийн техникт тухайн компьютер 1-н гүйцээлттэй арифметикт тулгуурдасан байна уу, эсвэл 2-н гүйцээлттэй арифметик дээр тулгуурласан байна уу гэдгээс хамаарч сөрөг тоог 1-н юм уу 2-н гүйцээлт ашиглан илэрхийлдэг. Орчин үеийн ихэнх компьютерүүд 2-н гүйцээлт ашигласан байдаг. Харин эерэг тоо нь аль ч арифметикт ижилхэн илэрхийлэгддэг: "+" тэмдгийг илэрхийлсэн хамгийн ахлах бит нь 0, Араас нь бусад битүүд бодит утгаараа дараалсан байна.

Харин нэмэх үйлдэл хийгдэх үед тэмдэгт бит нь бусад битүүдийн адилаар үйлдэлд оролцоно.Жишээ нь: (+87) + (-26) = +61 үйлдлийг 8 битийн 2-н гүйцээлт ашиглан гүйцэтгэе.



Бидний сая авч үзсэн жишээнд хасагч нь хасагдагчаас бага буюу ялгавар нь эерэг тоо байсан. Харин ялгаварын утга серег тоо байх тохиолдол гардаг. Дараахь тохиолдолд хасах үйлдлийн хэд хэдэн жишээг 1-н болон 2-н гүйцээлт ашиглан бодсон байна.

а) (+95) + (-63) в) (+42) + (-87) г) (-13) + (-59) д) (+38) + (-38) е) (-105) + (-120)

Бодолт:

$+95_{(10} = 01011111_{(2}$
$63_{(10} = 0111111_{(2}$
$-63_{(10} = 11000000_{(2}$
$-63_{(10} = 11000001_{(2}$

Хамгийн ахлах бит нь эерэг тоог илэрхийлж байна. 63-н 7 битийн утга.
-63 - г 1-н гүйцээлтээр илэрхийлсэн байдал. -63 - г 2-н гүйцээлтээр илэрхийлсэн байдал.

Дээрхи хөрвүүлэлтүүдээс харахад тухайн сөрөг тоог илэрхийлэхдээ гүйцээлтүүдийн өмнө (гүйцээлтүүд нь 7 битээр илэрхийлэгдэнэ) "-" тэмдэгийг илэрхийлэх 1-г тавьж өгсөнөөр шийдэгдэнэ. Одоо аравтын арифметик, 1-н гүйцээлттэй арифметик, 2-н гүйцээлттэй арифметик туе бүрээр үйлдлийг гүйцэтгэе.

+95
-63
+32

```
 0 1 0 1 1 1 1 1
 1 1 0 0 0 0 0 0
(1)0 0 0 1 1 1 1 1
               1
 0 0 1 0 0 0 0 0  = +32
```

```
 0 1 0 1 1 1 1 1
 1 1 0 0 0 0 0 1
(1)0 0 1 0 0 0 0 0  = +32
```

б)

$+42_{(10} = 00101010_{(2}$
$87_{(10} = 1010111_{(2}$
$-87_{(10} = 10101000_{(2}$
$-87_{(10} = 10101001_{(2}$

Хамгийн ахлах бит нь эерэг тоог илэрхийлж байна. 87-н 7 битийн утга.
-87 - г 1-н гүйцээлтээр илэрхийлсэн байдал. -87 - г 2-н гүйцээлтээр илэрхийлсэн байдал.

+42
-87
-45

```
 0 0 1 0 1 0 1 0
 1 0 1 0 1 0 0 0
 1 1 0 1 0 0 1 0  = -45
```

0101101=45 –н 1-н гүйцээлт

```
 0 0 1 0 1 0 1 0
 1 0 1 0 1 0 0 1
 1 1 0 1 0 0 1 1  = -45
```

0101101=45 –н 2-н гүйцээлт

Энэ жишээнд хасагч нь хасагдагчаас модулиараа их байна. Иймд ялгавар хасах тоо гарчээ. Тэмдэгт бит бодолтонд оролцож байгаагаас гадна ялгаврын утгад байрлахдаа мөн тэмдгийг илэрхийлж байна. Гэхдээ ялгаврын утга (7 битээр илэрхийлэгдсэн хэсэг нь) хэдийн гүйцэлттэй арифметику үйлдэл гүйцэтгэсэн байна вэ гэдгээс шалтгаалан гүйцэлтээр илэрхийлэгдэх утга гарч байна. Өөрөөр хэлбэл, Ялгавар сөрөг тоо гарсан бол түүний тоон утга нь гүйцээлтээрээ илэрхийлэгдсэн байна.

в)

$13_{(10} = 0001101_{(2}$

$-13_{(10} = 11110010_{(2}$

$-13_{(10} = 11110011_{(2}$

$59_{(10} = 0111011_{(2}$

$-59_{(10} = 11000100_{(2}$

$-59_{(10} = 11000101_{(2}$

-13
-59
-72

13-н 7 битийн утга.

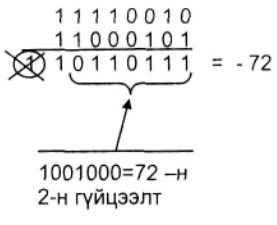
-13- г 1-н гүйцээлтээр илэрхийлсэн байдал.

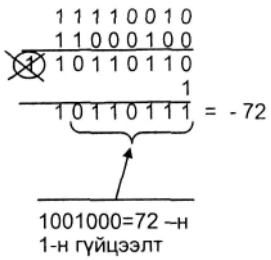
-13- г 2-н гүйцээлтээр илэрхийлсэн байдал.

59-н 7 битийн утга.

-59 - г 1-н гүйцээлтээр илэрхийлсэн байдал.

-59 - г 2-н гүйцээлтээр илэрхийлсэн байдал.





Энэ жишээнд хоёр сөрөг тоог хооронд нэмсэн байна. Ялшавар нь мөн л сөрөг тоо. Дээр өгүлсэний адилаар тэмдэгт бит мөн бодолтонд оролцож байгаагаас гадна тоон утга нь гүйцээлтээрээ илэрхийлэгдсэн байна.

г)

$+38_{(10} = 00100110_{(2}$

$-38_{(10} = 11011001_{(2}$

$-38_{(10} = 11011010_{(2}$

38
-38
0

+ 13-Н 7 битийн утга ("+" тэмдэгтэй).

-13- г 1-н гүйцээлтээр илэрхийлсэн байдал.

-13- г 2-н гүйцээлтээр илэрхийлсэн байдал.

```
  0 0 1 0 0 1 1 0
  1 1 0 1 1 0 1 0
  0 0 0 0 0 0 0 0   = + 0
```

00000000=0 –н
2-н гүйцээлт

```
  0 0 1 0 0 1 1 0
  1 1 0 1 1 0 0 1
  1 1 1 1 1 1 1 1   = - 0
```

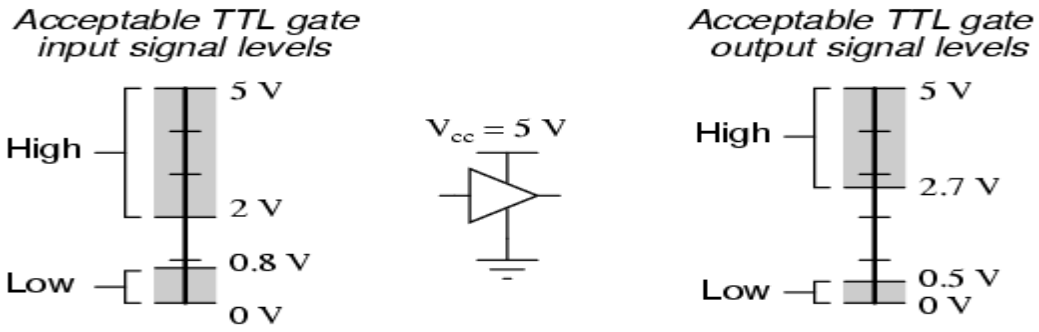
00000000=0 –н
1-н гүйцээлт

Энэ жишээнээс харахад 1-н гүйцээлттэй арифметикийн үр дүнд сөрөг "О" (хасах) гарч байна. Харин 2-н гүйцээлттэй арифметик ашигласан тооцооны үр дүнд эерэг "О" (нэмэх) гарч байна. Мэдээж "О"-н хувьд тэмдэг хэрэггүй нь оялгомжтой.

1.12 Логик дохионы хүчдэлийн төвшин.

Логик хэлхээ нь оролт гаралтын "өндөр" (1) ба "нам" (0) гэсэн хоёрхон дохиотой байхаар зохион бүтээгддэг. Тэжээлийн үүсгүүрийн бүтэн хүчдэл "өндөр" төлвийг харин тэг хүчдэл "нам" төлвийг илэрхийлнэ. Хэрэв бид төгс ертөнцөд амьдардагсан бол эдгээр логик төлвүүдийн хүчдлийн төвшин хэзээч өөрчлөгдөхгүй ("өндөр " төлөв нь хэзээ ч тэжээлийн үүсгүүрээс буухгүй, "нам" төвшин нь хэзээ ч тэгээс хэтрэхгүй) байхсан. Харамсалтай нь бид бодит ертөнцөд амьдарч байгаа учираас эдгээр логик төлвийн хүчдэлийн төвшин нь хэлхээний тэжээлийн уналт зэрэгээс болж ховорхон тохиолдол төгс утгаа авдаг. Тийм болохоор логик төвшингүүд нь тодорхой хүчдлийн хязгаараар илэрхийлэгддэг.

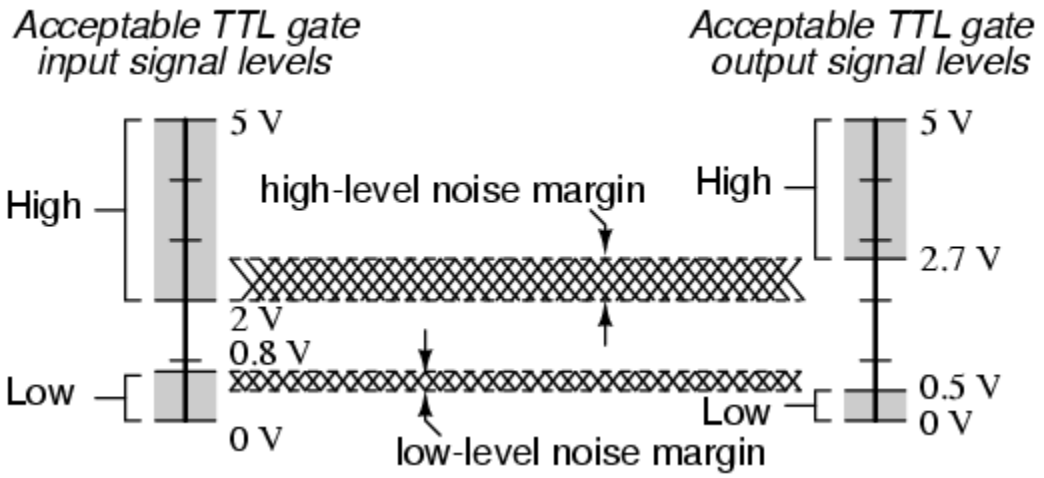
ТТЛ логик элементийн тэжээл нь 5В, +/-0.25В байдаг. Идиал нөхцөлд ТТЛ "өндөр" дохио нь яг 5В, ТТЛ "нам" дохио нь яг 0.00В байх ёстой. Гэвч бодит ТТЛ хэлхээ тийм өндөр нарийвчлалтай идиал дохиог гаргаж чаддаггүй. Иймд тэдгээрийг идиал "өндөр", "нам" дохионоос тодорхой хэмжээний өөрчлөлттөй "зөвшөөрөгдөх"дохионд хэвийн ажиллахаар зохион бүтээдэг. "Зөвшөөрөгдөх" оролтын дохионы хүчдэлийн хүрээ нь "нам" логик төлөвт 0-0.8 вольт, "өндөр" логик төлөвт 2-5 вольтын хооронд байдаг бол "зөвшөөрөгдөх" гаралтын дохионы хүчдэлийн хүрээ нь "нам" логик төлөвт 0-0.5 вольт, "өндөр" логик төлөвт 2.7-5 вольтын хооронд байх ёстой.



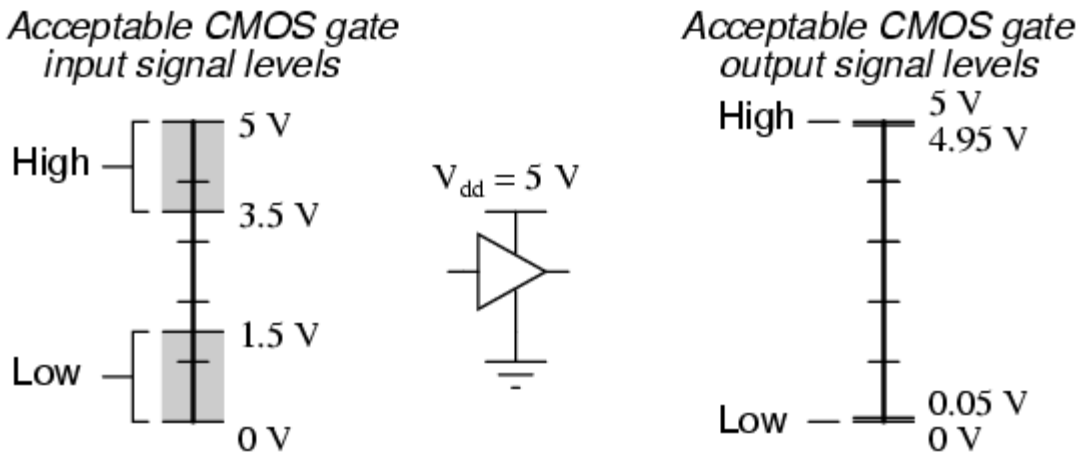
Хэрэв ТТЛ элементийн оролтонд 0.8-2 вольтын хооронд хүчдэл өгөдвөл ТТЛ элемент ямар нэгэн тодорхой үр дүн үзүүлэхгүй. Иймдохио нь элементүүдийн хувьд *тодорхой бус* дохио учираас үйлдвэрлэгчид ямар үр дүн гарах тухай баталгаа өгдөггүй.

Зургаас харахад элементүүдийн оролт, гаралтын дохионы зөвшөөрөгдөх хүрээ нь өөр өөр байна. Энэ нь

As you can see, the tolerable ranges for output signal levels are narrower than for input signal levels, to ensure that any TTL gate outputting a digital signal into the input of another TTL gate will transmit voltages acceptable to the receiving gate. The difference between the tolerable output and input ranges is called the *noise margin* of the gate. For TTL gates, the low-level noise margin is the difference between 0.8 volts and 0.5 volts (0.3 volts), while the high-level noise margin is the difference between 2.7 volts and 2 volts (0.7 volts). Simply put, the noise margin is the peak amount of spurious or "noise" voltage that may be superimposed on a weak gate output voltage signal before the receiving gate might interpret it wrongly:

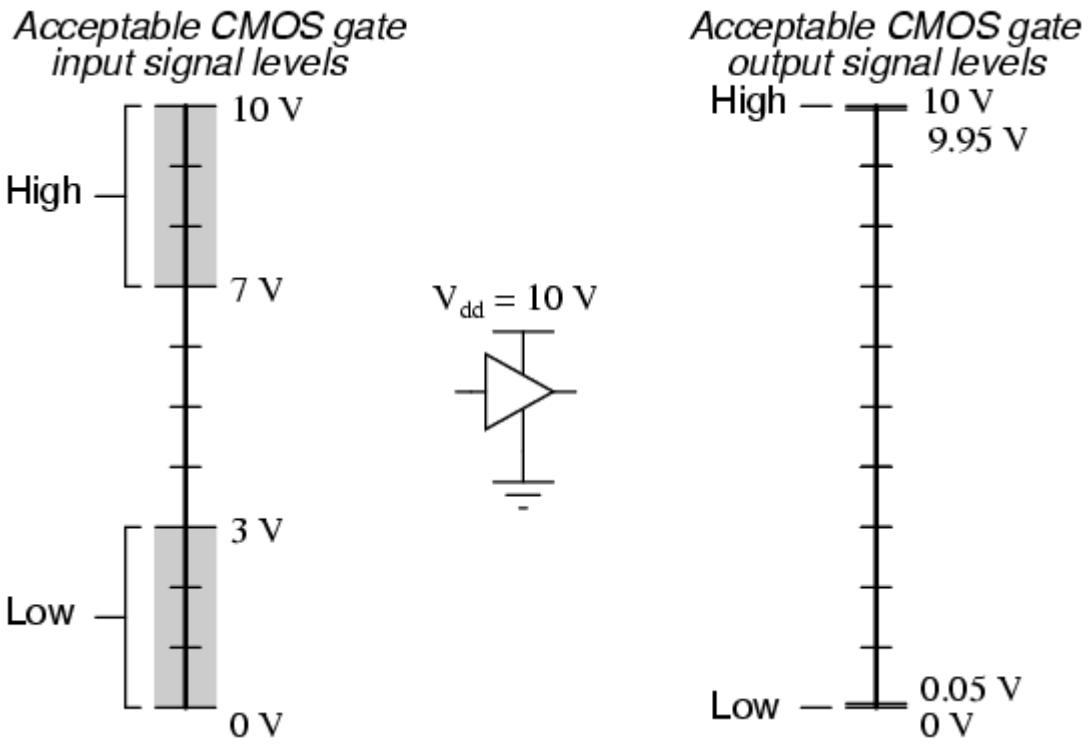


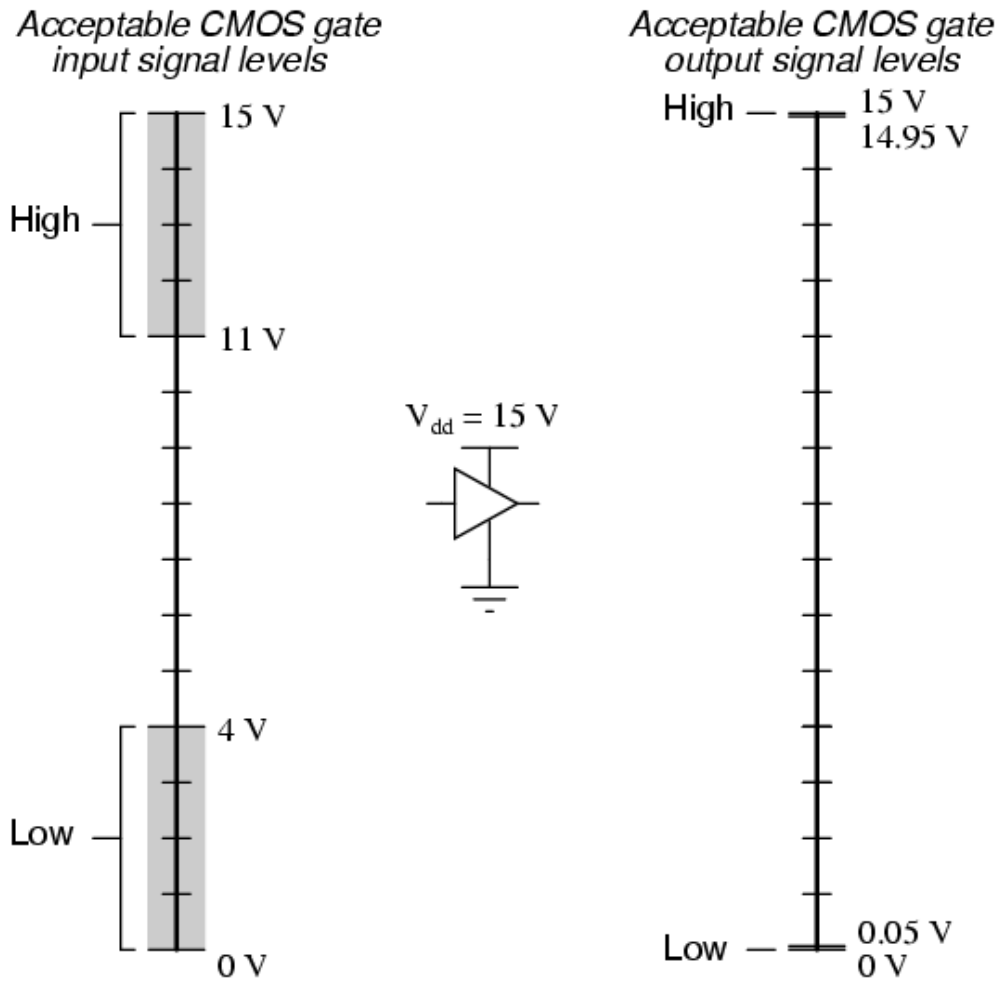
CMOS gate circuits have input and output signal specifications that are quite different from TTL. For a CMOS gate operating at a power supply voltage of 5 volts, the acceptable input signal voltages range from 0 volts to 1.5 volts for a "low" logic state, and 3.5 volts to 5 volts for a "high" logic state. "Acceptable" output signal voltages (voltage levels guaranteed by the gate manufacturer over a specified range of load conditions) range from 0 volts to 0.05 volts for a "low" logic state, and 4.95 volts to 5 volts for a "high" logic state:



It should be obvious from these figures that CMOS gate circuits have far greater noise margins than TTL: 1.45 volts for CMOS low-level and high-level margins, versus a maximum of 0.7 volts for TTL. In other words, CMOS circuits can tolerate over twice the amount of superimposed "noise" voltage on their input lines before signal interpretation errors will result.

CMOS noise margins widen even further with higher operating voltages. Unlike TTL, which is restricted to a power supply voltage of 5 volts, CMOS may be powered by voltages as high as 15 volts (some CMOS circuits as high as 18 volts). Shown here are the acceptable "high" and "low" states, for both input and output, of CMOS integrated circuits operating at 10 volts and 15 volts, respectively:


Acceptable CMOS gate input signal levels
10 V
High
7 V
3 V
Low
0 V
V_{dd} = 10 V
Acceptable CMOS gate output signal levels
High
10 V
9.95 V
Low
0.05 V
0 V

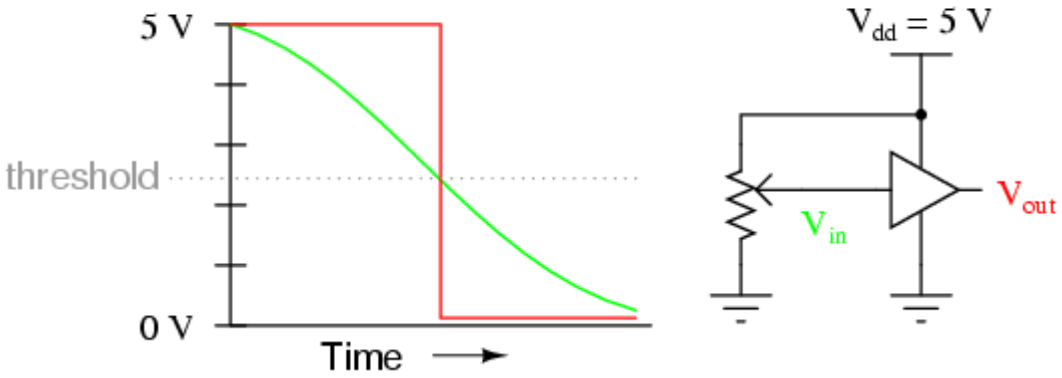


The margins for acceptable "high" and "low" signals may be greater than what is shown in the previous illustrations. What is shown represents "worst-case" input signal performance, based on manufacturer's specifications. In practice, it may be found that a gate circuit will tolerate "high" signals of considerably less voltage and "low" signals of considerably greater voltage than those specified here.

Conversely, the extremely small output margins shown -- guaranteeing output states for "high" and "low" signals to within 0.05 volts of the power supply "rails" -- are optimistic. Such "solid" output voltage levels will be true only for conditions of minimum loading. If the gate is sourcing or sinking substantial current to a load, the output voltage will not be able to maintain these optimum levels, due to internal channel resistance of the gate's final output MOSFETs.

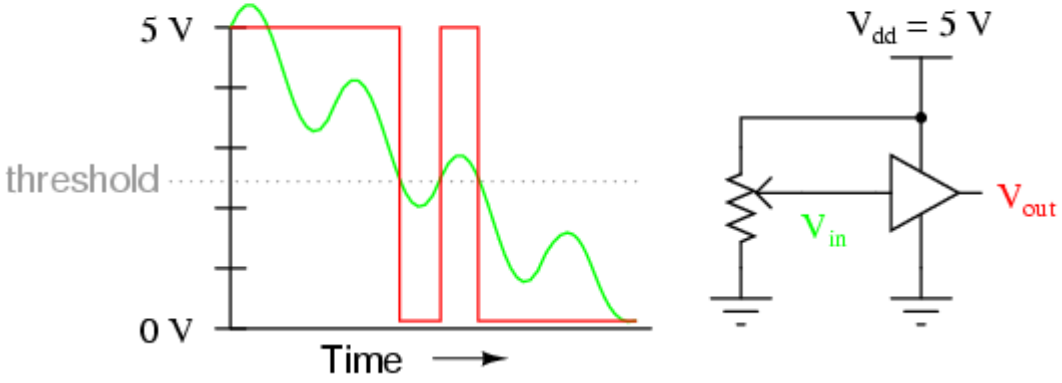
Within the "uncertain" range for any gate input, there will be some point of demarcation dividing the gate's actual "low" input signal range from its actual "high" input signal range. That is, somewhere between the lowest "high" signal voltage level and the highest "low" signal voltage level guaranteed by the gate manufacturer, there is a threshold voltage at which the gate will *actually* switch its interpretation of a signal from "low" or "high" or vice versa. For most gate circuits, this unspecified voltage is a single point:

Typical response of a logic gate to a variable (analog) input voltage

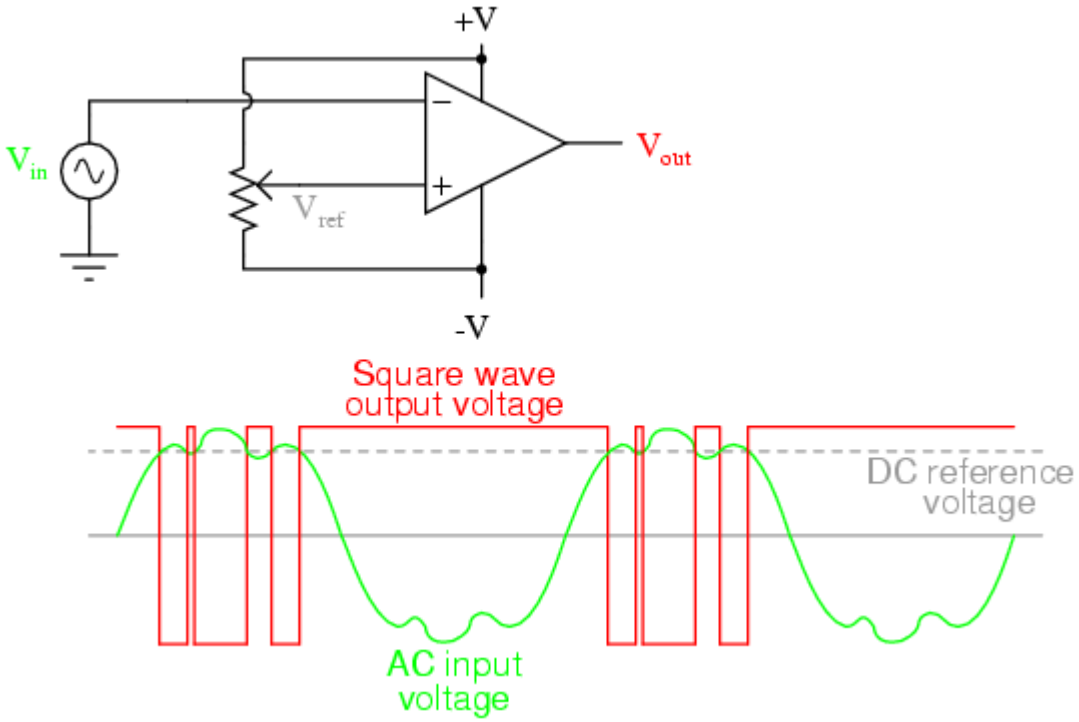


In the presence of AC "noise" voltage superimposed on the DC input signal, a single threshold point at which the gate alters its interpretation of logic level will result in an erratic output:

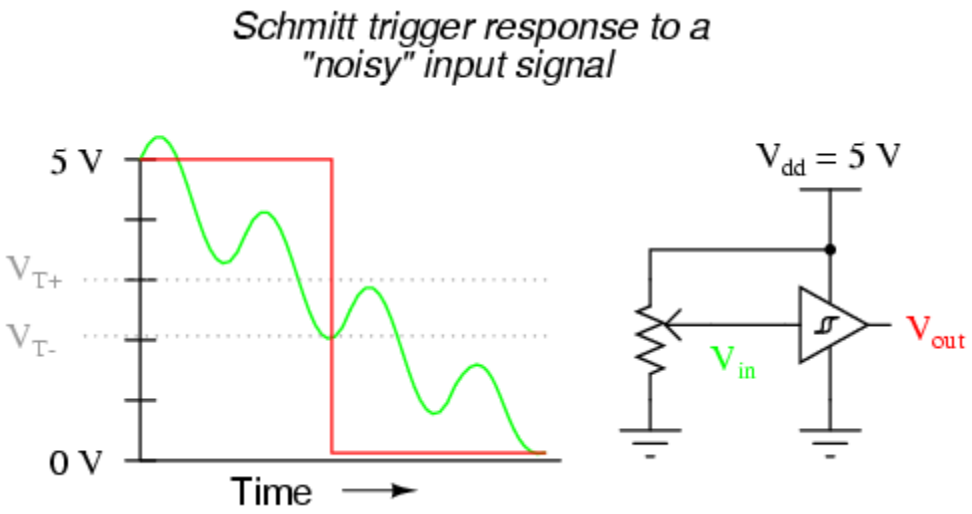
Slowly-changing DC signal with AC noise superimposed



If this scenario looks familiar to you, its because you remember a similar problem with (analog) voltage comparator op-amp circuits. With a single threshold point at which an input causes the output to switch between "high" and "low" states, the presence of significant noise will cause erratic changes in the output:



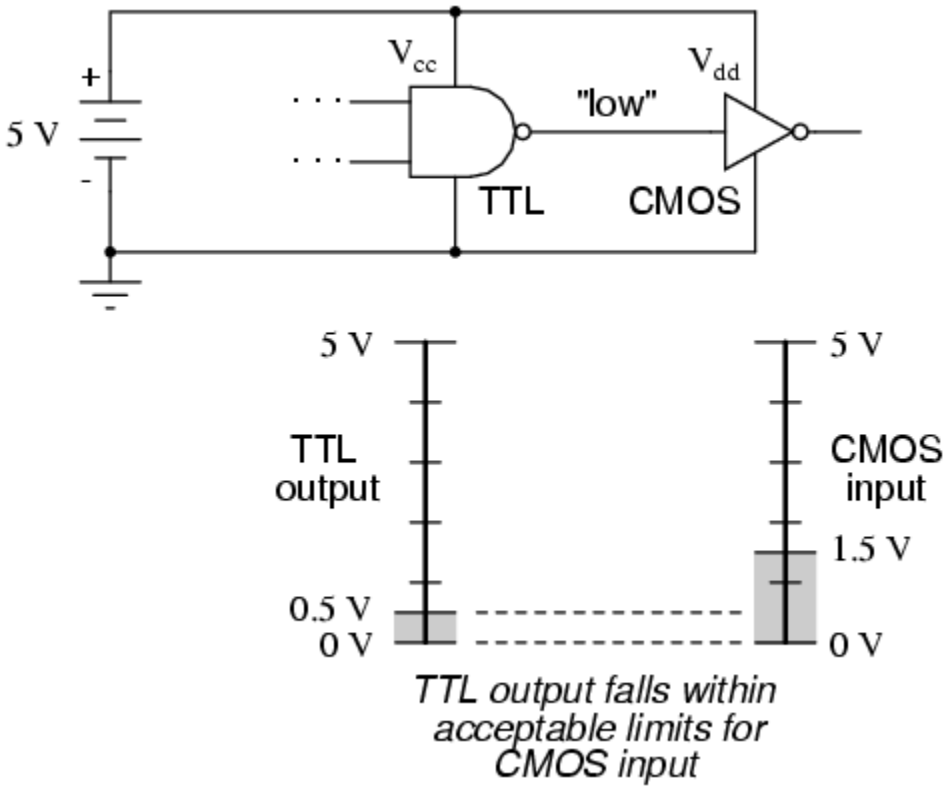
The solution to this problem is a bit of *positive feedback* introduced into the amplifier circuit. With an op-amp, this is done by connecting the output back around to the noninverting (+) input through a resistor. In a gate circuit, this entails redesigning the internal gate circuitry, establishing the feedback inside the gate package rather than through external connections. A gate so designed is called a *Schmitt trigger*. Schmitt triggers interpret varying input voltages according to *two* threshold voltages: a *positive-going* threshold (V_{T+}), and a *negative-going* threshold (V_{T-}):



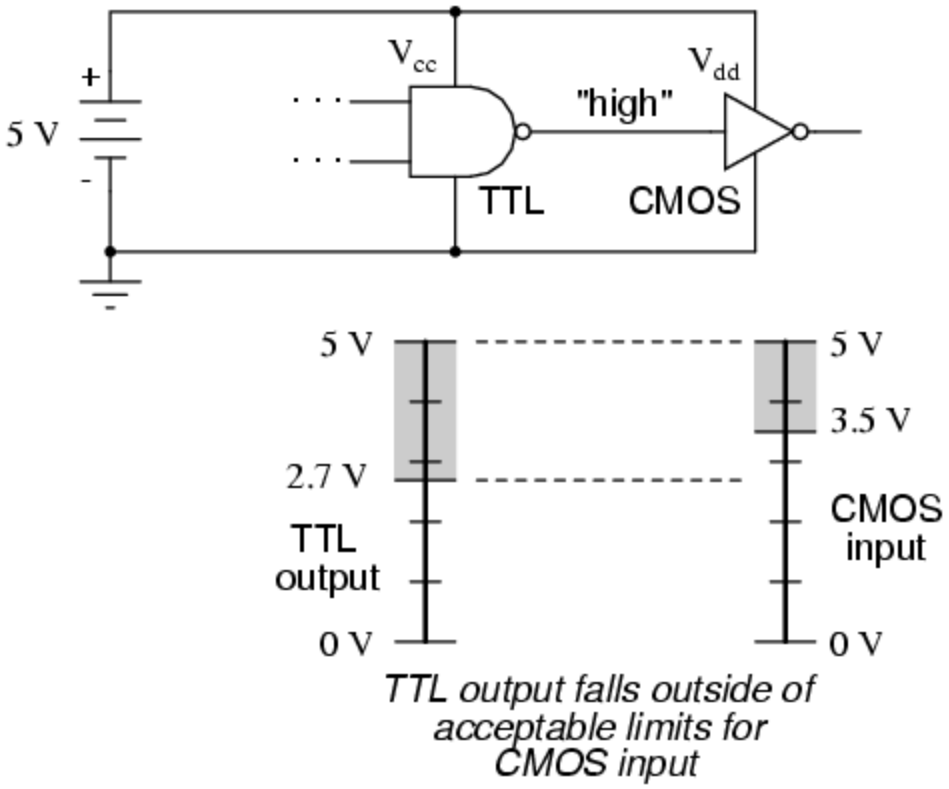
Schmitt trigger gates are distinguished in schematic diagrams by the small "hysteresis" symbol drawn within them, reminiscent of the B-H curve for a ferromagnetic material. Hysteresis engendered by positive feedback within the gate circuitry adds an additional level of noise immunity to the gate's performance. Schmitt trigger gates are frequently used in applications where noise is expected on the input signal line(s), and/or where an erratic output would be very detrimental to system performance.

The differing voltage level requirements of TTL and CMOS technology present problems when the two types of gates are used in the same system. Although operating CMOS gates on the same 5.00 volt power supply voltage required by the TTL gates is no problem, TTL output voltage levels will not be compatible with CMOS input voltage requirements.

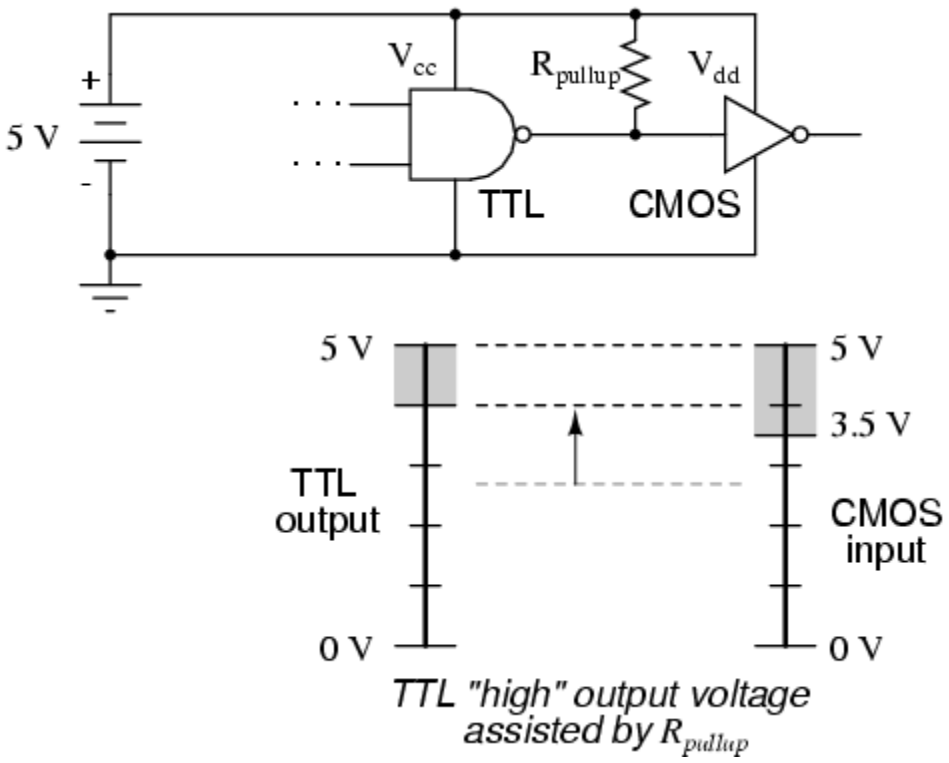
Take for instance a TTL NAND gate outputting a signal into the input of a CMOS inverter gate. Both gates are powered by the same 5.00 volt supply (V_{cc}). If the TTL gate outputs a "low" signal (guaranteed to be between 0 volts and 0.5 volts), it will be properly interpreted by the CMOS gate's input as a "low" (expecting a voltage between 0 volts and 1.5 volts):



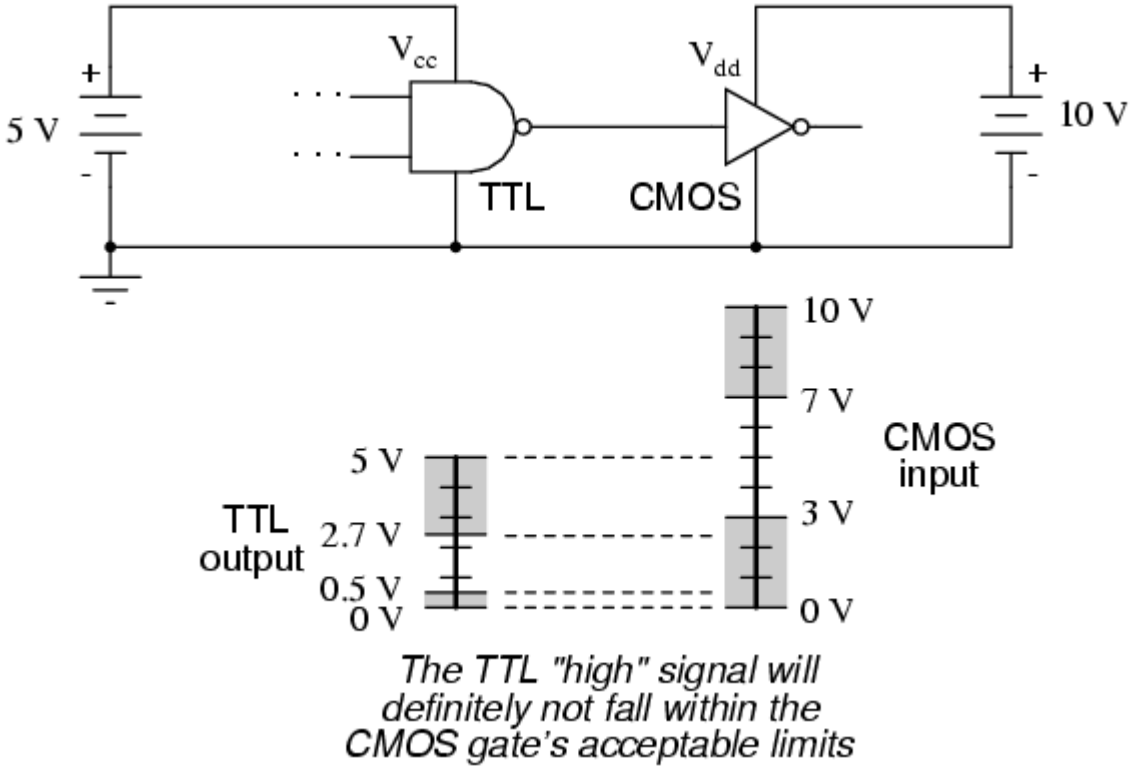
However, if the TTL gate outputs a "high" signal (guaranteed to be between 5 volts and 2.7 volts), it *might not* be properly interpreted by the CMOS gate's input as a "high" (expecting a voltage between 5 volts and *3.5* volts):



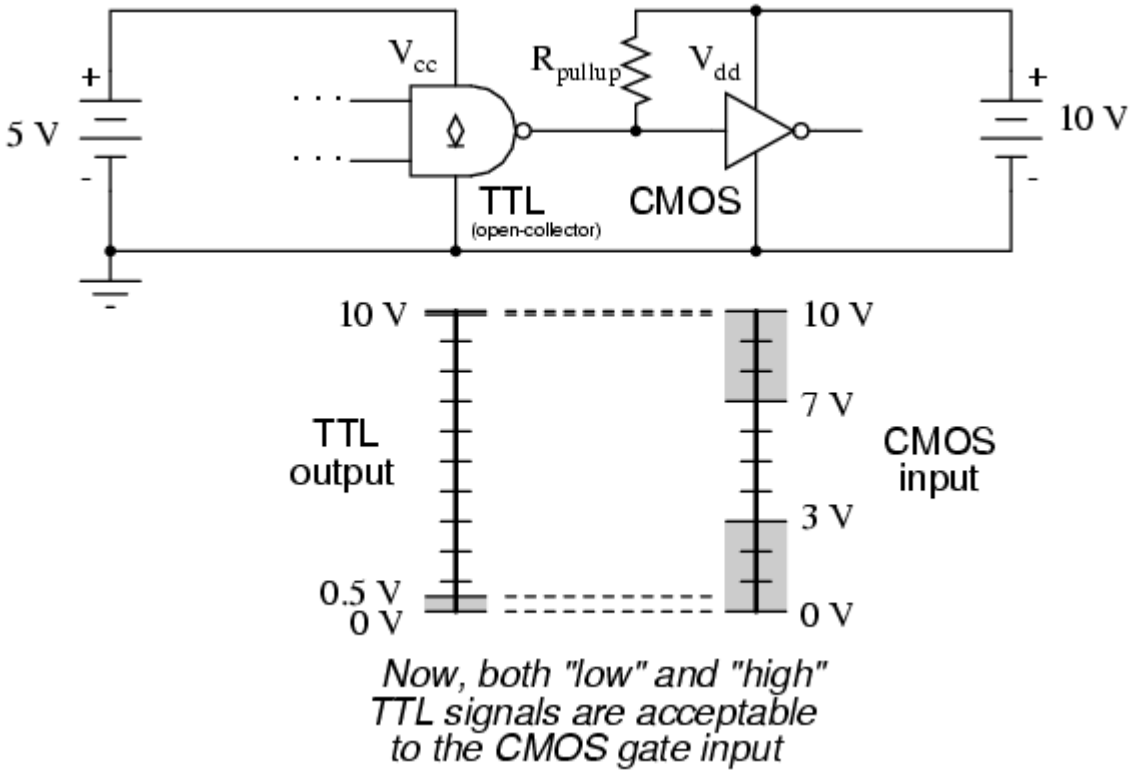
Given this mismatch, it is entirely possible for the TTL gate to output a valid "high" signal (valid, that is, according to the standards for TTL) that lies within the "uncertain" range for the CMOS input, and may be (falsely) interpreted as a "low" by the receiving gate. An easy "fix" for this problem is to augment the TTL gate's "high" signal voltage level by means of a pullup resistor:



Something more than this, though, is required to interface a TTL output with a CMOS input, if the receiving CMOS gate is powered by a greater power supply voltage:

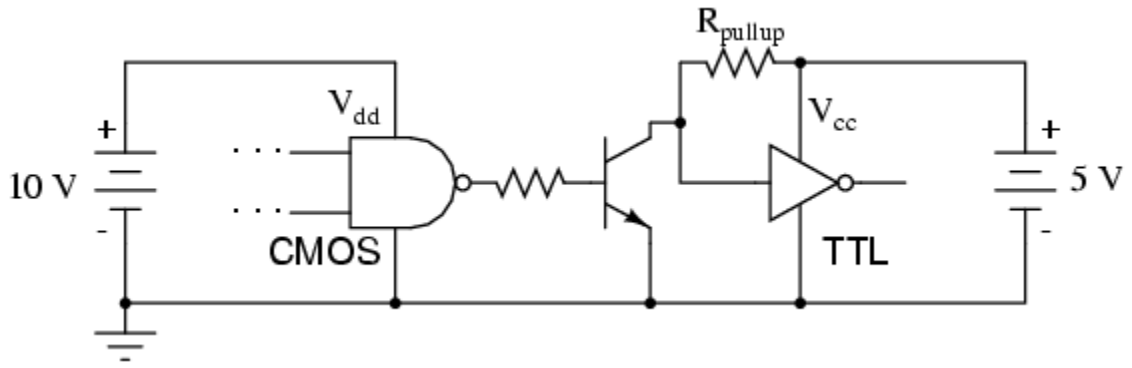


There will be no problem with the CMOS gate interpreting the TTL gate's "low" output, of course, but a "high" signal from the TTL gate is another matter entirely. The guaranteed output voltage range of 2.7 volts to 5 volts from the TTL gate output is nowhere near the CMOS gate's acceptable range of 7 volts to 10 volts for a "high" signal. If we use an *open-collector* TTL gate instead of a totem-pole output gate, though, a pullup resistor to the 10 volt V_{dd} supply rail will raise the TTL gate's "high" output voltage to the full power supply voltage supplying the CMOS gate. Since an open-collector gate can only sink current, not source current, the "high" state voltage level is entirely determined by the power supply to which the pullup resistor is attached, thus neatly solving the mismatch problem:



Due to the excellent output voltage characteristics of CMOS gates, there is typically no problem connecting a CMOS output to a TTL input. The only significant issue is the current loading presented by the TTL inputs, since the CMOS output must sink current for each of the TTL inputs while in the "low" state.

When the CMOS gate in question is powered by a voltage source in excess of 5 volts (V_{cc}), though, a problem will result. The "high" output state of the CMOS gate, being greater than 5 volts, will exceed the TTL gate's acceptable input limits for a "high" signal. A solution to this problem is to create an "open-collector" inverter circuit using a discrete NPN transistor, and use it to interface the two gates together:



The "R_{pullup}" resistor is optional, since TTL inputs automatically assume a "high" state when left floating, which is what will happen when the CMOS gate output is "low" and the transistor cuts off. Of course, one very important consequence of implementing this solution is the logical inversion created by the transistor: when the CMOS gate outputs a "low" signal, the TTL gate sees a "high" input; and when the CMOS gate outputs a "high" signal, the transistor saturates and the TTL gate sees a "low" input. So long as this inversion is accounted for in the logical scheme of the system, all will be well.