0

# B.Sc. Sem.- 4

MAJOR

# PHYSICS

## MAJOR - PHY- 401A

SC23MJDSCPHY401A

## UNIT- IV

# DIGITAL ELECTRONICS

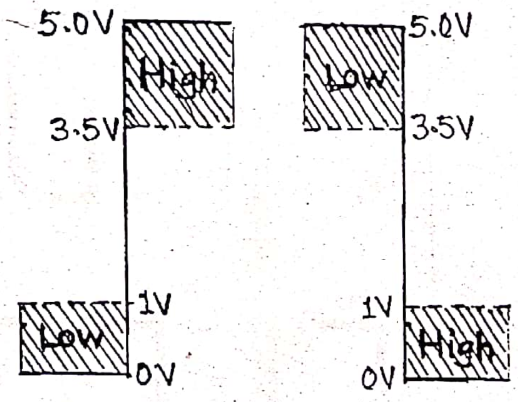
PROF. K C MEVADA

*** INTRODUCTION :-** પૂર્ણ અંકો કે તેના સમૂહોનો ઉપયોગ કરીને કોઈ ક્રિયા પૂર્ણ કરવાની પદ્ધતિને Digital (ડિજિટલ) સાથે સાંકળી શકાય. જ્યારે બીજી બાજુ વોલ્ટેજ, પરિભ્રમણો અને અંતરો જેવી પ્રત્યક્ષ મપાતી ભૌતિકરાશિઓ માટે Analog અંકો દ્વારા રજૂઆત થાય છે. સંદેશાવ્યવહાર (Communication) માટે ડિજિટલ અને એનાલોગ બંને પદ્ધતિઓ વપરાય છે.

ઈલેક્ટ્રોનિક વિવર્ધકો (Amplifiers) કે જેમાં ઈલેક્ટ્રિક સિગ્નલો સતત વિવર્ધિત થાય છે. આ સિગ્નલોને એનાલોગ સિગ્નલ કહે છે. ઈલેક્ટ્રોનિક ગણતરીઓમાં, સ્વિચની મદદથી ઈનપુટ અપાય છે, જે ઈલેક્ટ્રિક સિગ્નલમાં પરિવર્તિત થાય છે. જેને બે જુદી-જુદી કિંમતો અથવા સ્થિતિઓ હોય છે, જેને LOW લેવલ અને HIGH લેવલ કહે છે. તેની ચોક્કસ કિંમત અગત્યની નથી, પરંતુ તે LOW અને HIGH લેવલની ચોક્કસ અવધિમાં રહે છે. આ પ્રકારના સિગ્નલોને ડિજિટલ સિગ્નલ કહે છે અને આ સિગ્નલોનો ઉપયોગ કરીને તૈયાર કરેલ સરકિટ્સને ડિજિટલ સરકિટ કહે છે.

*** DIGITAL SIGNALS :-**

આકૃતિ (a) અને (b) માં બે જુદી-જુદી સ્થિતિઓ કે જેને LOW અને HIGH લેવલ કહેવાય છે તેવા ડિજિટલ સિગ્નલના LOW અને HIGH લેવલ દર્શાવ્યા છે. અહીં આકૃતિ (a) ને ધન લોજિક (Positive Logic) કહે છે જે ધન લોજિક પદ્ધતિમાં ઉપયોગી છે. આકૃતિ (b) ને ઋણ લોજિક (Negative Logic) કહે છે જે ઋણ લોજિક પદ્ધતિમાં ઉપયોગી છે.




(a) Positive logic   (b) Negative logic

અહીં 5V નો વોલ્ટેજ એ લેવલ નથી, પરંતુ 3.5 V થી 5V ની અવધિને આ લેવલ ગણવામાં આવે છે જ્યાં સુધી આપેલ વોલ્ટેજ આ અવધિમાં હોય તેને આ લેવલ તરીકે ગણવામાં

આવે છે અને આ વોલ્ટેજનું મૂલ્ય ચોક્કસ કેટલું છે તે અગત્યનું નથી.

અહીં એ નોંધો કે બધા પરિપથો માટે LOW અને HIGH લેવલ માટેની વોલ્ટેજની અવધિ સરખી હોતી નથી. જુદા-જુદા પરિપથો માટે આ અવધિ પણ જુદી-જુદી હોય છે.

HIGH અને LOW બે જુદી-જુદી સિગ્નલની સ્થિતિઓને Binary Digits (દ્વિઅંકો) 1 અને 0 વડે પણ દર્શાવાય છે. ડિજિટલ સિગ્નલમાં 1 અથવા 0 વાળી બે શક્ય સ્થિતિઓમાંથી કોઈ એક સ્થિતિ હોવાથી ડિજિટલ પદ્ધતિઓના વિશ્લેષણ અને રચના માટે દ્વિઅંકી સંખ્યા પદ્ધતિ વપરાય છે. બે લેવલ્સ (સ્થિતિઓ) ને કેટલીક વાર ON અને OFF અથવા TRUE અને FALSE તરીકે પણ રજૂ કરાય છે.

## * ડિજિટલ ઇલેક્ટ્રોનિક્સમાં વપરાતી સંખ્યા પદ્ધતિઓ :-

⇒ જુદા-જુદા અંક સાથે સંકળાયેલ પદ્ધતિને ડિજિટલ પદ સાથે સાંકળી શકાતી હોવાથી ચાર પ્રકારની અંકગણિત પદ્ધતિઓ વર્ણવી શકાય:

(i) દશાંકી (Decimal) (ii) દ્વિઅંકી (Binary)
(iii) સોળાંકી (Hexadecimal) (iv) અષ્ટાંકી (Octal)

⇒ ડિજિટલ પદ્ધતિમાં મુખ્યત્વે દ્વિઅંકી પદ્ધતિ જ વપરાય છે જેના વિશે પણ આપણે જાણીશું.

## ★ દશાંકી સંખ્યા પદ્ધતિ :-

⇒ આપણા રોજિંદા જીવનમાં કાયમી વપરાતી આ સંખ્યા પદ્ધતિથી આપણે સૌ સુપરિચિત છીએ.

⇒ આ સંખ્યા પદ્ધતિ દશ આગવા સંકેતો ધરાવે છે જેમ કે:
0, 1, 2, 3, 4, 5, 6, 7, 8 અને 9.

⇒ આ સંખ્યા પદ્ધતિના આધારમાં 10 ને દર્શાવાતા હોવાથી દશાંકી સંખ્યા પદ્ધતિ કહેવાય છે.

⇒ અહીં 9 થી મોટી સંખ્યાને એકમના અંકની ડાબી બાજુ ગોઠવીને રજૂ કરાય છે આવા દરેક અંકની સ્થાનકિંમત અથવા

ગુણક અવયવ જુદા-જુદા હોય છે જે એકમ, દશક, સો, હજાર, દશ હજાર વિગેરે જેવાં નામે ઓળખાય છે.

⇒ આ જ પ્રમાણે 1 થી નાની સંખ્યાને દશાંશ ચિહ્નની જમણીબાજુ એ ગોઠવવામાં આવે છે જે દશાંશ (tenths), શતાંશ (hundredths), સહસ્ત્રાંશ (thousandths) જેવાં નામે ઓળખાય છે.

⇒ ઉદાહરણ તરીકે સંખ્યા:

$$432.45_{10} = 4\times10^{2} + 3\times10^{1} + 2\times10^{0} + 4\times10^{-1} + 5\times10^{-2}$$

⇒ આમ, દશાંક પધ્ધતિમાં એકમના સ્થાનનું weight $10^0$, દશકના સ્થાનનું weight $10^1$ અને સોના સ્થાનનું weight $10^2$ છે. તે જ રીતે દશાંશ ચિહ્ન પછીના જમણી બાજુના અંકોનું weight અનુક્રમે $10^{-1}$, $10^{-2}$ વિગેરે છે.

## ★ દ્વિઅંકી સંખ્યા પધ્ધતિ :- (BINARY NUMBER SYSTEM)

⇒ દશાંક પધ્ધતિના દશ અંકોથી વિપરીત એવી દ્વિઅંકી સંખ્યા પધ્ધતિમાં માત્ર બે જ અંકો 0 અને 1 વપરાય છે.

⇒ આ પધ્ધતિ માટે આધારમાં 2 છે, અને દ્વિઅંકી દશાંશ ચિહ્નની ડાબી કે જમણી બાજુના સ્થાનનું weight 2 ની ઘાતમાં અનુક્રમે વધતું કે ઘટતું જોવાં મળે છે.

⇒ ઉદાહરણ તરીકે નીચે દર્શાવેલ દ્વિઅંકી સંખ્યા માટે જમણીબાજુથી ડાબીબાજુ તરફ જતાં weights અનુક્રમે $2^0, 2^1, 2^2, 2^3, 2^4, 2^5$ થશે.

$$\begin{aligned} 101011_2 &= (1\times2^5)+(0\times2^4)+(1\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0) \\ &= 32+0+8+0+2+1 \\ &= 43_{10} \text{ (દશાંક પધ્ધતિમાં)} \end{aligned}$$

આમ, $101011_2 = 43_{10}$

⇒ આ જ પ્રમાણે દશાંશ ચિહ્નવાળી દ્વિઅંકી સંખ્યા માટે...

$$\begin{aligned} 0.1101_2 &= (1\times2^{-1}) + (1\times2^{-2}) + (0\times2^{-3}) + (1\times2^{-4}) \\ &= \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16} \\ &= 0.5000 + 0.2500 + 0 + 0.0625 \end{aligned}$$

$= 0.8125_{10}$ (દશાંક પધ્ધતિમાં)

આ પરથી, $101011.1101_2 = 43.8125_{10}$

અહીં એ નોંધો કે દ્વિઅંકી પ્રક્રિયામાં બે જ શક્ય સ્થિતિઓ હોય. દા.ત. તર્કશાસ્ત્રમાં કોઈ વિધાન true અથવા false હોય, સ્વિચ માટે open અથવા closed સ્થિતિ, તેજ રીતે ઇલેક્ટ્રોનિક્સમાં ટ્રાન્ઝિસ્ટરની બે સ્થિતિ cutoff અથવા saturation લઈ શકાય. ઉપરોક્ત તમામ સ્થિતિઓને દ્વિઅંકી અંકગણિતમાં 1 અને 0 સંખ્યાઓમાં વર્ણવી શકાય.

- **દ્વિઅંકીનું દશાંકીમાં રૂપાંતર :-** ઉપરોક્ત ઉદાહરણમાં દર્શાવ્યા મુજબ આ પ્રક્રિયા ખૂબ જ સરળ છે. આ માટે દ્વિઅંકી સંખ્યાના દરેક 1 ને તેના 2 ની યોગ્ય ઘાતાંકવાળા weight સાથે ગુણી બધાનો સરવાળો કરી દશાંકી સંખ્યા મેળવી શકાય. ઉદાહરણ તરીકે $10111_2$ ને દશાંકીમાં દર્શાવવા...

$$10111_2 = (1\times2^4) + (0\times2^3) + (1\times2^2) + (1\times2^1) + (1\times2^0)$$
$$= 16+0+4+2+1$$
$$= 23_{10}$$

⇒ દશાંશ ચિહ્નવાળી દ્વિઅંકી સંખ્યા માટે દશાંશચિહ્નની જમણી બાજુના અંકને ક્રમશઃ અનુક્રમે $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}$ વડે ગુણી સરવાળો કરી સમતુલ્ય દશાંકી સંખ્યા મેળવી શકાય, દા.ત.

$$0.101_2 = (1\times2^{-1}) + (0\times2^{-2}) + (1\times2^{-3})$$
$$= 0.500 + 0 + 0.125$$
$$= 0.625_{10}$$

⇒ આ જ પ્રમાણે, મિશ્રિત સંખ્યા માટે દ્વિઅંકી દશાંશચિહ્નની ડાબી બાજુના અંકોનું ક્રમશઃ weight $2^0, 2^1, 2^2, 2^3$... વિગેરે લો અને દશાંશ ચિહ્નની જમણી બાજુના અંકોનું ક્રમશઃ weight $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}$,... વિગેરે લો અને ઉપરોક્ત બંને ઉદાહરણો મુજબ સમતુલ્ય દશાંકી સંખ્યા મેળવો. દા.ત.

$$1101.01_2 = (1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)+(0\times2^{-1})+(1\times2^{-2})$$
$$= 8+4+0+1+0+0.25$$
$$= 13.25_{10}$$

• **દશાંકીનું દ્વિઅંકીમાં રૂપાંતર:-** આ માટે સંખ્યાને સતત 2 વડે ભાગવામાં આવે છે અને તેની શેષને ઉલટા ક્રમમાં દર્શાવો. આ રીતને Double-Dabble Method કહે છે.

⇒ આ ડબલ-ડેબલ રીતમાં પ્રથમ આપેલ દશાંક સંખ્યાને 2 વડે ભાગીએ છીએ આમ કરવાથી આપણને ભાગફળ અને શેષ મળે છે. આ શેષને આપણે નોંધી લઈએ છીએ અને પછી જે ભાગફળ આવે તેને ફરીવાર 2 વડે ભાગીએ છીએ અને ફરીવાર તેનું ભાગફળ અને શેષ મેળવીએ છીએ આ રીતે મળેલ શેષને પણ નોંધી લઈએ છીએ.

⇒ આ પ્રક્રિયા જ્યાં સુધી ભાગફળ શૂન્ય ન આવે ત્યાં સુધી ચાલુ રાખવામાં આવે છે. અહીં છેલ્લી શેષ દ્વિઅંકીનો MSB (Most Significant Bit) અને પ્રથમ શેષ દ્વિઅંકીનો LSB (Least Significant Bit) દર્શાવે તે રીતે તમામ નોંધેલી શેષને ઉલટાક્રમમાં (નીચેથી ઉપર) લખવાથી સમતુલ્ય દ્વિઅંકી સંખ્યા મળશે. ઉદાહરણ તરીકે $43_{10} = (?)_2$ મેળવવા...

| | | શેષ | |
|---|---|---|---|
| 2) | 43 | | |
| 2) | 21 | 1 | LSB |
| 2) | 10 | 1 | |
| 2) | 5 | 0 | |
| 2) | 2 | 1 | |
| 2) | 1 | 0 | |
| | 0 | 1 | MSB |

⇒ નોંધેલ દરેક શેષને નીચેથી ઉપર તરફ ક્રમમાં દર્શાવતાં...

$$(43)_{10} = (101011)_2$$

⇒ દશાંક અપૂર્ણાંક માટે દશાંક અપૂર્ણાંકને 2 વડે ગુણીએ છીએ અને મળતા ગુણાકારમાં પૂર્ણાંક ભાગને દૂર કરી અલગ નોંધી લઈએ છીએ. હવે આ ગુણાકારના અપૂર્ણાંક ભાગને ફરી 2 વડે ગુણી મળતા ગુણાકારમાં ફરી પૂર્ણાંક ભાગને દૂર કરી અલગ નોંધી અપૂર્ણાંક ભાગને 2 વડે ગુણીએ છીએ. આ પ્રક્રિયાનું પુનરાવર્તન જેટલા bit ની દ્વિઅંકી સંખ્યા જોઈતી હોય ત્યાં સુધી અથવા અપૂર્ણાંક ભાગ શૂન્ય ન બને ત્યાં સુધી ચાલુ રાખીએ છીએ. ઉદાહરણ તરીકે $0.4375_{10}$ માટે...

| | |
|---|---|
| $0.4375 \times 2 = 0.8750$ | પૂર્ણાંક ભાગ 0 |
| $0.8750 \times 2 = 1.7500$ | પૂર્ણાંક ભાગ 1 |
| $0.7500 \times 2 = 1.5000$ | પૂર્ણાંક ભાગ 1 |
| $0.5000 \times 2 = 1.0000$ | પૂર્ણાંક ભાગ 1 |

નોંધેલ દરેક પૂર્ણાંક ભાગને ઉપરથી નીચે તરફ ક્રમમાં દર્શાવો.

$\therefore 0.4375_{10} = 0.0111_2$

**Ex. 1:** $200_{10}$ ને દ્વિઅંકીમાં રૂપાંતર કરો :

| | | શેષ |
|---|---|---|
| 2) | 200 | |
| 2) | 100 | 0 |
| 2) | 50 | 0 |
| 2) | 25 | 0 |
| 2) | 12 | 1 |
| 2) | 6 | 0 |
| 2) | 3 | 0 |
| 2) | 1 | 1 |
| | 0 | 1 |

$\therefore 200_{10}$

$= 11001000_2$

**Ex. 2:** 59.4375 ને દ્વિઅંકીમાં રૂપાંતર કરો :

59 માટે

| | | શેષ |
|---|---|---|
| 2) | 59 | |
| 2) | 29 | 1 |
| 2) | 14 | 1 |
| 2) | 7 | 0 |
| 2) | 3 | 1 |
| 2) | 1 | 1 |
| | 0 | 1 |

0.4375 માટે

| | પૂર્ણાંક ભાગ |
|---|---|
| $0.4375 \times 2 = 0.8750$ | 0 |
| $0.8750 \times 2 = 1.7500$ | 1 |
| $0.7500 \times 2 = 1.5000$ | 1 |
| $0.5000 \times 2 = 1.0000$ | 1 |

તેથી $59.4375_{10} = 111011.0111_2$

**Home Work:** નીચેની સંખ્યાઓને દ્વિઅંકી સંખ્યામાં રૂપાંતર કરો

1. $0.95_{10}$ (8 bit સુધી)
2. $47_{10}$
3. $300_{10}$
4. $57.4375_{10}$
5. $25.7_{10}$ (5 bit સુધી)

Ans. ① $0.11110011_2$
" ② $101111_2$
" ③ $100101100_2$
" ④ $111001.0111_2$
" ⑤ $11001.10110_2$

PROF. KALPESH C. MEVADA

## -: Hexadecimal Number System :-

### ★ સોળાંકી સંખ્યા પદ્ધતિ :-

⇒ આ સંખ્યા પદ્ધતિ દ્વિઅંકી સંખ્યાને સંક્ષિપ્તમાં રજૂ કરે છે અને તે સામાન્ય રીતે કમ્પ્યૂટર્સમાં વપરાય છે.

⇒ સોળાંકી સંખ્યાને દ્વિઅંકી દશાંશ સ્થળથી શરૂ કરીને દ્વિઅંકી સંખ્યાના bits ને 4 bits ના સમૂહ (group) માં દર્શાવી રજૂ કરવામાં આવે છે.

⇒ ઉદાહરણ તરીકે દ્વિઅંકી સંખ્યા 1010111010 ને 0010 1011 1010 ના ગ્રુપમાં દર્શાવી શકાય.

⇒ આ સંખ્યા પદ્ધતિ માઇક્રોપ્રોસેસરના કાર્યમાં પણ ઉપયોગી છે.

⇒ આ સંખ્યા પદ્ધતિના આધાર (base) માં 16 છે અને તેથી સોળાંકી દશાંશચિહ્નની ડાબી બાજુના સ્થાનનું weight 16 ની ઘાતમાં ક્રમશઃ વધતું ($16^0$, $16^1$, $16^2$, $16^3$ વિગેરે) જોવાં મળે છે.

⇒ નીચેના ટેબલમાં દશાંકી સંખ્યાઓ 0 થી 15 ને 4 bits ના ગ્રુપમાં સોળાંકી અને અષ્ટાંકી સંખ્યા પદ્ધતિમાં દર્શાવેલ છે.

ટેબલ : સંખ્યા પદ્ધતિઓ

| દશાંકી | દ્વિઅંકી (4 bits) | સોળાંકી | અષ્ટાંકી |
|---|---|---|---|
| 0 | 0000 | 0 | 0 |
| 1 | 0001 | 1 | 1 |
| 2 | 0010 | 2 | 2 |
| 3 | 0011 | 3 | 3 |
| 4 | 0100 | 4 | 4 |
| 5 | 0101 | 5 | 5 |
| 6 | 0110 | 6 | 6 |
| 7 | 0111 | 7 | 7 |
| 8 | 1000 | 8 | 10 |
| 9 | 1001 | 9 | 11 |
| 10 | 1010 | A | 12 |
| 11 | 1011 | B | 13 |
| 12 | 1100 | C | 14 |
| 13 | 1101 | D | 15 |
| 14 | 1110 | E | 16 |
| 15 | 1111 | F | 17 |
| 16 | 0001 0000 | 10 | 20 |

⇒ આમ, સોળાંકી સંખ્યા પદ્ધતિમાં સોળ સંકેતો જેવા કે, 0,1,2,3,4, 5,6,7,8,9,A,B,C,D,E અને F નો ઉપયોગ થાય છે.

⇒ આગળ દર્શાવેલ ટેબલ પરથી,

37AE એટલે 0011 0111 1010 1110 થશે અને

તેથી 2BA એટલે 0010 1011 1010.

***Ex.1:-** $FACE_{16}$ ને દ્વિઅંકીમાં રૂપાંતર કરો.

| | | | | |
|---|---|---|---|---|
| અહીં | F | A | C | E |
| દરેક અંક માટે 4bit દ્વિઅંકી | 1111 | 1010 | 1100 | 1110 |

તેથી દ્વિઅંકી સંખ્યા $1111\ 1010\ 1100\ 1110_2$

***Ex.2:-** $AB2_{16}$ ને દ્વિઅંકીમાં રૂપાંતર કરો.

| | | | |
|---|---|---|---|
| અહીં | A | B | 2 |
| દ્વિઅંકી(4bit) | 1010 | 1011 | 0010 |

તેથી $AB2_{16} = 1010\ 1011\ 0010_2$

***Ex.3:-** $11010010_2$ ને સોળાંકીમાં રૂપાંતર કરો.

| | | |
|---|---|---|
| 4bitsના ગ્રુપમાં | 1101 | 0010 |
| અનુરૂપ કોડ(સંખ્યા) સોળાંકી | D | 2 |

તેથી, $11010010_2 = D2_{16}$

***Ex.4:-** $ED2_{16}$ ને દશાંકીમાં રૂપાંતર કરો

⇒ આપણે જાણીએ છીએ કે સોળાંકી સંખ્યાના આધારમાં 16 છે. તેમજ E ને અનુરૂપ દશાંકી સંખ્યા 14 અને D ને અનુરૂપ દશાંકી સંખ્યા 13 છે. તેથી

| | E | D | 2 |
|---|---|---|---|
| weight | $16^2$ | $16^1$ | $16^0$ |

$$ED2_{16} = (14\times16^2) + (13\times16^1) + (2\times16^0)$$
$$= (14\times256) + (13\times16) + (2\times1)$$
$$= 3584 + 208 + 2$$
$$= 3794_{10}$$

***Ex.5:-** $ABCD_{16}$ ને દશાંકીમાં રૂપાંતર કરો

$$ABCD_{16} = (10\times16^3) + (11\times16^2) + (12\times16^1) + (13\times16^0)$$
$$= 40960 + 2816 + 192 + 13$$
$$= 43981_{10}$$

દશાંકીમાં
A → 10
B → 11
C → 12
D → 13

• **દશાંકીનું સોળાંકીમાં રૂપાંતર :-** આ માટે ડબલ-ડેબલ રીતની જેમ અહીં સતત 16 વડે ભાગાકાર કરી શેષ નોંધી, નોંધેલ શેષને ઉલટા ક્રમમાં દર્શાવો. (અહીં દરેક શેષને અનુરૂપ સોળાંકી સંખ્યામાં જ દર્શાવવી)

✱ **Ex. 6 :-** $72905_{10}$ ને સોળાંકીમાં રૂપાંતર કરો.

| | | શેષ | સોળાંકી સંખ્યા |
|---|---|---|---|
| 16 ) | 72905 | | |
| 16 ) | 4556 | 9 | 9 |
| 16 ) | 284 | 12 | C |
| 16 ) | 17 | 12 | C |
| 16 ) | 1 | 1 | 1 |
| | 0 | 1 | 1 |

$\therefore 72905_{10} = 11CC9_{16}$

▫ **HomeWork:**

① $2585_{10}$ ને સોળાંકીમાં રૂપાંતર કરો. Ans. $A19_{16}$
② $43981_{10}$ ને સોળાંકીમાં રૂપાંતર કરો. " $ABCD_{16}$
③ $8BF_{16}$ ને દ્વિઅંકીમાં દર્શાવો. " $100010111111_2$
④ $101110101101_2$ ને સોળાંકીમાં દર્શાવો. " $BAD_{16}$

## -: The Octal Number System :-

## ★ અષ્ટાંકી સંખ્યા પદ્ધતિ :-

⇒ અષ્ટાંકી સંખ્યા પદ્ધતિમાં આઠ અંકો 0, 1, 2, 3, 4, 5, 6 અને 7 હોય છે.
⇒ અષ્ટાંકી સંખ્યાને દ્વિઅંકી દશાંશ સ્થળથી શરૂ કરીને દ્વિઅંકી સંખ્યાના bits ને 3 bits ના સમૂહોમાં દર્શાવી રજૂ કરવામાં આવે છે.
⇒ અષ્ટાંકી સંખ્યા પદ્ધતિનો મહત્તમ અંક 7 છે. આ સંખ્યા પદ્ધતિમાં 8 અને 9 આવતાં નથી.
⇒ આ સંખ્યા પદ્ધતિના આધારમાં 8 છે અને તેથી અષ્ટાંકી દશાંશચિહ્ન ની ડાબીબાજુના સ્થાનનું weight 8 ની ઘાતમાં ક્રમશઃ વધતું ($8^0$, $8^1$, $8^2$, $8^3$ વિગેરે) જોવાં મળે છે.
⇒ આપેલ ટેબલમાં દશાંકી સંખ્યાઓ 0 થી 7 ને 3 bits ના ગ્રુપમાં અષ્ટાંકી સંખ્યા પદ્ધતિમાં દર્શાવેલ છે :

ટેબલ: સંખ્યા પદ્ધતિઓ

| દશાંકી | દ્વિઅંકી(3bits) | અષ્ટાંકી |
|---|---|---|
| 0 | 000 | 0 |
| 1 | 001 | 1 |
| 2 | 010 | 2 |
| 3 | 011 | 3 |
| 4 | 100 | 4 |
| 5 | 101 | 5 |
| 6 | 110 | 6 |
| 7 | 111 | 7 |
| 8 | 001 000 | 10 |

⇒ વધુ સમજવા નીચેના ઉદાહરણો લેતાં...

*** Ex.1:-** $11111011110101_2$ ને અષ્ટાંકીમાં રૂપાંતર કરો.

ત્રણ-ત્રણના સમૂહોમાં વહેંચતા 011 111 011 110 101

દરેક સમૂહને અષ્ટાંકીમાં લખતાં 3 7 3 6 5

તેથી $11111011110101_2 = 37365_8$

*** Ex.2:-** $7423_8$ ને દ્વિઅંકીમાં રૂપાંતર કરો.

આપેલ અષ્ટાંકી સંખ્યા 7 4 2 3

અનુરૂપ દ્વિ અંકી (3bits) 111 100 010 011

તેથી $7423_8 = 111100010011_2$

- દશાંકીનું અષ્ટાંકીમાં રૂપાંતર કરવા સતત 8 વડે ભાગાકાર કરી શેષ નોંધો અને નોંધેલ શેષને ઉલટાક્રમમાં દર્શાવો.

*** Ex.3:-** $56_{10}$ ને અષ્ટાંકીમાં રૂપાંતર કરો.

| | | શેષ |
|---|---|---|
| 8 | 56 | |
| 8 | 7 | 0 |
| | 0 | 7 |

તેથી $56_{10} = 70_8$

*** Ex.4:-** $43981_{10} = ABCD_{16}$ ને અષ્ટાંકીમાં રૂપાંતર કરો.

| | | શેષ |
|---|---|---|
| 8 ) | 43981 | |
| 8 ) | 5497 | 5 |
| 8 ) | 687 | 1 |
| 8 ) | 85 | 7 |
| 8 ) | 10 | 5 |
| 8 ) | 1 | 2 |
| | 0 | 1 |

તેથી

$43981_{10} = ABCD_{16}$

$= 125715_8$

**યાદ રાખો :**

bit : દ્વિઅંકી સંખ્યા પદ્ધતિમાંના અંક (0 કે 1) ને bit (**bi**nary digi**t**) કહે છે.

nibble : ચાર bits વાળી દ્વિઅંકી સંખ્યાને nibble કહેવાય છે.

byte : આઠ bit વાળી દ્વિઅંકી સંખ્યાને byte કહે છે.

આમ, 1 byte = 8 bit = 2 nibble

1 nibble = 4 bit

bit : X

nibble : XXXX

byte : XXXXXXXX

જ્યાં X = 0 અથવા 1

MSB (Most Significant Bit) : દ્વિઅંકી સંખ્યામાં મહત્તમ weight દર્શાવતો bit MSB કહેવાય છે.

LSB (Least Significant Bit) : દ્વિઅંકી સંખ્યામાં ન્યૂનતમ weight દર્શાવતો bit LSB કહેવાય છે.

દા.ત. 1011010

MSB LSB

## * BINARY CODES (દ્વિઅંકી કોડ):-

કોડ એ સામાન્ય રીતે સંજ્ઞાઓની પદ્ધતિ (System of Symbols) છે જેના દ્વારા ડિજિટલ પરિપથમાંથી અર્થપૂર્ણ કમ્યુનિકેશન મેળવવામાં આવે છે.

⇒ કમ્પ્યૂટર અને બીજા (અન્ય) ડિજિટલ પરિપથોમાં સંખ્યાઓ, મૂળાક્ષરો અથવા વિશિષ્ટ સંકેતો જેવી માહિતી (data) સંભાળવી જરૂરી છે.

⇒ પરંતુ ડિજિટલ પરિપથો દ્વિઅંકી પ્રણાલીમાં જ કાર્યરત હોવાથી, આપણે આવી સંખ્યાઓ કે મૂળાક્ષરોનું દ્વિઅંકીમાં (Binary format) રૂપાંતર કરવું પડે.

⇒ આ માટે ઘણી બધી જુદી-જુદી પદ્ધતિઓ છે અને આવી પદ્ધતિઓ ને encoding કહે છે.

⇒ આવાં જુદાં-જુદાં codes અસ્તિત્વમાં છે અને તે બધા જુદાં-જુદાં હેતુઓ માટે કાર્યરત છે.

⇒ Codes નો ઉપયોગ ભૂલો શોધવા અને સુધારવા માટે પણ ડિજિટલ પરિપથોમાં થાય છે.

⇒ જેમાંના કેટલાક સામાન્ય રીતે વપરાતા codes ની અહીં ચર્ચા કરીશું

## A BCD અથવા 8421 Code:-

⇒ સૌથી વધુ વપરાતી કોડીંગ પદ્ધતિ straight binary code છે, જેમાં બધી સંખ્યાઓને દ્વિઅંકી સ્વરૂપમાં દર્શાવાય છે.

⇒ જેમ કે 1 ને 001 વડે, 7 ને 0111 વડે, 105 ને 1101001 વડે દર્શાવાય.

⇒ આ પદ્ધતિનો ગેરફાયદો એ છે કે તેમાં વધારે અંકોનો ઉપયોગ થતો હોઈ હાર્ડવેર દ્વારા આ અંકોને રજૂ કરવું થોડું મુશ્કેલ થઈ પડે છે.

⇒ ઉપરોક્ત મુશ્કેલીનું નિવારણ બીજી કોડીંગ પદ્ધતિ દ્વારા કરી શકાય છે આ કોડીંગ પદ્ધતિને binary coded decimal અથવા ટૂંકમાં BCD કહે છે

⇒ આ પદ્ધતિમાં દરેક સ્વતંત્ર દશાંક અંકોને દ્વિઅંકી સ્વરૂપમાં કોડ અપાય છે અને તે સ્વતંત્ર રૂપે કાર્યરત રહે છે.

⇒ આમ, 0 થી 9 સુધીના દશાંક અંકોના binary codes ને આ પદ્ધતિમાં દર્શાવવામાં આવે છે.

⇒ ઉદાહરણ તરીકે 342 ને BCD માં દર્શાવતા...

0011 0100 0010
(3) (4) (2)

⇒ તેથી કોઈપણ દશાંકી અંકને રજૂ કરવા માટે 8 bit ની પદ્ધતિ પૂરતી છે. હાર્ડવેરના માત્ર 4 ઘટકો જ તેને operate (કાર્યરત) કરવા જરૂરી છે, વળી દરેક નંબર માટે તેને ઘણીબધી વાર વાપરી શકાય છે.

⇒ તેથી ચાર Flip-Flop સાથેના એક અવરોધવાળા ચાર તબક્કાનો પરિપથ કોઈપણ દશાંકી અંકને રજૂ કરવા માટે પૂરતો છે.

⇒ આ કોડને 8-4-2-1 code પણ કહે છે.

Table : BCD Code

| Decimal | 8421 | | Binary |
|---|---|---|---|
| 0 | | 0000 | 0000 |
| 1 | | 0001 | 0001 |
| 2 | | 0010 | 0010 |
| 3 | | 0011 | 0011 |
| 4 | | 0100 | 0100 |
| 5 | | 0101 | 0101 |
| 6 | | 0110 | 0110 |
| 7 | | 0111 | 0111 |
| 8 | | 1000 | 1000 |
| 9 | | 1001 | 1001 |
| 10 | 0001 | 0000 | 1010 |
| 11 | 0001 | 0001 | 1011 |
| ... | ... | ... | ... |
| 15 | 0001 | 0101 | 1111 |
| ... | ... | ... | ... |

⇒ ઉપરોક્ત ટેબલમાં દર્શાવ્યા મુજબ 8421 કોડ માટે મોટામાંમોટો 4 bit સમૂહ 1001 છે. બીજા શબ્દોમાં 4 bit ના શક્ય એવાં 16 સમૂહોમાંથી માત્ર 10 જ સમૂહો વપરાય છે.

⇒ 8421 કોડમાં દ્વિઅંકી સંખ્યાઓ 1010, 1011, 1100, 1101, 1110 અને 1111 નો ઉપયોગ થતો નથી.

⇒ તેથી BCD અંકો 0000 થી 1001 સુધીના જ 4-bit અંકો સુધી મર્યાદિત છે અને 1010 થી 1111 સુધીની છ અમાન્ય સ્થિતિઓ BCD ની સામાન્ય ક્રિયામાં ક્યારેય આવતી નથી.

⇒ આમ, BCD અથવા 8421 કોડ એ 9 સુધીની દશાંકી સંખ્યાઓ માટે વધુ પ્રચલિત છે

- **Ex.1:-** નીચે દર્શાવેલ દરેક અંકોને BCD માં રૂપાંતર કરો:

(a) 3 (b) 9 (c) 18 (d) 34 (e) 65 (f) 321 (g) 1472

Sol$^n$: (a) 3 → 0011

(b) 9 → 1001

(c) 18 → 0001 1000

(d) 65 → 0110 0101

(e) 321 → 0011 0010 0001

(f) 1472 → 0001 0100 0111 0010

- **Ex.2:-** BCD કોડ્સ દ્વારા રજૂ થતાં નીચેના codes ની દશાંકી સંખ્યા શોધો:

(a) 10000110 (b) 00110001 (c) 100101110100

Sol$^n$: (a) 1000 0110 (∵ 4-bit ના સમૂહોમાં અલગ દર્શાવતાં)

↓ ↓

8 6 (∵ અનુરૂપ દશાંક સંખ્યા દર્શાવતાં)

તેથી 86

(b) 0011 0001 → 3 1

તેથી 31

(c) 1001 0111 0100 → 9 7 4

તેથી 974

## B Gray Code (ગ્રે કોડ):-

⇒ બીજા અન્ય પ્રકારનો ઉપયોગી કોડ એ ગ્રે કોડ છે.

⇒ દ્વિઅંકી પદ્ધતિમાં જ્યારે 7 થી 8 તરફ જઈએ છીએ ત્યારે બધા જ 4 bits બદલાઈ જાય છે (0111 થી 1000 થાય છે), જ્યારે ગ્રે કોડ માં આવાં ફેરફારમાં માત્ર જે તે ક્ષણે એક જ bit બદલાય છે.

⇒ દ્વિઅંકી સંખ્યાને તેના સમતુલ્ય ગ્રે કોડમાં બદલવા નીચે દર્શાવેલ પદ્ધતિને અનુસરો:

(i) સૌપ્રથમ most significant bit (MSB) ને નોંધો.

(ii) હવે આ bit ને તેના આગળના સ્થાનમાં ઉમેરો, જેનો સરવાળો (sum) નોંધો અને વદ્ધીને (carry ને) અવગણો.

(iii) આ પ્રમાણે છેલ્લા સ્થાન (LSB) સુધી પૂર્ણ ન થાય ત્યાં સુધીના સતત સરવાળા નોંધો.

ટેબલ: Gray Code

| Decimal | (Binary) | Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |
| 16 | 10000 | 11000 |

• **Ex.1:- દ્વિઅંકી 1011 ને ગ્રે કોડ માં ફેરવો.**

Soln:

Binary 1 0 1 1

↓ ↓ ↓ ↓

Gray 1 1 1 0

MSB

1 0 1 1 ← દ્વિઅંકી

8 4 2 1 ← સ્થાનનું weightage

પદ્ધતિ:-

① પ્રથમ MSB એટલે કે 8's bit નોંધો (અહીં MSB 1 છે)

② હવે 8's bit ને 4's bit માં ઉમેરો અને સરવાળો નોંધો. (1+0=1)

③ ત્યારબાદ 4's bit ને 2's bit માં ઉમેરો અને સરવાળો નોંધો. (0+1=1)

④ ત્યારબાદ 2's bit ને 1's bit માં ઉમેરો અને carry ને અવગણી સરવાળો જ નોંધો. (1+1=0 with a carry 1)

**• Ex.2:-** દ્વિઅંકી 1001 ને ગ્રે કોડમાં ફેરવો.

**Sol$^n$:** Position Weight 8 4 2 1

Binary ⟶ 1 0 0 1

**પદ્ધતિ:-**

① સૌ પ્રથમ MSB (જે દ્વિઅંકીનો 8's bit થશે) નોંધો. (જે અહીં 1 છે)

MSB
1 0 0 1
1

② હવે 8's bit ને 4's bit માં ઉમેરો અને carry ને અવગણી સરવાળો નોંધો.

1 0 0 1
1 1

③ ત્યારબાદ 4's bit ને 2's bit માં ઉમેરી carry ને અવગણી સરવાળો નોંધો.

1 0 0 1
1 1 0

④ ત્યારબાદ 2's bit ને 1's bit માં ઉમેરી carry ને અવગણી સરવાળો નોંધો.

1 0 0 1
1 1 0 1

Final Result ⟶ 1 1 0 1 (Gray)

⇒ ઉપરોક્ત પ્રક્રિયાને કોઈપણ નંબરના bits માટે લંબાવી શકાય છે.

**• Ex.3:-** ગ્રે કોડ 1011 ને દ્વિઅંકી (binary) માં ફેરવો.

**Sol$^n$:**

| | MSB | | | LSB |
|---|---|---|---|---|
| Gray | 1 | 0 | 1 | 1 |
| Binary | 1 | 1 | 0 | 1 |
| Position Weightage | 8 | 4 | 2 | 1 |

**પદ્ધતિ:-** ① સૌ પ્રથમ MSB ને નોંધો (જે અહીં 1 છે) તે દ્વિઅંકીનો 8's bit થશે.

② હવે ગ્રે કોડ સંખ્યાના 4's bit માં MSB ને ઉમેરો અને carry અવગણી સરવાળો (sum) નોંધો. (અહીં 1+0=**1**) તે દ્વિઅંકીનો 4's bit થશે.

③ ત્યારબાદ ગ્રે કોડના 2's bit માં દ્વિઅંકીના 4's bit ને ઉમેરી carry અવગણી sum નોંધો (અહીં 1+1=**0** with a carry 1) જે દ્વિઅંકીનો 2's bit થશે.

④ છેલ્લે દ્વિઅંકીના 2's bit ને ગ્રે કોડના 1's bit (LSB) માં ઉમેરી carry અવગણી sum નોંધો (અહીં 0+1=**1**) જે દ્વિઅંકીનો 1's bit અથવા LSB થશે.

## C The Excess-3 Code અથવા XS3 :-

⇒ દરેક દશાંકી અંકમાં 3 ઉમેરી અને ત્યારબાદ તેને 4 bit binary (દ્વિઅંકી) માં રૂપાંતરિત કરેલ પરિણામ દ્વારા એક્સેસ-3 કોડ તારવી શકાય છે.

⇒ તેથી આ કોડના ચાર અંકના સ્થાનનું કોઈ ચોક્કસ weight રજૂ કરી શકાય નહીં, Excess-3 કોડ એ unweighted code છે.

⇒ ઉદાહરણ તરીકે દશાંકી સંખ્યાઓ 3, 4 અને 5 માટેનો એક્સેસ-3 કોડ ...

$$
\begin{array}{ccc}
3 & 4 & 5 \\
+3 & +3 & +3 \\
\hline
6 & 7 & 8 \\
\downarrow & \downarrow & \downarrow \\
0110 & 0111 & 1000
\end{array}
$$

← (ત્રણ ઉમેરતાં)

← (4 bit દ્વિઅંકીમાં દર્શાવતાં)

⇒ નીચેના ટેબલમાં એક્સેસ-3 કોડ દર્શાવેલ છે.

ટેબલ: Excess-3 Code

| Decimal | Binary | Excess-3 |
|---|---|---|
| 0 | 0000 | 0011 |
| 1 | 0001 | 0100 |
| 2 | 0010 | 0101 |
| 3 | 0011 | 0110 |
| 4 | 0100 | 0111 |
| 5 | 0101 | 1000 |
| 6 | 0110 | 1001 |
| 7 | 0111 | 1010 |
| 8 | 1000 | 1011 |
| 9 | 1001 | 1100 |

⇒ અહીં એ નોંધો કે શક્ય એવા 16 કોડના સમૂહોમાંથી માત્ર 10 જ કોડ સંયોજન Excess-3 કોડમાં વપરાય છે.

⇒ છ અમાન્ય સંયોજનો 0000, 0001, 0010, 1101, 1110 અને 1111 છે.

⇒ Excess-3 કોડનો સૌથી અગત્યનો ગુણધર્મ એ છે કે તે self complementing property (સ્વ-પૂરક ગુણધર્મ) ધરાવે છે, જેને આપણે ઉદાહરણ દ્વારા સમજીએ :

→ દશાંકી સંખ્યા 4 માટેનો એક્સેસ-3 કોડ 0111 છે

→ આ કોડનો 1's complement 1000 છે.

→ પણ 1000 એ દશાંકી સંખ્યા 5 માટેનો એક્સેસ-3 કોડ છે.

→ અને 5 એ 4 નો 9's complement છે.

⇒ તેથી આપણે પૂર્ણતઃ કહી શકીએ કે "Excess-3 સંખ્યા (અહીં 0111) નો 1's complement (અહીં 1000) એ અનુરૂપ દશાંકી સંખ્યાના 9's complement (અહીં 5) નો Excess-3 કોડ હોય છે."

• **Ex. 1 :-** $(643)_{10}$ ને XS3 કોડમાં રૂપાંતર કરો :

**Sol$^n$ :**

| | | | |
|---|---|---|---|
| Decimal number → | 6 | 4 | 3 |
| Add 3 to each bit | +3 | +3 | +3 |
| Sum → | 9 | 7 | 6 |
| Convert to BCD code → | 1001 | 0111 | 0110 ← XS3 code |

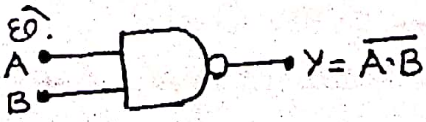
⇒ તેથી $(643)_{10}$ નો XS-3 code 1001 0111 0110 છે.

## ❑ UNIVERSAL GATE (સાર્વત્રિક ગેટ) :-

⇒ AND, OR અને NOT ગેટને મૂળભૂત લોજિક ગેટ કહે છે.

⇒ જ્યારે NAND અને NOR ગેટને સાર્વત્રિક ગેટ (Universal Gate) કહેવામાં આવે છે, કારણકે NAND અને NOR ગેટ AND, OR, NOT જેવી બધી જ તાર્કિક ક્રિયાઓ માટે વાપરી શકાય છે. કોઈપણ બુલિયન સમીકરણને NAND અને NOR ગેટનો ઉપયોગ કરી રજૂ કરી શકાય છે. તેથી Universal Gate કહેવાય છે.

*** NAND gate :** ⇒ AND ગેટના આઉટપુટને NOT ગેટના ઈનપુટ સાથે શ્રેણીબદ્ધ જોડતાં તૈયાર થતા ગેટને (NOTAND) NAND ગેટ કહે છે. જેની પરિપથ સંજ્ઞા અને ટ્રુથ ટેબલ નીચે દર્શાવેલ છે.




| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| Y | 1 | 1 | 1 | 0 |

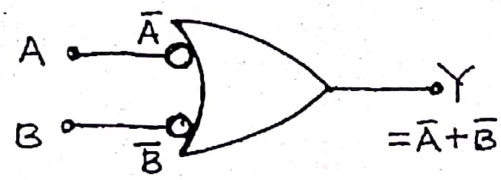
*** Bubbled OR gate :** ⇒ NAND ગેટની બીજી રીતની રજૂઆત bubbled OR gate દ્વારા કરી શકાય. કારણકે NAND ગેટ માટે આઉટપુટ 1 ત્યારે જ મળે કે જ્યારે ઇનપુટ A 0 હોય અથવા(OR) ઇનપુટ B 0 હોય.

⇒ વળી, De-Morgan's Theorem મુજબ $\overline{A\cdot B} = \overline{A}+\overline{B}$

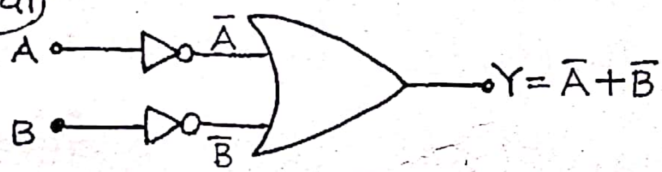
| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| $\overline{A}$ | 1 | 1 | 0 | 0 |
| $\overline{B}$ | 1 | 0 | 1 | 0 |
| $Y=\overline{A}+\overline{B}$ | 1 | 1 | 1 | 0 |

⇒ આ આઉટપુટ NAND ગેટના ટ્રુથ ટેબલના આઉટપુટને મળતું આવે છે તેથી કહી શકાય કે NAND ગેટ અને Bubbled OR ગેટ એકબીજાને સમતુલ્ય છે.

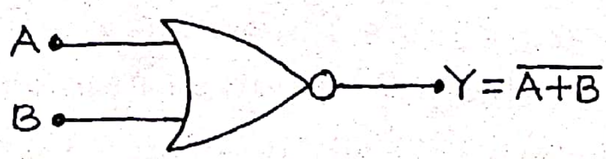
⇒ બાજુની આકૃતિમાં bubbled OR ગેટની પરિપથ સંજ્ઞા દર્શાવેલ છે.




અથવા




*** NOR gate :** ⇒ OR ગેટના આઉટપુટને NOT ગેટના ઇનપુટ સાથે શ્રેણીબદ્ધ જોડતાં તૈયાર થતા ગેટને (NOT OR) NOR ગેટ કહે છે જેની પરિપથ સંજ્ઞા અને ટ્રુથ ટેબલ નીચે દર્શાવેલ છે.




| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 |
| Y | 1 | 0 | 0 | 0 |

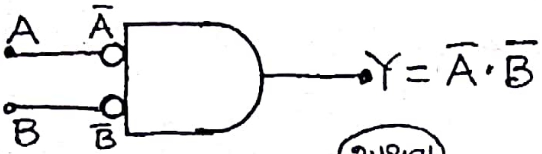
*** Bubbled AND gate :** ⇒ NOR ગેટની બીજી રીતની રજૂઆત bubbled AND gate દ્વારા કરી શકાય. કારણકે NOR ગેટ માટે આઉટપુટ 1 ત્યારે જ મળે કે જ્યારે ઇનપુટ A 0 હોય અને(AND) ઇનપુટ B પણ 0 હોય.

⇒ વળી, De-Morgan's Theorem મુજબ $\overline{A+B} = \overline{A}\cdot\overline{B}$

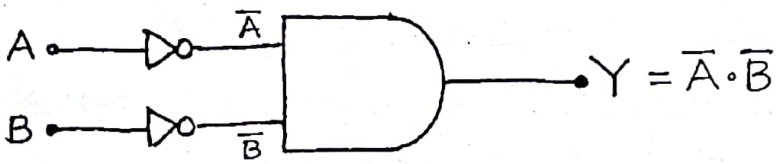
| A | 0 | 0 | 1 | 1 |
|---|---|---|---|---|
| B | 0 | 1 | 0 | 1 |
| $\bar{A}$ | 1 | 1 | 0 | 0 |
| $\bar{B}$ | 1 | 0 | 1 | 0 |
| $Y=\bar{A}\cdot\bar{B}$ | 1 | 0 | 0 | 0 |

⇒ આ આઉટપુટ NOR ગેટના ટ્રુથ ટેબલના આઉટપુટને મળતું આવે છે તેથી કહી શકાય કે NOR ગેટ અને Bubbled AND ગેટ એકબીજાને સમતુલ્ય છે.

⇒ બાજુની આકૃતિમાં bubbled AND ગેટની પરિપથ સંજ્ઞા દર્શાવેલ છે.




(અથવા)



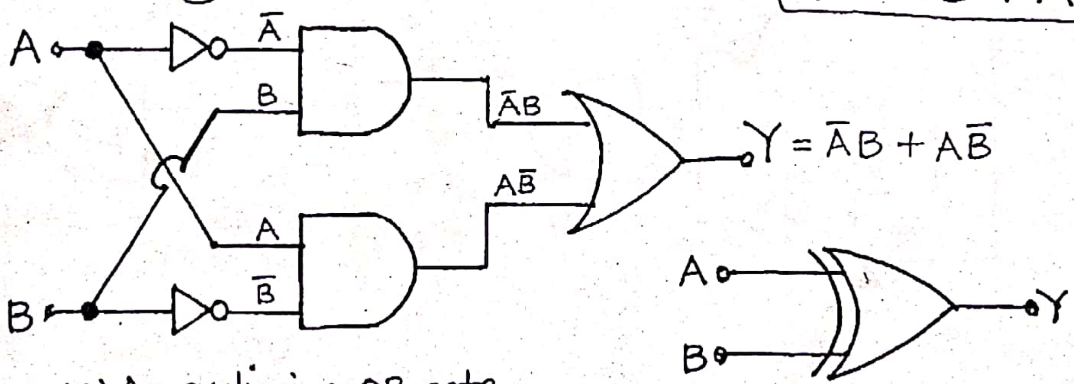

## ★ અંકગણિતીય પરિપથો

### Exclusive OR gate (XOR gate):

⇒ ડિજિટલ કમ્પ્યૂટર્સમાં વપરાતા હાફ અને ફૂલ એડર જેવા કેટલાક મૂળભૂત અંકગણિતીય પરિપથોમાં Exclusive OR ગેટ (XOR gate) જ વપરાય છે. જેને નીચે મુજબ વર્ણવી શકાય:

The XOR gate: નીચેની આકૃતિમાં XOR gate (Exclusive OR ગેટ) ને તેની લોજિક સંજ્ઞા સાથે દર્શાવેલ છે. જેને બે ઇનપુટ્સ અને એક આઉટપુટ હોય છે.

⇒ જો તેના ઇનપુટને A અને B વડે દર્શાવીએ તો પછી XOR ગેટનો આઉટપુટ Y આ મુજબ દર્શાવી શકાય: $\boxed{Y = A\bar{B} + \bar{A}B}$




(a) An exclusive OR gate

(b) Logic symbol of XOR gate

⇒ XOR ગેટના ટ્રુથટેબલની ક્રમબદ્ધ રચના માટે આપણે નીચે દર્શાવ્યા મુજબના ઈનપુટ મૂલ્યો લેતાં...

(i) જ્યારે $A=0$ અને $B=0$ હોય, તેથી $\bar{A}=1$ અને $\bar{B}=1$ થશે.

$\therefore Y = 0\cdot1 + 1\cdot0 = 0+0 = 0$

(ii) જ્યારે $A=0$ અને $B=1$ હોય, તેથી $\bar{A}=1$ અને $\bar{B}=0$ થશે.

$\therefore Y = 0\cdot0 + 1\cdot1 = 0+1 = 1$

(iii) જ્યારે $A=1$ અને $B=0$ હોય, તેથી $\bar{A}=0$ અને $\bar{B}=1$ થશે.

$\therefore Y = 1\cdot1 + 0\cdot0 = 1+0 = 1$

(iv) જ્યારે $A=1$ અને $B=1$ હોય, તેથી $\bar{A}=0$ અને $\bar{B}=0$ થશે.

$\therefore Y = 1\cdot0 + 0\cdot1 = 0+0 = 0$

XOR TRUTH TABLE

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

⇒ ટ્રુથ ટેબલ પરથી આપણે જોઈ શકીએ છીએ કે આઉટપુટ Y ત્યારે જ 1 મળશે કે જ્યારે બંને ઈનપુટ્સમાંથી માત્ર કોઈ એક જ ઈનપુટ 1 હોય, પણ બંને નહીં.

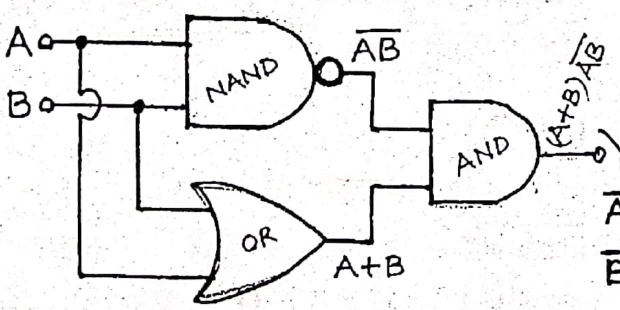
⇒ તેથી નામ exclusive OR કહેવાય છે.

⇒ સાંકેતિક રીતે આપણે XOR operation ને $\boxed{Y = A \oplus B}$ લખી શકાય.

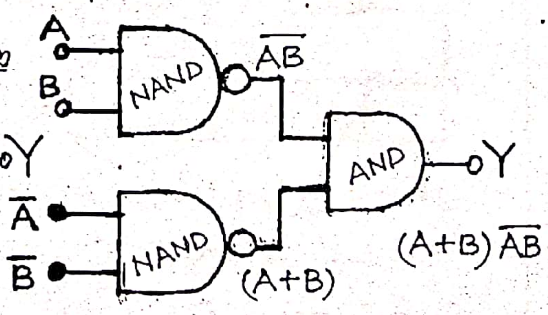
⇒ XOR ઓપરેશનને mod-2 સરવાળો પણ કહે છે અને આ સરવાળાના નિયમો નીચે મુજબના છે :

$0 \oplus 0 = 0$ , $0 \oplus 1 = 1$ , $1 \oplus 0 = 1$ , $1 \oplus 1 = 0$

⇒ ઉપરોક્ત નિયમો પરથી કહી શકાય કે mod-2 સરવાળો એ carry ને ધ્યાનમાં લીધા વગરનો દ્વિઅંકી સરવાળો છે.




(c)




(d)

⇒ આકૃતિ (c) અને (d) XOR ગેટની બીજી અન્ય રીતની રજૂઆત

દર્શાવે છે કે જે તેના અનુરૂપ ટ્રુથ ટેબલ દ્વારા દર્શાવવામાં આવ્યું છે.

⇒ ટ્રુથ ટેબલ નીચે દર્શાવ્યું છે:

| A | B | A+B | AB | $\overline{AB}$ | $Y=(A+B)\overline{AB}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0·0=0 | 1 | 0·1= 0 |
| 0 | 1 | 1 | 0·1=0 | 1 | 1·1= 1 |
| 1 | 0 | 1 | 1·0=0 | 1 | 1·1= 1 |
| 1 | 1 | 1 | 1·1=1 | 0 | 1·0= 0 |

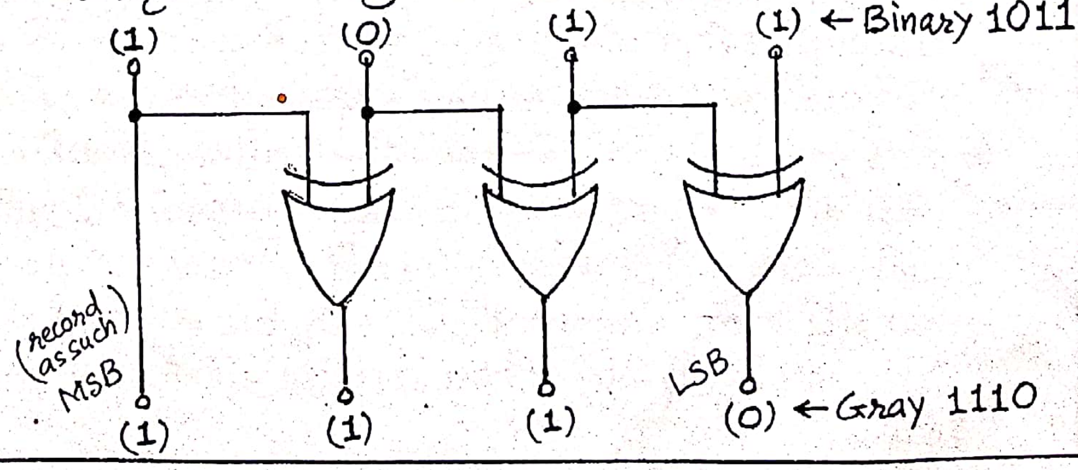
⇒ તેથી આઉટપુટ ત્યારે જ 1 મળશે કે જ્યારે બંને ઈનપુટ્સ માંથી માત્ર કોઈ એક જ ઈનપુટ 1 હોય, પણ બંને નહીં - જે exclusive OR operation છે.

## XOR ગેટના ઉપયોગો :-

### (1) Binary to Gray Code Converter (દ્વિઅંકી માંથી ગ્રે કોડમાં રૂપાંતર કરનાર) :-

⇒ XOR ગેટ દ્વિઅંકી માંથી ગ્રે કોડમાં રૂપાંતર કરવા માટે વાપરી શકાય છે.

⇒ અગાઉ આપણે આવું રૂપાંતર શીખી ગયા છીએ. આ રૂપાંતરણની પ્રક્રિયામાં દ્વિઅંકી સંખ્યાના દરેક ક્રમિક bit નો સરવાળો સામેલ છે અને carry નો અવગણેલ છે (MSB થી શરૂ કરીને) જે વાસ્તવમાં mod-2 સરવાળો છે એટલે કે XOR gate operation. જેને નીચેની આકૃતિમાં દર્શાવ્યું છે:



⇒ દ્વિઅંકી સંખ્યા 1011 નો ગ્રે કોડ 1110 છે. અહીં તેજ કોડ mod-2 સરવાળા દ્વારા મળેલ છે.

⇒ અહીં ડાબીબાજુના પ્રથમ XOR ગેટ માટે ઇનપુટ A=1, B=0 તેથી આઉટપુટ Y = 1 મળશે.

⇒ ડાબીબાજુના બીજા XOR ગેટ માટે ઇનપુટ A=0, B=1 તેથી આઉટપુટ Y = 1 મળશે.

⇒ ડાબીબાજુના ત્રીજા XOR ગેટ માટે ઇનપુટ A=1, B=1 તેથી આઉટપુટ Y=0

⇒ આમ, આકૃતિમાં દર્શાવેલ દ્વિઅંકી 1011 નો ગ્રે કોડ 1110 થશે.

## ② A Parity Checker (પેરિટી ચેકર):-

⇒ ડિજિટલ મશીન પ્રક્રિયા દરમિયાન તેની મેમરિ(memory) માંથી દરેક અંક કે શબ્દ (bits સમૂહ) ને ખેંચે છે.

⇒ તેથી bits સમૂહ, દા.ત. મશીન બે BCD સંખ્યાઓ 0100 0000 (જે 40 છે) અને 0100 0101 (જે 45 છે) નો સરવાળો કરે છે, ત્યારે દેખીતી રીતે 0 ને બદલે 1 થઈ જાય કે તેનાથી ઊંધું, એટલે કે પરિણામમાં ત્રૂટિ ઉદ્ભવે છે.

⇒ આની ચકાસણી કરવા bit સમૂહને પેરિટી બીટ જોડવી પડે.

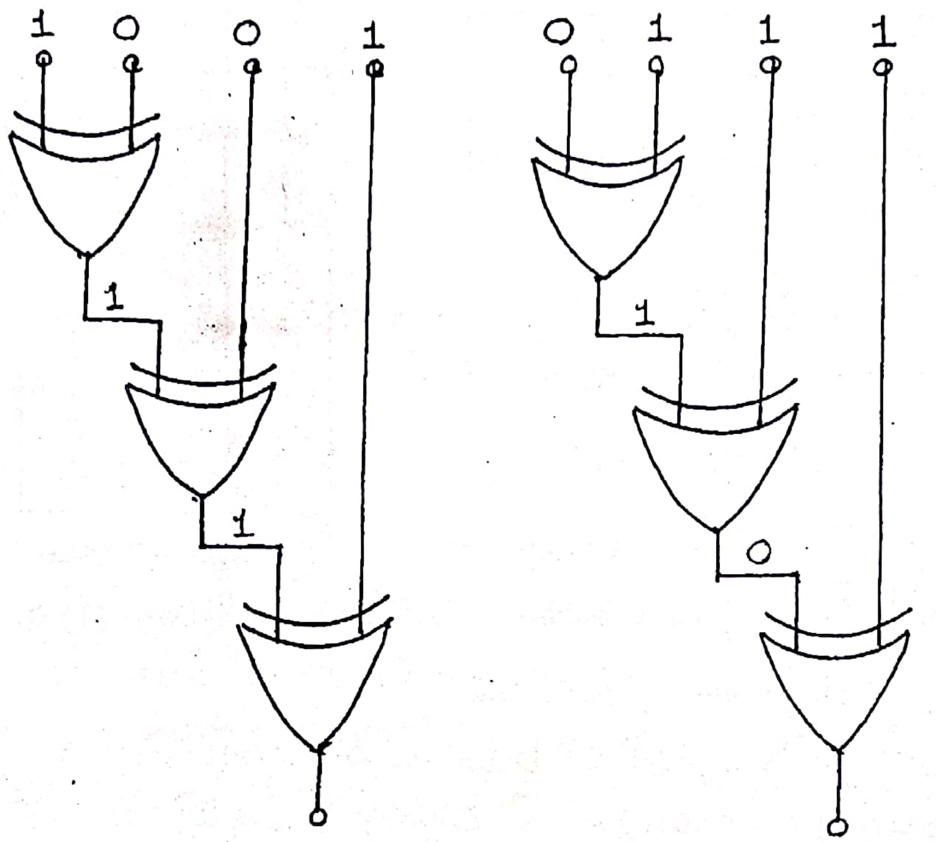
⇒ બે પ્રકારની પેરિટી bit અસ્તિત્વમાં છે: બેકી પેરિટી (even parity) અને એકી પેરિટી (odd parity)

⇒ બેકી(even) પેરિટી એટલે દ્વિઅંકી 4-bit સમૂહમાં 1 ની સંખ્યા બેકી હોય અને એકી(odd) પેરિટી એટલે દ્વિઅંકી 4-bit સમૂહમાં 1 ની સંખ્યા એકી હોય.

⇒ આવા 4-bit સમૂહવાળા દ્વિઅંકીની ગમે ત્યારે મશીન દ્વારા ચકાસણી કરી તે બેકી કે એકી પેરિટી ધરાવે છે તે ચોક્કસ રીતે નક્કી કરી શકાય છે, અને ઉદ્ભવતી ત્રૂટિ દૂર કરી શકાય છે.

⇒ તેથી આવી બેકી કે એકી પેરિટીની ચકાસણી માટે આપણે એવા પરિપથની જરૂરિયાત રહેશે કે જે બેકી પેરિટી માટે 0 અને એકી પેરિટી માટે 1 પરિણામ રજૂ કરે.

⇒ આ માટે આપણે 4-bit સમૂહના દરેક bit ના mod-2 સરવાળા દ્વારા આકૃતિ (a) અને (b) માં દર્શાવેલ પરિપથ દ્વારા પૂર્ણ કરી શકીએ.




Y=0
(even parity)
આકૃતિ (a)

Y=1
(odd parity)
આકૃતિ (b)

⇒ બેકી પેરિટી માટે આઉટપુટ Y=0 અને એકી પેરિટી માટે આઉટપુટ Y=1 મળે છે.

⇒ અહીં પ્રથમ દ્વિઅંકી સંખ્યા 1001 માં 1 બેવાર આવે છે એટલે કે બેકી પેરિટી ધરાવે છે અને બીજી સંખ્યા 0111માં 1 ત્રણ વાર એટલે કે એકી પેરિટી ધરાવે છે.

## ③ હાફ એડર (The half adder):

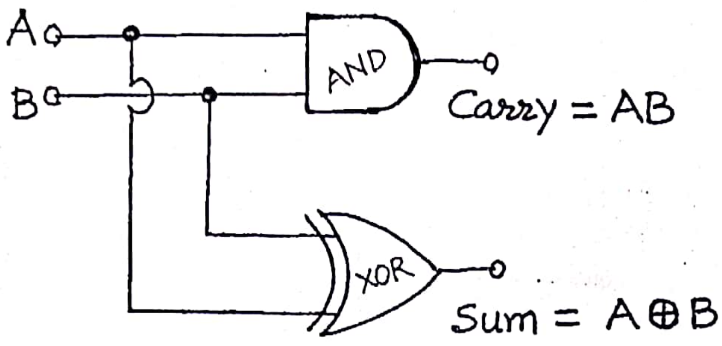
⇒ હાફ એડર જે તે ક્ષણે બે દ્વિઅંકી digits નો સરવાળો કરે છે અને તેથી તે સરવાળા માટેનો પાયાનો પરિચય છે.

⇒ જ્યારે આપણે દ્વિઅંકીમાં 1 અને 1 નો સરવાળો કરીએ છીએ ત્યારે સરવાળો (sum) = 0 અને વદ્દી (carry) = 1 મળે છે.

એટલે કે

$$\begin{array}{r l} & 1 \leftarrow \text{A input} \\ + & 1 \leftarrow \text{B input} \\ \hline & 10 \end{array}$$

carry → 1, 0 ← sum




(a) Half adder

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

TRUTH TABLE


(b) Symbol of Half Adder

⇒ આમ એક જ ક્ષણે બે bits નો સરવાળો લેતાં આપણને sum (સરવાળા) સાથે carry bit મળે છે.

⇒ સરવાળા (sum) માટે આપણે XOR ગેટનો અને carry માટે AND ગેટનો ઉપયોગ કરી મેળવીએ છીએ.

⇒ આકૃતિ (a) માં XOR ગેટ અને AND ગેટ સાથેનો હાફ એડર પરિપથ દર્શાવેલ છે.

⇒ આકૃતિ (b) તેની સંજ્ઞા દર્શાવે છે.

⇒ હાફ એડરનું ટ્રુથ ટેબલ ઉપર દર્શાવેલ છે.

⇒ આ સાદામાં સાદો adder પરિપથ છે, કારણકે તેના ઇનપુટમાં કોઈ carry નથી.

⇒ તેની ઉપયોગિતા નાના ક્રમના બીટ ધરાવતા વધુ તબક્કા વાળા adder પરિપથો સુધી મર્યાદિત રહે છે.

⇒ તે મોટા ક્રમના bits માટે કેરી (carry) ઇનપુટ ધરાવતું ન હોવાથી ઉપયોગી નથી.

PROF. KALPESH C. MEVADA

## ④ ફૂલ એડર (The Full Adder) :-

⇒ ફૂલ એડર carry ને પણ ઉમેરે છે.

⇒ તે એક જ ક્ષણે ત્રણ દ્વિઅંકી digits નો સરવાળો કરે છે અને તેથી તેમાં carry ઇનપુટ હોય છે.

⇒ જ્યારે 101 ને 101 સાથે ઉમેરવામાં આવે, તો આપણને...

$$\begin{array}{r} 1\,0\,1 \\ +\;1\,0\,1 \\ \hline 1\,0\,1\,0 \end{array}$$

મળશે.

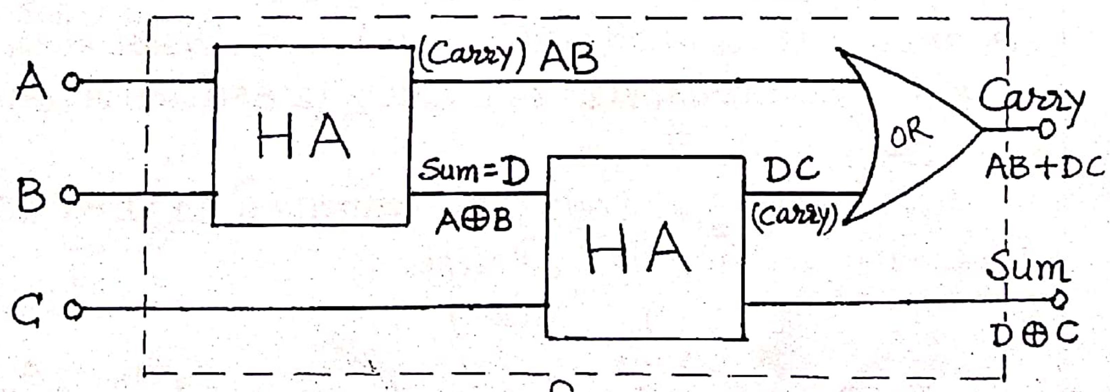
⇒ (i) જમણી બાજુની છેલ્લી કોલમ (Least Significant Column) માટે (sum) 0 અને (carry) 1 મળશે ($\because$ 1+1 = 0 sum with a carry 1)

⇒ (ii) આગળના વચ્ચેના કોલમ માટે આપણે ત્રણ અંકોનો સરવાળો કરીએ છીએ, 0 + 0 + 1 = 1 (sum) અને 0 (carry)

⇒ (iii) ત્યારબાદના આગળના (ડા.બા. પ્રથમ) કોલમ માટે આપણે જે ત્રણ અંકોનો સરવાળો કરીએ છીએ તે 1 + 1 + 0 = 0 (sum) અને 1 (carry).

⇒ તેથી ફૂલ એડર માટે એવાં પરિપથની જરૂર પડે કે જે એક જ ક્ષણે ત્રણ અંકોને કાર્યરત કરી શકે.

⇒ જે આકૃતિમાં દર્શાવ્યા મુજબના બે હાફ એડર્સ અને OR ગેટને ગોઠવીને બનાવેલા ફૂલ એડર દ્વારા થઈ શકે.




આકૃતિ
(A Full adder using two half adders and an OR gate)

⇒ ફૂલ એડરનું ટ્રુથ ટેબલ નીચે દર્શાવેલ છે.

| A | B | C | Sum | Carry |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

TRUTH TABLE OF A FULL ADDER

⇒ ફૂલ એડરની આકૃતિ પ્રમાણે, $A=1$, $B=1$ અને $C=0$ માટે આપણી પાસે $D = A \oplus B = 1 \oplus 1 = 0$ થશે, અને તેથી ફૂલ એડરની carry $= AB + DC = 1 \cdot 1 + 0 \cdot 0 = 1$ અને ફૂલ એડરનો sum $= D \oplus C = 0 \oplus 0 = 0$ મળશે.

## (5) પેરેલલ એડર (Parallel Adder) :-

⇒ જરૂરી સંખ્યાના ઘણાબધા ફૂલ એડર્સને શ્રેણીબદ્ધ જોડીને આપણે n-bit દ્વિઅંકી એડર પરિપથ મેળવી શકીએ જેને સામાન્ય રીતે પેરેલલ એડર કહેવાય છે.

⇒ આ પરિપથની ખામી સૌથી લાંબો સિગ્નલ વિલંબ સમય છે. (ઈનપુટ આપ્યાના તરત બાદ આઉટપુટ દર્શાવવામાં લાગતો સમય)

⇒ ધારોકે આપણે ચાર બીટ દ્વિઅંકી સંખ્યાઓ 1111 નો 1011 સાથે સરવાળો કરીએ તો પછી...

```
    1 1 1      ← Carry
    1 1 1 1    ← ઈનપુટ A
 +  1 0 1 1    ← ઈનપુટ B
 -----------
  1 1 0 1 0
```

જો સરવાળો 4 bit થી વધુ હોય તો તે માટેની overflow carry

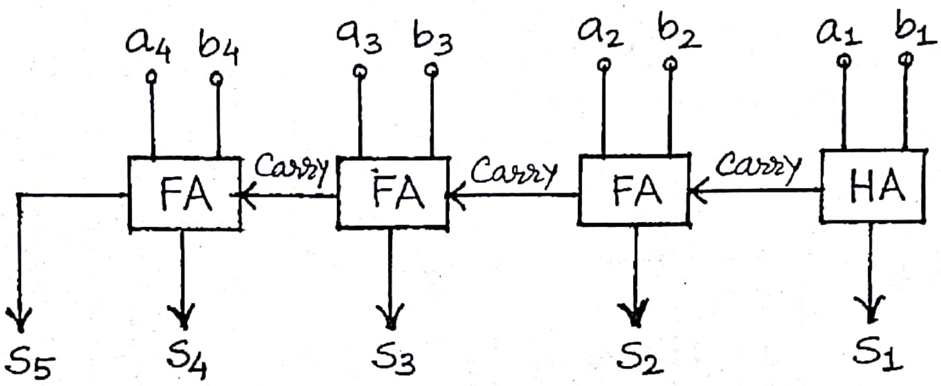
અથવા વ્યાપક રીતે

```
      a4 a3 a2 a1
 +    b4 b3 b2 b1
 ----------------
   s5 s4 s3 s2 s1
```

Overflow carry

⇒ અહીં જમણીબાજુના પ્રથમ કોલમ માટે માત્ર બે bits નો જ સરવાળો કરવાનો હોઈ અને carry ન હોવાથી ફક્ત હાફ એડરની જ જરૂરીયાત રહેશે.

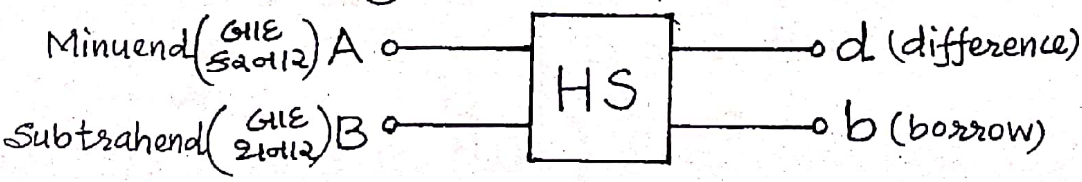
⇒ પણ, બાકીના બધા કોલમો માટે ફૂલ એડરની જરૂરીયાત પડશે કારણકે આપણે ત્રણ bits (બે અંકો અને એક carry) ઉમેરવાના છે. નીચેની આકૃતિમાં પેરેલલ ચાર bit દ્વિઅંકી એડર દર્શાવેલ છે.




A FOUR BIT BINARY ADDER

## ⑥ હાફ સબટ્રેક્ટર (Half Subtractor):-

⇒ તે એક જ ક્ષણે બે અંકોની બાદબાકી કરે છે અને આઉટપુટમાં difference (તફાવત કે બાદબાકી) અને borrow (ઉછીનું કે દશકો) રજૂ કરે છે.




$$\begin{array}{cc} A & 0 \\ B & -\,0 \\ \hline 0 & 0 \\ b & d \end{array} \qquad \begin{array}{cc} A & 0 \\ B & -\,1 \\ \hline 1 & 1 \\ b & d \end{array}$$

$$\begin{array}{cc} A & 1 \\ B & -\,0 \\ \hline 0 & 1 \\ b & d \end{array} \qquad \begin{array}{cc} A & 1 \\ B & -\,1 \\ \hline 0 & 0 \\ b & d \end{array}$$

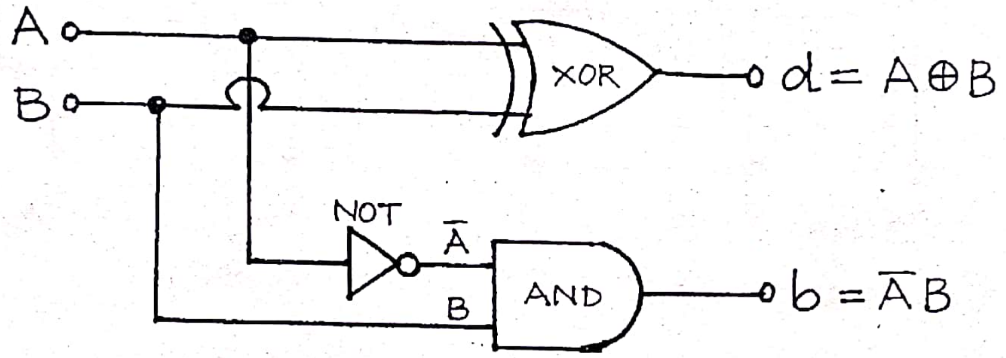
⇒ આ કિસ્સા માટે $0 - 1 = -1$. ઋણ નિશાની સૂચવે છે કે આગળના ઉચ્ચ અંકમાંથી borrow લેવાની જરૂર છે. જેમ કે આગળના ઉચ્ચ અંકમાંથી લીધેલ borrow નું મૂલ્ય અહીં માટે 2 થશે. તેથી $2 - 1 = 1$ difference માટે અને 1 borrow માટે.

⇒ આ પરિણામો ટ્રુથ ટેબલમાં દર્શાવ્યા છે.

| ઇનપુટ | | આઉટપુટ | |
|---|---|---|---|
| A | B | b | d |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

TRUTH TABLE : H.S.

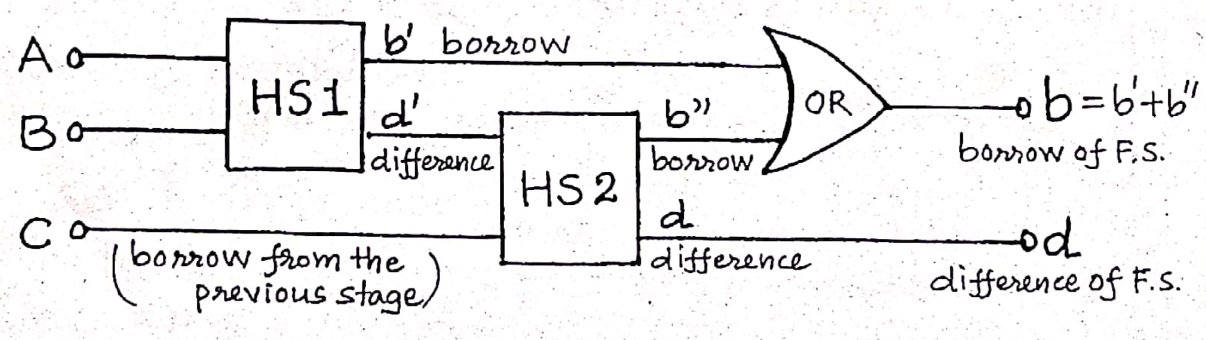
⇒ આવાં પરિણામો મેળવવા માટે નીચે આકૃતિમાં દર્શાવ્યા મુજબનો XOR, AND અને NOT ગેટનો ઉપયોગ કરીને બનાવેલ પરિપથ હાફ સબટ્રેક્ટર ઉપયોગી થઈ પડે છે.




આકૃતિ HALF SUBTRACTOR

| $A \oplus B$ | d | $\bar{A} \cdot B$ | b |
|---|---|---|---|
| $0 \oplus 0$ | $= 0$ | $\bar{0} \cdot 0 = 1 \cdot 0$ | $= 0$ |
| $0 \oplus 1$ | $= 1$ | $\bar{0} \cdot 1 = 1 \cdot 1$ | $= 1$ |
| $1 \oplus 0$ | $= 1$ | $\bar{1} \cdot 0 = 0 \cdot 0$ | $= 0$ |
| $1 \oplus 1$ | $= 0$ | $\bar{1} \cdot 1 = 0 \cdot 1$ | $= 0$ |

## 7. ફૂલ સબટ્રેક્ટર (Full Subtractor):




⇒ આમાં પાછળના તબક્કાની બાદબાકી પ્રક્રિયાની borrow (જેને અહીં C તરીકે દર્શાવેલ છે) એક ઇનપુટ તરીકે, A બાદ કરનાર

અને B બાદ થનાર છે.

⇒ તે બે હાફ સબટ્રેક્ટર્સનો ઉપયોગ કરે છે.

⇒ આકૃતિમાં ફૂલ સબટ્રેક્ટર દર્શાવેલ છે

| HS-1 | | | | HS-2 | | | | Final Output FS | |
|---|---|---|---|---|---|---|---|---|---|
| Input | | Output | | Input | | Output | | OR ગેટ | Diff. |
| A | B | b' | d' | d' | C | b'' | d | b'+b''=b | d |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Table 1 : RESULTS OF HS-1, HS-2 AND FINAL OUTPUT

⇒ તેથી FS ના truth table ને નીચે દર્શાવેલ ટેબલ-2 મુજબ લખી શકાય:

TRUTH TABLE OF F.S.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| A | B | C | b | d |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

PROF. KALPESH MEVADA