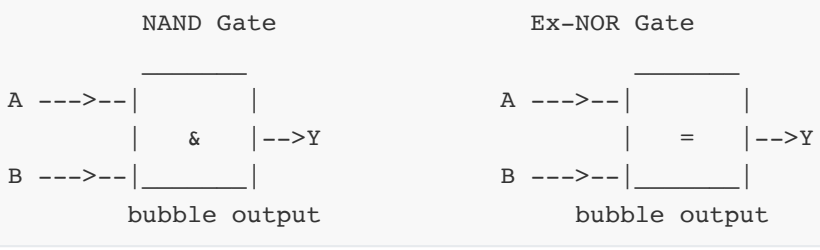
પ્રશ્ન 1(અ) [3 માર્ક્સ]

NAND અને Ex-NOR ગેટનો સીમ્બોલ દોરો અને તેમનુ લોજિક ટેબલ લખો.

જવાબ:

NAND અને Ex-NOR ગેટના સિમ્બોલ અને ટ્રુથ ટેબલ:



A	B	Y (NAND)
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Y (Ex-NOR)
0	0	1
0	1	0
1	0	0
1	1	1

- **NAND ગેટ**: ફક્ત ત્યારે જ આઉટપુટ LOW હોય છે જ્યારે બધા ઇનપુટ HIGH હોય
- **Ex-NOR ગેટ**: જ્યારે ઇનપુટ SAME હોય ત્યારે આઉટપુટ HIGH હોય છે

મેમરી ટ્રીક: "NAND બધા એક માટે ના કહે છે, Ex-NOR સરખા સિગ્નલ માટે હા કહે છે"

પ્રશ્ન 1(બ) [4 માર્ક્સ]

જનદેશ મુિબ કરો: (i) 2's કોમ્પ્લેમેંટ નો ઉપયોગ કરીને બાદબાકી કરો $(1011001)_2 - (1001101)_2$ (ii) $(10110101)_2 = (\ )_{10} = (\ )_{16}$

જવાબ:

(i) 2's કોમ્પ્લેમેંટનો ઉપયોગ કરીને બાદબાકી:

```
પગલું 1: બીજા નંબરનો 2's કોમ્પ્લેમેંટ શોધો (1001101)₂
         1's કોમ્પ્લેમેંટ: 0110010
         1 ઉમેરો:          0110011

પગલું 2: મિનુએંડ અને 2's કોમ્પ્લેમેંટને સરવાળો કરો
         1011001
       + 0110011
         -------
        10001100

પગલું 3: ઓવરફ્લો બિટને છોડી દો
         પરિણામ = 0001100 = (0001100)₂
```

(ii) $(10110101)_2$ નું રૂપાંતર:

```
દશાંશમાં:
1×2⁷ + 0×2⁶ + 1×2⁵ + 1×2⁴ + 0×2³ + 1×2² + 0×2¹ + 1×2⁰
= 128 + 0 + 32 + 16 + 0 + 4 + 0 + 1
= 181₁₀

હેક્સાડેસિમલમાં:
1011 0101
 B    5
= B5₁₆
```

- **2's કોમ્પ્લેમેંટ**: બિટ્સને ઉલટાવો અને 1 ઉમેરો
- **બાઇનરી થી દશાંશ**: દરેક બિટને તેની પોઝિશન વેલ્યુ (2^n) થી ગુણો
- **બાઇનરી થી હેક્સ**: બિટ્સને ચારના જૂથમાં વિભાજિત કરો, દરેક જૂથને રૂપાંતરિત કરો

મેમરી ટ્રીક: "બિટ્સ ઉલટાવો 1 ઉમેરો, કેરી છોડી દો"

પ્રશ્ન 1(ક) [7 માર્ક્સ]

શોધો (i) $(4356)_{10} = (\ )_8 = (\ )_{16} = ()_2$ (ii) $(101.01)_2 \times (11.01)_2$ (iii) ભાગાકાર કરો $(101101)_2$ ને $(110)_2$ વડે.

જવાબ:

(i) નંબર સિસ્ટમ રૂપાંતર:

```
દશાંશથી ઓક્ટલ:
4356 ÷ 8 = 544 બાકી 4
544 ÷ 8 = 68  બાકી 0
68 ÷ 8 = 8    બાકી 4
8 ÷ 8 = 1     બાકી 0
1 ÷ 8 = 0     બાકી 1
નીચેથી વાંચીને: (4356)₁₀ = (10404)₈

દશાંશથી હેક્સાડેસિમલ:
4356 ÷ 16 = 272 બાકી 4
272 ÷ 16 = 17  બાકી 0
```

```
17 ÷ 16 = 1    બાકી 1
1 ÷ 16 = 0     બાકી 1
નીચેથી વાંચીને: (4356)₁₀ = (1104)₁₆

દશાંશથી બાઇનરી:
4356 = 1000100000100₂
```

(ii) બાઇનરી ગુણાકાર:

```
      101.01
    × 11.01
    -------
      10101
     10101
    10101
   10101
   ---------
   1111.1101
```

(iii) બાઇનરી ભાગાકાર:

```
          111.
       ------
110 ) 101101
      110
      -----
       11101
       110
       -----
       1001
        110
        ----
        11
```

- **દશાંશ થી ઓક્ટલ**: વારંવાર 8 થી ભાગો
- **દશાંશ થી હેક્સ**: વારંવાર 16 થી ભાગો
- **બાઇનરી ઓપરેશન્સ**: દશાંશની જેમ જ પ્રક્રિયા અનુસરો

મેમરી ટ્રીક: "ભાગો અને બાકીને નીચેથી ઉપર ગોઠવો"

પ્રશ્ન 1(ક-OR) [7 માર્ક્સ]

શોધો $(8642)_{10} = (\ )_8 = (\ )_{16} = (\ )_2$ (ii) NOR અને Ex-OR ગેટનો સીમ્બોલ દોરો અને તેમનુ લોજિક ટેબલ લખો.

જવાબ:

(i) નંબર સિસ્ટમ રૂપાંતર:

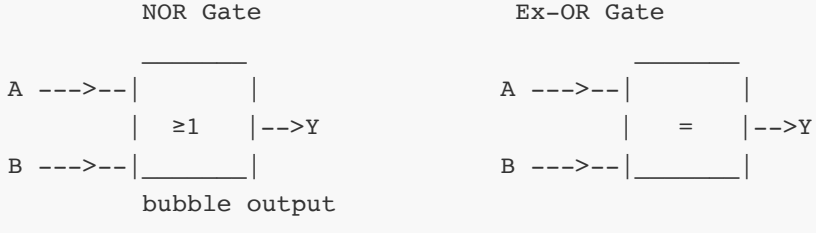
```
દશાંશથી ઓક્ટલ:
8642 ÷ 8 = 1080 બાકી 2
```

```
1080 ÷ 8 = 135  બાકી 0
135 ÷ 8 = 16    બાકી 7
16 ÷ 8 = 2      બાકી 0
2 ÷ 8 = 0       બાકી 2
નીચેથી વાંચીને: (8642)₁₀ = (20702)₈

દશાંશથી હેક્સાડેસિમલ:
8642 ÷ 16 = 540 બાકી 2
540 ÷ 16 = 33   બાકી 12(C)
33 ÷ 16 = 2     બાકી 1
2 ÷ 16 = 0      બાકી 2
નીચેથી વાંચીને: (8642)₁₀ = (21C2)₁₆

દશાંશથી બાઇનરી:
8642 = 10000111000010₂
```

(ii) NOR અને Ex-OR ગેટ્સ:



A	B	Y (NOR)
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Y (Ex-OR)
0	0	0
0	1	1
1	0	1
1	1	0

- **NOR ગેટ**: ફક્ત ત્યારે જ આઉટપુટ HIGH હોય છે જ્યારે બધા ઇનપુટ LOW હોય
- **Ex-OR ગેટ**: જ્યારે ઇનપુટ DIFFERENT હોય ત્યારે આઉટપુટ HIGH હોય છે

મેમરી ટ્રીક: "NOR બધા શૂન્ય માટે હા કહે છે, Ex-OR અલગ સિગ્નલ માટે હા કહે છે"

પ્રશ્ન 2(અ) [3 માર્ક્સ]

સાબિત કરો xy+xz+yz' = xz+yz'

જવાબ:

```
ડાબી બાજુ: xy + xz + yz'
= xy + xz + yz'
= x(y + z) + yz'          [વિતરણ ગુણધર્મ]
= xy + xz + yz'           [વિસ્તાર]
= xy + yz' + xz           [પુનર્ગઠન]
= y(x + z') + xz          [વિતરણ ગુણધર્મ]
= xy + yz' + xz           [વિસ્તાર]
= (x + y)z' + xz          [પુનર્ગઠન]
= xz' + yz' + xz          [વિસ્તાર]
= x(z' + z) + yz'         [વિતરણ ગુણધર્મ]
= x(1) + yz'              [પૂરક ગુણધર્મ]
= x + yz'                 [ઓળખ ગુણધર્મ]
= xz + x(1-z) + yz'       [x = xz + xz']
= xz + xz' + yz'          [વિસ્તાર]
= xz + z'(x + y)          [વિતરણ ગુણધર્મ]
= xz + z'x + z'y          [વિસ્તાર]
= xz + xz' + yz'          [પુનર્ગઠન]
= x(z + z') + yz'         [વિતરણ ગુણધર્મ]
= x(1) + yz'              [પૂરક ગુણધર્મ]
= x + yz'                 [ઓળખ ગુણધર્મ]
= xz + yz'                [જમણી બાજુ સમાન]
```

- **વિતરણ ગુણધર્મ**: x(y+z) = xy+xz
- **પૂરક ગુણધર્મ**: z+z' = 1
- **ઓળખ ગુણધર્મ**: x×1 = x

મેમરી ટ્રીક: "ફેક્ટર કરો, એક્સપાન્ડ કરો, ફરીથી ગોઠવો, ફરીથી ફેક્ટર કરો"

પ્રશ્ન 2(બ) [4 માર્ક્સ]

k-મેપની મદદથી f(W,X,Y,Z) = ∑m(0,1,2,3,5,7,8,9,11,14) એક્સ્પ્રેશન ઘટાડો.

જવાબ:

f(W,X,Y,Z) = ∑m(0,1,2,3,5,7,8,9,11,14) માટે K-Map:

```
      YZ
WX   00  01  11  10
00    1   1   0   1
01    1   1   1   0
11    0   0   1   1
10    1   1   0   0
```

ગ્રુપિંગ:

- ગ્રુપ 1: m(0,1,2,3) = W'X' (2×2 ગ્રુપ)
- ગ્રુપ 2: m(0,1,8,9) = Y' (2×2 ગ્રુપ)
- ગ્રુપ 3: m(2,3,11) = X'Z (2×2 ગ્રુપ, વ્રેપિંગ સાથે)
- ગ્રુપ 4: m(7,14) = XZ (જોડી)

સરળીકૃત સમીકરણ:
f(W,X,Y,Z) = W'X' + Y' + X'Z + XZ

- **K-Map ટેકનિક**: બાજુના 1ને 2ની ઘાતમાં ગ્રુપ કરો
- **દરેક ગ્રુપ**: સરળીકૃત સમીકરણમાં એક પદનું પ્રતિનિધિત્વ કરે છે
- **મોટા ગ્રુપ**: વધુ સરળ સમીકરણનો અર્થ

મેમરી ટ્રીક: "2ની ઘાતો સમીકરણને નવું બનાવે છે"

પ્રશ્ન 2(ક) [7 માર્ક્સ]

NOR ગેટને યુજનવસસલ ગેટ તરીકે સમજાવો

જવાબ:

NOR યુનિવર્સલ ગેટ તરીકે:

NOR ગેટ બધા મૂળભૂત લોજિક ફંકશન્સને અમલમાં મૂકી શકે છે, જે તેને યુનિવર્સલ ગેટ બનાવે છે.

NOR વડે મૂળભૂત ગેટ્સનું અમલીકરણ:

ગેટ	NOR સાથે અમલીકરણ
NOT	A NOR A
OR	(A NOR B) NOR (A NOR B)
AND	(A NOR A) NOR (B NOR B)

સર્કિટ ડાયાગ્રામ્સ:

```
NOT Gate using NOR:
A--->|NOR|-->Y

OR Gate using NOR:
A--->|     |       |     |
     | NOR |------|     |
B--->|     |       | NOR |-->Y
                   |     |
                   |     |

AND Gate using NOR:
A--->|NOR|-------|     |
                  |     |
                  | NOR |-->Y
                  |     |
```

```
B--->|NOR|-------|       |
```

- **યુનિવર્સલ ગેટ**: કોઈપણ બુલિયન ફંક્શન અમલમાં મૂકી શકે છે
- **NOR ઓપરેશન**: NOT OR, આઉટપુટ હાઈ ફક્ત ત્યારે જ જ્યારે બધા ઇનપુટ લો હોય
- **અમલીકરણ ખર્ચ**: જટિલ ફંક્શન્સ માટે બહુવિધ NOR ગેટ્સની જરૂર પડે છે

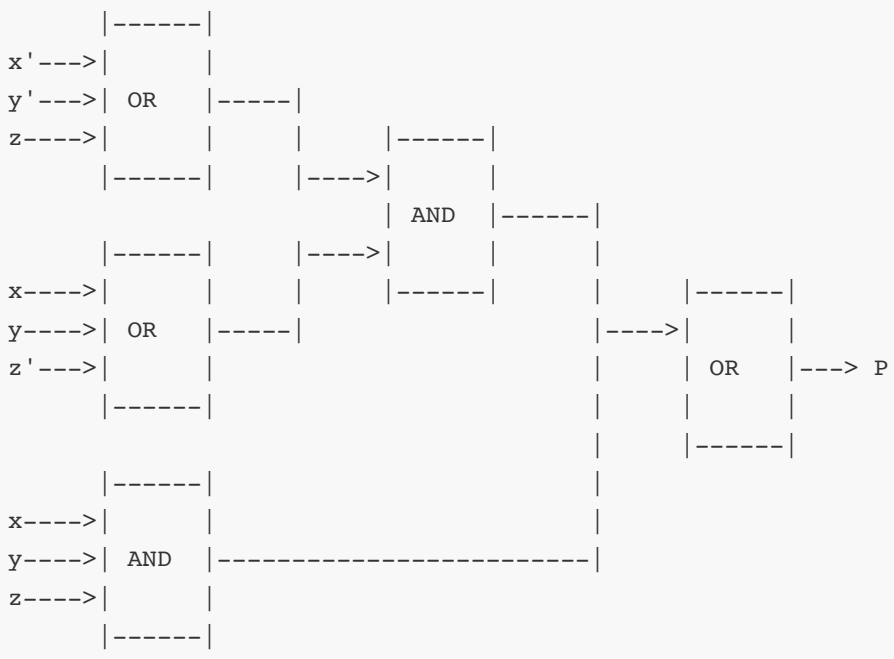
મેમરી ટ્રીક: "NOR એટલે Not-OR, પણ Not-AND-OR બધું કરી શકે છે"

પ્રશ્ન 2(અ-OR) [3 માર્ક્સ]

બુજલયન એક્સ્પ્રેશન P = (x'+y'+z)(x+y+z')+(xyz) માટે લોજિક સજકસટ દોરો

જવાબ:

P = (x'+y'+z)(x+y+z')+(xyz) માટે લોજિક સર્કિટ:



- **પગલું 1**: દરેક પ્રોડક્ટ ટર્મનું અમલીકરણ કરો
- **પગલું 2**: પછી OR ગેટ સાથે જોડો
- **પગલું 3**: ઓપરેટર પ્રાથમિકતા અનુસરો

મેમરી ટ્રીક: "પહેલા પ્રોડક્ટ્સ, પછી તેમનો સરવાળો કરો"

પ્રશ્ન 2(બ-OR) [4 માર્ક્સ]

K-મેપ પદ્ધજતનો ઉપયોગ કરીને f(W,X,Y,Z) = ∑m(1,3,7,11,15) એક્સ્પ્રેશન ને રીડ્યુસ કરો િેમા ડોંટ કેર ની શરત d(0,2,5) વાપરો.

જવાબ:

ડોન્ટ કેર કન્ડિશન્સ સાથે K-Map:

```
      YZ
WX   00  01  11  10
00    d   1   0   d
01    0   0   1   d
11    0   0   1   1
10    0   0   1   0
```

ગ્રુપિંગ:

- ગ્રુપ 1: m(1,3,7,15) + d(0,2) = X'Z + YZ (જોડીઓ)
- ગ્રુપ 2: m(7,15) + d(5) = WYZ (ચતુષ્ક)

સરળીકૃત સમીકરણ:
f(W,X,Y,Z) = X'Z + YZ

- **ડોન્ટ કેર કન્ડિશન્સ**: સરળતા માટે 0 અથવા 1 તરીકે ગણી શકાય છે
- **ઇષ્ટતમ ગ્રુપિંગ**: મોટા જૂથો બનાવવા માટે ડોન્ટ કેર્સનો ઉપયોગ કરો
- **સરળીકરણનો ધ્યેય**: પદોની સંખ્યા ઘટાડવી

મેમરી ટ્રીક: "ડોન્ટ કેર્સ મોટા ચોરસ બનાવવામાં મદદ કરે છે"

પ્રશ્ન 2(ક-OR) [7 માર્ક્સ]

બુજલયન થીયરમ અને તેની તમામ પ્ર ોપ્રટીઝ લખો.

જવાબ:

મૂળભૂત બુલિયન થિયરમ અને તેના ગુણધર્મો:

નિયમ/ગુણધર્મ	સમીકરણ
ઓળખ નિયમ	A + 0 = A, A · 1 = A
નલ નિયમ	A + 1 = 1, A · 0 = 0
ઇડેમપોટન્ટ નિયમ	A + A = A, A · A = A
પૂરક નિયમ	A + A' = 1, A · A' = 0
ક્રમવિનિમય નિયમ	A + B = B + A, A · B = B · A
સંગઠન નિયમ	A + (B + C) = (A + B) + C, A · (B · C) = (A · B) · C
વિતરણ નિયમ	A · (B + C) = A · B + A · C, A + (B · C) = (A + B) · (A + C)
અવશોષણ નિયમ	A + (A · B) = A, A · (A + B) = A
ડીમોર્ગનનો થિયરમ	(A + B)' = A' · B', (A · B)' = A' + B'
ડબલ કોમ્પ્લિમેન્ટ	(A')' = A
કોન્સેન્સસ થિયરમ	(A · B) + (A' · C) + (B · C) = (A · B) + (A' · C)

- **મૂળભૂત ઓપરેશન્સ**: AND (·), OR (+), NOT (')
- **કી એપ્લિકેશન્સ**: સર્કિટ સરળીકરણ અને ડિઝાઇન
- **થિયરમનું મહત્વ**: ગેટ કાઉન્ટ અને જટિલતા ઘટાડે છે

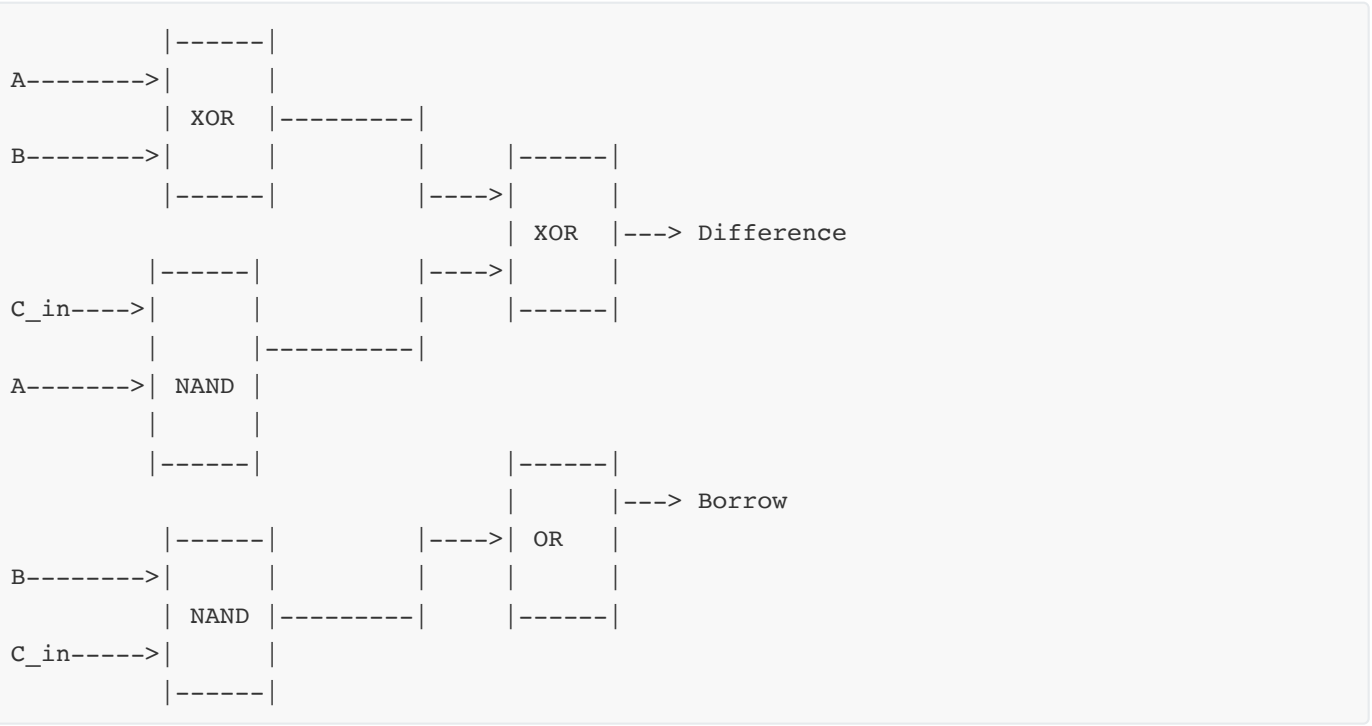
મેમરી ટ્રીક: "COIN-CADDAM" (કોમ્પ્લિમેન્ટરી, ડિસ્ટ્રિબ્યુટિવ, એસોસિએટિવ, વગેરે)

પ્રશ્ન 3(અ) [3 માર્ક્સ]

ફુલ સબ્ટરેક્સ્પટરની લોજિક સજકસટ દોરો અને તેનું કાયસ સમજાવો.

જવાબ:

ફુલ સબ્ટ્રેક્ટર સર્કિટ:



ટ્રુથ ટેબલ:

A	B	C_in	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- **ડિફરન્સ**: A ⊕ B ⊕ C_in (બધા ઇનપુટનો XOR)
- **બોરો**: C_in·(A ⊕ B) + B·A' (જરૂર પડે ત્યારે જનરેટ થાય છે)

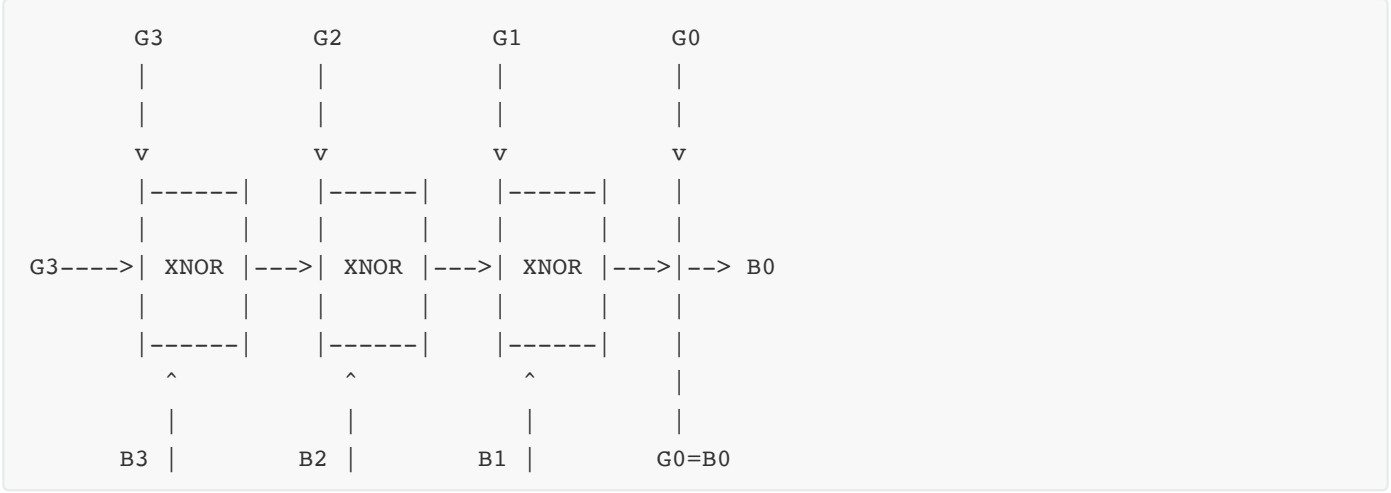
મેમરી ટ્રીક: "જ્યારે સબ્ટ્રાહેન્ડ મિનુએન્ડ કરતા વધારે હોય ત્યારે બોરોની જરૂર પડે છે"

પ્રશ્ન 3(બ) [4 માર્ક્સ]

ગ્રે થી બાઈનરી કોડ કન્વર્ટરની સર્કિટ દોરો.

જવાબ:

ગ્રે થી બાઇનરી કોડ કન્વર્ટર (4-બિટ):



રૂપાંતરણ ટેબલ:

ગ્રે	બાઇનરી
G3G2G1G0	B3B2B1B0
0000	0000
0001	0001
0011	0010
0010	0011
0110	0100
...	...

- **રૂપાંતરણ સિદ્ધાંત**: $B_3 = G_3$, $B_2 = B_3 \oplus G_2$, $B_1 = B_2 \oplus G_1$, $B_0 = B_1 \oplus G_0$
- **મુખ્ય વિશેષતા**: દરેક બાઇનરી બિટ તમામ અગાઉના ગ્રે બિટ્સ પર આધાર રાખે છે
- **અનુપ્રયોગ**: ડિજિટલ ટ્રાન્સમિશનમાં ભૂલ શોધન

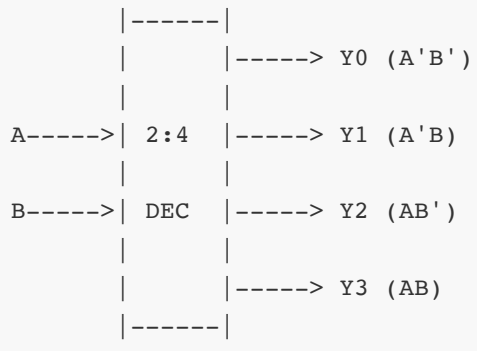
મેમરી ટ્રીક: "MSB રહે છે, અન્ય અગાઉના બાઇનરીની સાથે XOR થાય છે"

પ્રશ્ન 3(ક) [7 માર્ક્સ]

2:4 ડીકોડર અને 4:1 મજટ્લેક્સ્પસર દોરો અને તેનું કાયસ સમજાવો.

જવાબ:

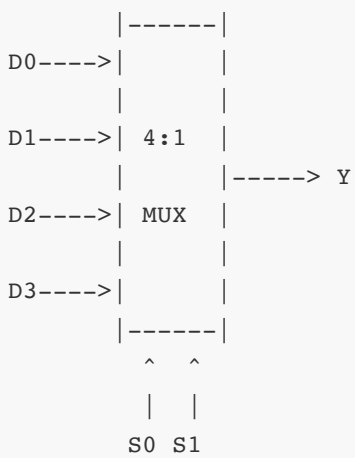
2:4 ડિકોડર:



ટ્રુથ ટેબલ:

A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

4:1 મલ્ટિપ્લેક્સર:



ટ્રુથ ટેબલ:

S1	S0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

- **ડિકોડર**: બાઇનરી કોડને વન-હોટ આઉટપુટમાં રૂપાંતરિત કરે છે
- **મલ્ટિપ્લેક્સર**: સિલેક્શન લાઇન્સના આધારે ઘણા ઇનપુટમાંથી એક પસંદ કરે છે
- **અનુપ્રયોગો**: મેમરી એડ્રેસિંગ, ડેટા રાઉટિંગ

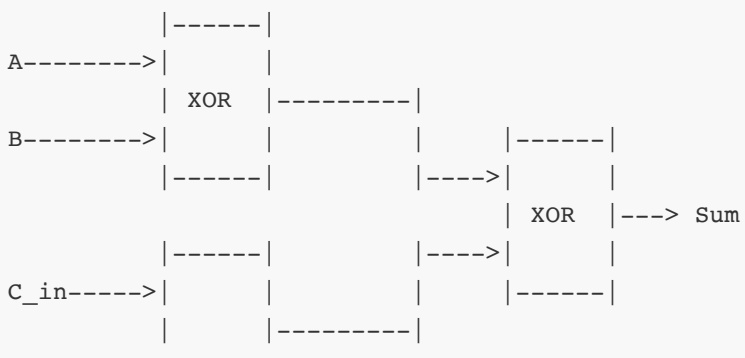
મેમરી ટ્રીક: "ડિકોડર: એક-થી-ઘણા, મક્સ: ઘણા-થી-એક"

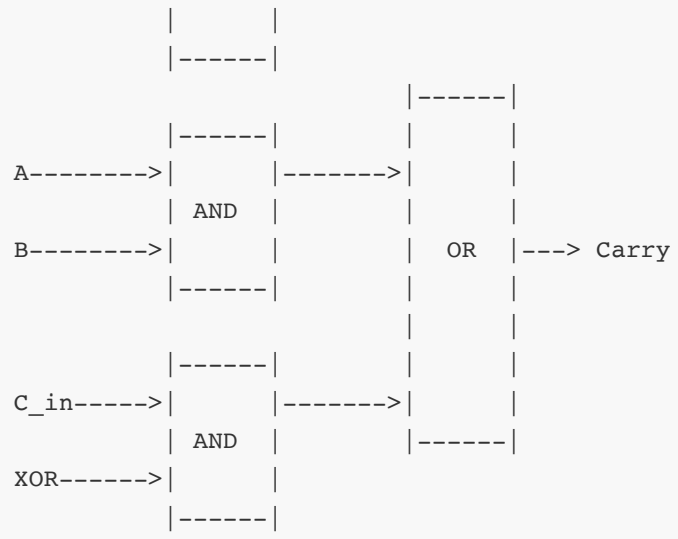
પ્રશ્ન 3(અ-OR) [3 માર્ક્સ]

ફુલ એડરની લોજિક સજકસટ દોરો અને તેનું કાયસ સમજાવો.

જવાબ:

ફુલ એડર સર્કિટ:





ટ્રુથ ટેબલ:

A	B	C_in	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

- **સમ**: A ⊕ B ⊕ C_in (બધા ઇનપુટનો XOR)
- **કેરી**: (A · B) + (C_in · (A ⊕ B)) (જરૂર પડે ત્યારે જનરેટ થાય છે)

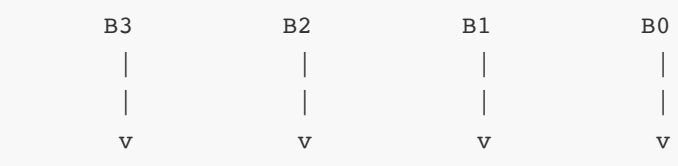
મેમરી ટ્રીક: "સમ વિષમ હોય છે, કેરીને ઓછામાં ઓછા બે 1ની જરૂર પડે છે"

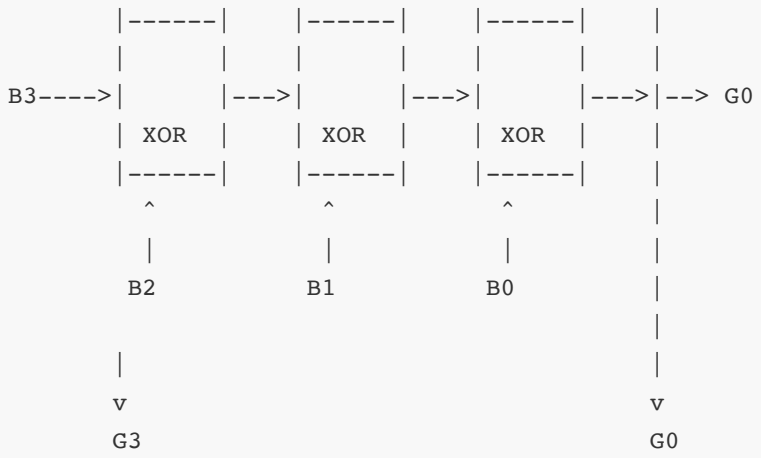
પ્રશ્ન 3(બ-OR) [4 માર્ક્સ]

બાઈનરી થી ગ્રે કોડ કન્વર્ટરની સર્કિટ દોરો.

જવાબ:

બાઇનરી થી ગ્રે કોડ કન્વર્ટર (4-બિટ):





રૂપાંતરણ ટેબલ:

બાઇનરી	ગ્રે
B3B2B1B0	G3G2G1G0
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
...	...

- **રૂપાંતરણ સિદ્ધાંત**: $G_3 = B_3$, $G_2 = B_3 \oplus B_2$, $G_1 = B_2 \oplus B_1$, $G_0 = B_1 \oplus B_0$
- **મુખ્ય વિશેષતા**: આસન્ન કોડ વચ્ચે ફક્ત એક બિટ બદલાય છે
- **અનુપ્રયોગ**: રોટરી એન્કોડર્સ, ભૂલ શોધન

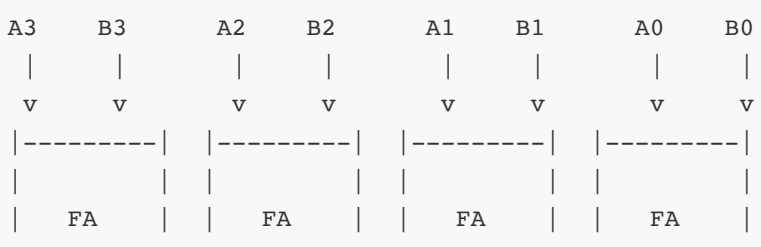
મેમરી ટ્રીક: "MSB રહે છે, અન્ય બિટ્સ આસન્ન બાઇનરી બિટ્સ સાથે XOR કરે છે"

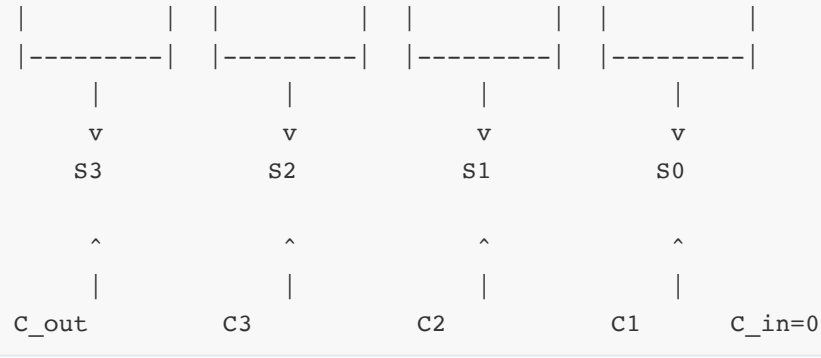
પ્રશ્ન 3(ક-OR) [7 માર્ક્સ]

ફુલ ઍડરનો ઉપયોગ કરીને 4 બીટ પેરેલલ ઍડરનો લૉજિક ડાયાગ્રામ દોરો અને તેનું કાયસ સમજાવો

જવાબ:

ફુલ એડરનો ઉપયોગ કરીને 4-બિટ પેરેલલ એડર:





ઓપરેશન:

1. દરેક ફુલ એડર (FA) તત્સ્થાની બિટ્સ (Ai, Bi) તેમજ અગાઉના સ્ટેજમાંથી કેરી ઉમેરે છે
2. સમ (Si) અને કેરી (Ci+1) આગળના સ્ટેજ માટે ઉત્પન્ન કરે છે
3. પ્રથમ FA નું C_in 0 છે (અથવા 1 ઉમેરવા માટે 1 હોઈ શકે છે)
4. છેલ્લા FA નું C_out ઓવરફ્લો સૂચવે છે

ઉદાહરણ સરવાળો: 1101 + 1011

- $A_3A_2A_1A_0$ = 1101
- $B_3B_2B_1B_0$ = 1011
- C_in = 0
- $S_3S_2S_1S_0$ = 1000
- C_out = 1 (ઓવરફ્લો સૂચવે છે, વાસ્તવિક પરિણામ 11000 છે)
- **પેરેલલ એડર**: એક સાથે ઘણી બિટ્સ ઉમેરે છે
- **કેરી પ્રોપેગેશન**: સ્પીડ માટે મુખ્ય મર્યાદિત પરિબળ
- **એડર એપ્લિકેશન્સ**: ALU, એડ્રેસ ગણતરી

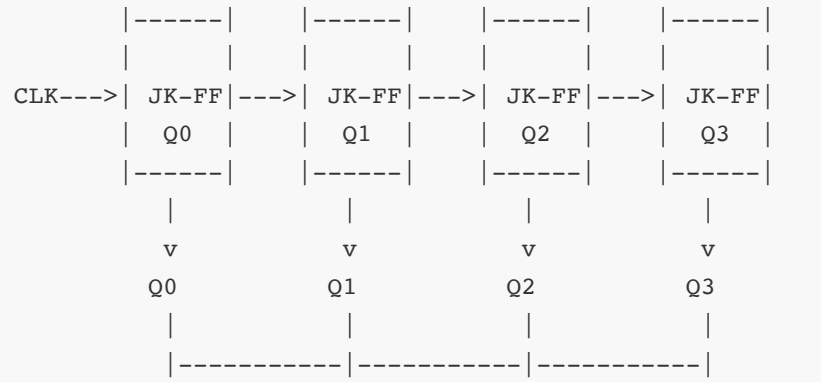
મેમરી ટ્રીક: "કેરી જમણેથી ડાબે તરફ વહે છે"

પ્રશ્ન 4(અ) [3 માર્ક્સ]

BCD કાઉન્ટર નો ડાયાગ્રામ દોરો.

જવાબ:

BCD કાઉન્ટર ડાયાગ્રામ:



```
          |
          v
      |------|
      | NAND |----+
      |------|    |
                  |
                  v
               RESET
```

કાઉન્ટર સિક્વન્સ:

કાઉન્ટ	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0

- **BCD કાઉન્ટર**: 0 થી 9 સુધી ગણે છે, પછી રીસેટ થાય છે
- **રીસેટ મેકેનિઝમ**: 10 (1010) ની ગણતરીને શોધે છે અને 0 પર રીસેટ કરે છે
- **અનુપ્રયોગો**: ડિજિટલ ઘડિયાળ, ફ્રિક્વન્સી કાઉન્ટર્સ

મેમરી ટ્રીક: "માત્ર દશાંશ અંકો (0-9) ગણે છે"

પ્રશ્ન 4(બ) [4 માર્ક્સ]

T જલલપ લલોપનો ડાયાગ્રામ દોરો અને ટ્રુથ ટેબલ સાથે તેનું કાયસ સમજાવો

જવાબ:

T ફ્લિપ-ફ્લોપ ડાયાગ્રામ:

```
        |------|
        |      |
T------>|      |
        |  T   |
CLK---->|  FF  |-----> Q
        |      |
        |      |-----> Q'
        |------|
```

JK ફ્લિપ-ફ્લોપનો ઉપયોગ કરીને અમલીકરણ:

```
        |------|
        |      |
T------>| J    |
        |      |
        | JK   |-----> Q
CLK---->|      |
        | FF   |-----> Q'
        |      |
T------>| K    |
        |------|
```

ટ્રુથ ટેબલ:

T	CLK	Q(next)
0	↑	Q
1	↑	Q'

- **T=0**: આઉટપુટમાં કોઈ ફેરફાર નહીં (હોલ્ડ)
- **T=1**: આઉટપુટ ટોગલ થાય છે (કોમ્પ્લિમેન્ટ)
- **ટોગલ ઓપરેશન**: T=1 હોય ત્યારે દરેક ક્લોક પલ્સ પર સ્થિતિ બદલે છે

મેમરી ટ્રીક: "T એટલે ટોગલ, 0 રાખે છે 1 પલટાવે છે"

પ્રશ્ન 4(ક) [7 માર્ક્સ]

જશલટ રજીટર શું છે? જવજવધ પ્ર કારના જશલટ રજીટરની યાદી આપે છે. કોઈપણ એક પ્ર કારના જશલટ રજીટરની કામગીરી તેની લોજીક સકીટ બનાવીને સમજાવો.

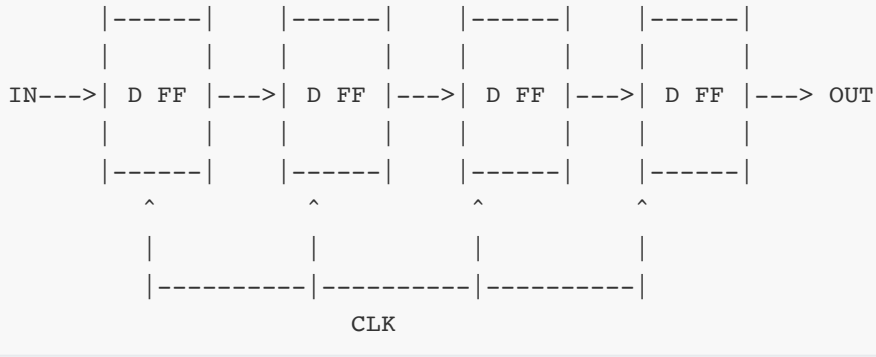
જવાબ:

શિફ્ટ રજિસ્ટર વ્યાખ્યા:
શિફ્ટ રજિસ્ટર એ એક સિક્વેન્શિયલ લોજિક સર્કિટ છે જે બાઇનરી ડેટા સ્ટોર કરે છે અને શિફ્ટ કરે છે. તેમાં એક શ્રેણીબદ્ધ ફ્લિપ-ફ્લોપ્સ હોય છે જ્યાં એક ફ્લિપ-ફ્લોપનો આઉટપુટ પછીના ફ્લિપ-ફ્લોપનો ઇનપુટ બને છે.

શિફ્ટ રજિસ્ટરના પ્રકારો:

પ્રકાર	વર્ણન
SISO	સીરિયલ ઇનપુટ સીરિયલ આઉટપુટ
SIPO	સીરિયલ ઇનપુટ પેરેલલ આઉટપુટ
PISO	પેરેલલ ઇનપુટ સીરિયલ આઉટપુટ
PIPO	પેરેલલ ઇનપુટ પેરેલલ આઉટપુટ
બિડાયરેક્શનલ	કોઈપણ દિશામાં શિફ્ટ કરી શકે છે
રિંગ કાઉન્ટર	છેલ્લા સ્ટેજનો આઉટપુટ પ્રથમ સ્ટેજને ફીડ કરાય છે
જોન્સન કાઉન્ટર	છેલ્લા સ્ટેજનું કોમ્પ્લિમેન્ટ પ્રથમ સ્ટેજને ફીડ કરાય છે

સીરિયલ-ઇન સીરિયલ-આઉટ (SISO) શિફ્ટ રજિસ્ટર:



ઓપરેશન:

1. ડેટા સીરિયલમાં બિટ દર બિટ ઇનપુટ મારફતે દાખલ થાય છે
2. દરેક ક્લોક પલ્સ સાથે, ડેટા એક સ્થાન જમણી તરફ શિફ્ટ થાય છે
3. 4 ક્લોક પલ્સ પછી, પ્રથમ ઇનપુટ બિટ આઉટપુટ પર દેખાય છે
4. ઉદાહરણ: "1101" ઇનપુટ માટે, સંપૂર્ણ ટ્રાન્સમિશન માટે 4 ક્લોક પલ્સની જરૂર પડે છે

- **મુખ્ય ઉપયોગ**: સીરિયલ અને પેરેલલ ફોર્મેટ વચ્ચે ડેટા રૂપાંતરણ
- **અનુપ્રયોગો**: કોમ્યુનિકેશન સિસ્ટમ્સ, ઉપકરણો વચ્ચે ડેટા ટ્રાન્સફર
- **ફાયદાઓ**: સરળ ડિઝાઇન, ન્યૂનતમ ઇન્ટરકનેક્શન્સ

મેમરી ટ્રીક: "શિફ્ટ રજિસ્ટર બકેટ બ્રિગેડની જેમ બિટ્સ પસાર કરે છે"

પ્રશ્ન 4(અ-OR) [3 માર્ક્સ]

4:2 ઓંકોડર દોરો અને સમજાવો.

જવાબ:

4:2 એન્કોડર ડાયાગ્રામ:

```
      |------|
D0--->|      |
      |      |---> A
D1--->|      |
      | 4:2  |
D2--->|      |
      |      |---> B
D3--->|      |
      |------|
```

ટ્રુથ ટેબલ:

D3	D2	D1	D0	B	A
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

લોજિકલ એક્સપ્રેશન્સ:

- A = D1 + D3
- B = D2 + D3
- **એન્કોડર ફંક્શન**: વન-હોટ ઇનપુટને બાઇનરી કોડમાં રૂપાંતરિત કરે છે
- **પ્રાયોરિટી એન્કોડર્સ**: પ્રાયોરિટી દ્વારા ઘણા સક્રિય ઇનપુટ્સને હેન્ડલ કરે છે
- **અનુપ્રયોગો**: કીબોર્ડ સ્કેનિંગ, ઇન્ટરપ્ટ હેન્ડલિંગ

મેમરી ટ્રીક: "એક સક્રિય લાઇન અંદર, બાઇનરી કોડ બહાર"

પ્રશ્ન 4(બ-OR) [4 માર્ક્સ]

ોહ્નન્સન કાઉન્ટર દોરો અને સમજાવો.

જવાબ:

જોન્સન કાઉન્ટર (4-બિટ):

```
    |------|    |------|    |------|    |------|
    |      |    |      |    |      |    |      |
    | D FF |----| D FF |----| D FF |----| D FF |
    |      |    |      |    |      |    |      |
    |------|    |------|    |------|    |------|
       ^           ^           ^           ^
       |           |           |           |
       |-----------|-----------|-----------|
                     CLK
                |
```

```
          Q3'      |
           |       |
           |       v
           ------->|
```

કાઉન્ટર સિક્વન્સ:

કાઉન્ટ	Q3	Q2	Q1	Q0
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
0	0	0	0	0

- **જોન્સન કાઉન્ટર**: ટ્વિસ્ટેડ રિંગ કાઉન્ટર તરીકે પણ ઓળખાય છે
- **સિક્વન્સ લંબાઈ**: 2n સ્ટેટ્સ જ્યાં n ફ્લિપ-ફ્લોપ્સની સંખ્યા છે
- **મુખ્ય વિશેષતા**: આસન્ન સ્ટેટ્સ વચ્ચે ફક્ત એક બિટ બદલાય છે

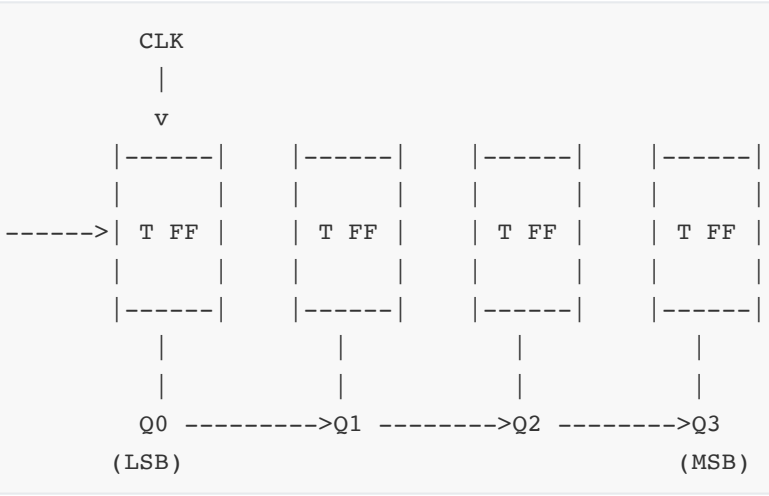
મેમરી ટ્રીક: "1 થી ભરો પછી 0 થી સાફ કરો"

પ્રશ્ન 4(ક-OR) [7 માર્ક્સ]

૪ બીટ જરપલ કાઉન્ટર દોરો અને સમજાવો.

જવાબ:

4-બિટ રિપલ કાઉન્ટર:



ટ્રુથ ટેબલ (કાઉન્ટિંગ સિક્વન્સ):

કાઉન્ટ	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
...	..	..	..	..
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0

કાર્ય સિદ્ધાંત:

1. બધા T ઇનપુટ્સ લોજિક 1 સાથે જોડાયેલા છે (ટોગલ મોડ)
2. પ્રથમ ફ્લિપ-ફ્લોપ દરેક ક્લોક પલ્સ પર ટોગલ થાય છે
3. દરેક પછીનું ફ્લિપ-ફ્લોપ ત્યારે ટોગલ થાય છે જ્યારે અગાઉનું 1 થી 0 માં બદલાય છે
4. દરેક સ્ટેજ સાથે પ્રોપેગેશન ડિલે વધે છે

- **અસિંક્રોનસ કાઉન્ટર**: ક્લોક ફક્ત પ્રથમ ફ્લિપ-ફ્લોપને ડ્રાઇવ કરે છે
- **રિપલ ઇફેક્ટ**: ફેરફારો સ્ટેજમાંથી પસાર થાય છે
- **ગેરલાભ**: સંચિત પ્રોપેગેશન ડિલેને કારણે ધીમું

મેમરી ટ્રીક: "પડતા ડોમિનોની જેમ ફેરફાર ફેલાય છે"

પ્રશ્ન 5(અ) [3 માર્ક્સ]

ટ ોંકમાોં DRAM સમજાવો.

જવાબ:

ડાયનેમિક રેન્ડમ એક્સેસ મેમરી (DRAM):

DRAM એક પ્રકારની સેમિકન્ડક્ટર મેમરી છે જે દરેક બિટને અલગ કેપેસિટરમાં સ્ટોર કરે છે.

મુખ્ય વિશેષતાઓ:

વિશેષતા	વર્ણન
સ્ટોરેજ એલિમેન્ટ	દરેક બિટ દીઠ સિંગલ કેપેસિટર + ટ્રાન્ઝિસ્ટર
ડેન્સિટી	ખૂબ ઊંચી (ચિપ દીઠ વધુ બિટ્સ)
સ્પીડ	મધ્યમ (SRAM કરતાં ધીમી)
રિફ્રેશ	સમયાંતરે જરૂરી (સામાન્ય રીતે દર થોડી મિલિસેકન્ડ)
પાવર વપરાશ	SRAM કરતાં ઓછો
કિંમત	SRAM કરતાં ઓછી ખર્ચાળ

- **ડાયનેમિક પ્રકૃતિ**: ચાર્જ સમય સાથે લીક થાય છે, રિફ્રેશની જરૂર પડે છે
- **અનુપ્રયોગો**: કમ્પ્યુટરમાં મુખ્ય મેમરી
- **ફાયદો**: ઉચ્ચ ડેન્સિટી, બિટ દીઠ ઓછી કિંમત

મેમરી ટ્રીક: "DRAM ને થાકેલા મન જેવી તાજગીની જરૂર પડે છે"

પ્રશ્ન 5(બ) [4 માર્ક્સ]

નીચેની વ્ યાખ્યા આપો (1)ફેન ઇન (2) પ્ર પોગેશન ડીલે

જવાબ:

ફેન-ઇન:

ફેન-ઇન એ લોજિક ગેટ સ્વીકારી શકે તેવા ઇનપુટની મહત્તમ સંખ્યા છે.

ફેન-ઇનની વિશેષતાઓ:

- ઇનપુટ લોડ ક્ષમતા માપે છે
- સર્કિટ જટિલતા અને ડિઝાઇનને અસર કરે છે
- ઉચ્ચ ફેન-ઇન ગેટની સંખ્યા ઘટાડે છે પરંતુ જટિલતા વધારે છે
- વિવિધ લોજિક ફેમિલીઓની વિવિધ ફેન-ઇન મર્યાદાઓ છે

ઉદાહરણ: એક સ્ટાન્ડર્ડ TTL NAND ગેટમાં સામાન્ય રીતે 8 ઇનપુટનો ફેન-ઇન હોય છે.

પ્રોપેગેશન ડિલે:

પ્રોપેગેશન ડિલે એ લોજિક ગેટના ઇનપુટથી આઉટપુટ સુધી સિગ્નલ પહોંચવામાં લાગતો સમય છે.

પ્રોપેગેશન ડિલેની વિશેષતાઓ:

- નેનોસેકન્ડ (ns)માં માપવામાં આવે છે

- હાઇ-સ્પીડ સર્કિટ પરફોર્મન્સ માટે મહત્વપૂર્ણ
- તાપમાન, લોડિંગ અને સપ્લાય વોલ્ટેજ સાથે બદલાય છે
- રાઇઝિંગ અને ફોલિંગ ટ્રાન્ઝિશન માટે અલગ છે

ઉદાહરણ: એક સામાન્ય TTL ગેટમાં 10-20 ns પ્રોપેગેશન ડિલે હોય છે.

- **સર્કિટ પર અસર**: મહત્તમ ઓપરેટિંગ ફ્રિક્વન્સી મર્યાદિત કરે છે
- **ગણતરી**: ઇનપુટ અને આઉટપુટ સિગ્નલના 50% પોઇન્ટ વચ્ચેનો સમય

મેમરી ટ્રીક: "ફેન-ઇન ઇનપુટ ગણે છે, પ્રોપ-ડિલે સમય ગણે છે"

પ્રશ્ન 5(ક) [7 માર્ક્સ]

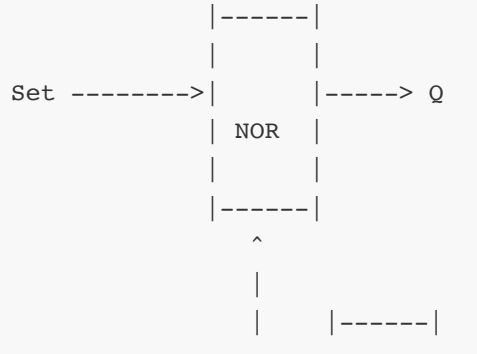
જનદેશ મુિબ કરો (i) લોજિક ફેમીલી TTL અને CMOS ની સરખામણી કરો.(ii) SR નો સકીટ ડાયાગ્રામ દોરો.

જવાબ:

(i) TTL અને CMOS લોજિક ફેમિલીની સરખામણી:

પેરામીટર	TTL	CMOS
ટેકનોલોજી	બાયપોલર ટ્રાન્ઝિસ્ટર્સ	MOSFETs
સપ્લાય વોલ્ટેજ	5V (ફિક્સ્ડ)	3-15V (ફ્લેક્સિબલ)
પાવર વપરાશ	ઉચ્ચ	ખૂબ નીચો (સ્ટેટિક)
સ્પીડ	મધ્યમથી ઉચ્ચ	નીચેથી ખૂબ ઉચ્ચ
નોઇઝ માર્જિન	મધ્યમ	ઉચ્ચ
ફેન-આઉટ	10-20	>50
પ્રોપેગેશન ડિલે	5-10 ns	10-100 ns (સ્ટાન્ડર્ડ)
ઇનપુટ ઇમ્પિડન્સ	4-40 kΩ	ખૂબ ઉચ્ચ (10^{12} Ω)
આઉટપુટ ઇમ્પિડન્સ	100-300 Ω	ચલ
સ્ટેટિક પ્રત્યે સંવેદનશીલતા	નીચી	ઉચ્ચ

(ii) SR ફ્લિપ-ફ્લોપ સર્કિટ ડાયાગ્રામ:



```
                |    |      |
                |----|      |-----> Q'
                     | NOR  |
                     |      |
Reset ------------->|      |
                     |------|
```

ટ્રુથ ટેબલ:

S	R	Q	Q'	રિમાર્ક્સ
0	0	Q	Q'	મેમરી (કોઈ ફેરફાર નહીં)
0	1	0	1	રીસેટ
1	0	1	0	સેટ
1	1	0	0	અમાન્ય (ટાળવું)

- **SR ફ્લિપ-ફ્લોપ**: ડિજિટલ સર્કિટમાં મૂળભૂત મેમરી એલિમેન્ટ
- **ઓપરેશન**: સેટ (S=1, R=0) Q=1 બનાવે છે; રીસેટ (S=0, R=1) Q=0 બનાવે છે
- **મેમરી સ્ટેટ**: જ્યારે S=0, R=0, આઉટપુટ અપરિવર્તિત રહે છે

મેમરી ટ્રીક: "SR: સેટ-રીસેટ, બંને નીચા હોય ત્યારે મેમરી"

પ્રશ્ન 5(અ-OR) [3 માર્ક્સ]

જડજિટલ જય્સના E વે્ટ પર ટ ોંકી નોંધ લખો.

જવાબ:

ડિજિટલ ચિપ્સનો E-વેસ્ટ:

ડિજિટલ ચિપ્સનો E-વેસ્ટ એ ત્યજી દેવાયેલા ઇલેક્ટ્રોનિક ઉપકરણોનો ઉલ્લેખ કરે છે જેમાં સેમિકન્ડક્ટર કોમ્પોનન્ટ્સ હોય છે જે ખાસ હેન્ડલિંગ અને નિકાલની જરૂર હોય છે.

મુખ્ય ચિંતાઓ:

પાસું	વિગતો
જોખમી સામગ્રી	લેડ, મર્ક્યુરી, કેડમિયમ, બ્રોમિનેટેડ ફ્લેમ રિટાર્ડન્ટ
પર્યાવરણીય અસર	યોગ્ય રીતે ફેંકવામાં ન આવે તો માટી અને પાણીનું પ્રદૂષણ
સંસાધન પુનઃપ્રાપ્તિ	કિંમતી ધાતુઓ ધરાવે છે (સોનું, ચાંદી, તાંબું)
જથ્થો	ટેકનોલોજિકલ પ્રગતિ સાથે ઝડપથી વધી રહ્યો છે
નિયમો	ઘણા દેશોમાં WEEE, RoHS દિશાનિર્દેશો દ્વારા સંચાલિત

મેનેજમેન્ટ અભિગમો:

- અધિકૃત ઇ-કચરા હેન્ડલર્સ દ્વારા રિસાયકલિંગ
- કિંમતી ધાતુઓની પુનઃપ્રાપ્તિ
- જોખમી ઘટકોનો સુરક્ષિત નિકાલ
- વિસ્તારિત ઉત્પાદક જવાબદારી કાર્યક્રમો
- **પડકારો**: અનૌપચારિક રિસાયકલિંગ આરોગ્ય જોખમો પેદા કરી રહ્યું છે
- **ઉકેલો**: ડિસએસેમ્બલી માટે ડિઝાઇન, ગ્રીન મેન્યુફેક્ચરિંગ

મેમરી ટ્રીક: "ડિજિટલ કચરાને ડિજિટલ-યુગના ઉકેલોની જરૂર છે"

પ્રશ્ન 5(બ-OR) [4 માર્ક્સ]

નીચેની વ્ યાખ્યા આપો (1) ફેન આઉટ (2)નોઇઝ માર્જીન

જવાબ:

ફેન-આઉટ:

ફેન-આઉટ એ એક લોજિક ગેટ આઉટપુટ દ્વારા ડ્રાઇવ કરી શકાતા ગેટ ઇનપુટની મહત્તમ સંખ્યા છે જે યોગ્ય લોજિક લેવલ જાળવી રાખે છે.

ફેન-આઉટની વિશેષતાઓ:

- આઉટપુટ ડ્રાઇવ ક્ષમતા માપે છે
- ડિઝાઇન ફ્લેક્સિબિલિટી અને કિંમતને અસર કરે છે
- ઉચ્ચ ફેન-આઉટ સરળ વાયરિંગ માટે પરવાનગી આપે છે
- કરંટ સોર્સિંગ/સિંકિંગ ક્ષમતા દ્વારા મર્યાદિત

ઉદાહરણ: એક સ્ટાન્ડર્ડ TTL ગેટમાં 10નો ફેન-આઉટ હોય છે, એટલે કે તે 10 સમાન ગેટ્સને ડ્રાઇવ કરી શકે છે.

નોઇઝ માર્જિન:

નોઇઝ માર્જિન એ નોઇઝ વોલ્ટેજની માત્રા છે જે ઇનપુટ સિગ્નલમાં ઉમેરી શકાય છે જેથી સર્કિટ આઉટપુટમાં અનિચ્છનીય ફેરફાર થવા ન પામે.

નોઇઝ માર્જિનની વિશેષતાઓ:

- વોલ્ટ્સમાં વ્યક્ત
- ઇલેક્ટ્રિકલ નોઇઝ સામે સર્કિટ ઇમ્યુનિટી માપે છે
- ઉચ્ચ નોઇઝ માર્જિનનો અર્થ વધુ વિશ્વસનીય ઓપરેશન
- હાઇ અને લો લોજિક લેવલ માટે અલગ

ઉદાહરણ: TTLમાં લોજિક લો માટે આશરે 0.4V અને લોજિક હાઇ માટે 0.7V નોઇઝ માર્જિન હોય છે.

- **ગણતરી**: ગેરંટેડ આઉટપુટ અને જરૂરી ઇનપુટ લેવલ વચ્ચેનો તફાવત
- **મહત્વ**: ઇલેક્ટ્રિકલી નોઇઝી વાતાવરણમાં મહત્વપૂર્ણ

મેમરી ટ્રીક: "ફેન-આઉટ આઉટપુટ ગણે છે, નોઇઝ માર્જિન દખલગીરી સામે લડે છે"

પ્રશ્ન 5(ક-OR) [7 માર્ક્સ]

જનદેશ મુિબ કરો (i) ROM મેમરી ઉપર ટુંક નોધ લખો ii) મા્ટર ્ લેવ JK જલલપ લલોપ સમજાવો.

જવાબ:

(i) ROM પર ટૂંક નોંધ:

ROM (રીડ-ઓન્લી મેમરી) એક નોન-વોલેટાઇલ મેમરી છે જેનો ઉપયોગ કાયમી અથવા અર્ધ-કાયમી ડેટા સ્ટોર કરવા માટે થાય છે.

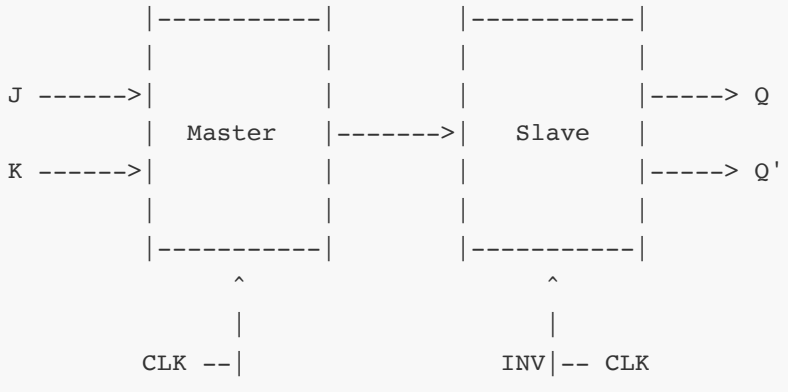
ROM ના પ્રકારો:

પ્રકાર	વિશેષતાઓ	પ્રોગ્રામિંગ
માસ્ક ROM	ફેક્ટરી પ્રોગ્રામ્ડ	ઉત્પાદન દરમિયાન
PROM	એક-વાર પ્રોગ્રામેબલ	યુઝર દ્વારા ઇલેક્ટ્રિકલ ફ્યુઝિંગ
EPROM	UV લાઇટ સાથે ભૂંસી શકાય	ઇલેક્ટ્રિકલ પ્રોગ્રામિંગ
EEPROM	ઇલેક્ટ્રિકલી ભૂંસી શકાય	ઇલેક્ટ્રિકલ પ્રોગ્રામિંગ/ભૂંસવું
ફ્લેશ ROM	ઝડપી ઇલેક્ટ્રિકલ ભૂંસવું	બ્લોક-વાઇઝ ભૂંસવું/લખવું

અનુપ્રયોગો:

- ફર્મવેર અને BIOS સ્ટોરેજ
- ફિક્સ્ડ ફંક્શન્સ માટે લુક-અપ ટેબલ્સ
- પ્રોસેસરમાં માઇક્રોકોડ
- કમ્પ્યુટરમાં બૂટ કોડ
- **ડેટા રિટેન્શન**: પાવર વગર ડેટા જાળવી રાખે છે
- **એક્સેસ ટાઇમ**: સામાન્ય રીતે 45-150 ns
- **ડેન્સિટી**: ઉચ્ચ સ્ટોરેજ ક્ષમતા

(ii) JK માસ્ટર-સ્લેવ ફ્લિપ-ફ્લોપ:



ટ્રુથ ટેબલ:

J	K	Q(next)	ફંક્શન
0	0	Q	કોઈ ફેરફાર નહીં
0	1	0	રીસેટ
1	0	1	સેટ
1	1	Q'	ટોગલ

ઓપરેશન:

1. **માસ્ટર સ્ટેજ**: જ્યારે CLK=1, માસ્ટર લેચ J અને K ઇનપુટ્સને સેમ્પલ કરે છે
2. **સ્લેવ સ્ટેજ**: જ્યારે CLK=0, સ્લેવ લેચ માસ્ટર આઉટપુટને સેમ્પલ કરે છે
3. **ટુ-ફેઝ ઓપરેશન**: રેસ કન્ડિશન અટકાવે છે (ફેરફારો ફક્ત ક્લોક એજ પર થાય છે)
4. **ફાયદો**: SR ફ્લિપ-ફ્લોપ કરતાં વધુ બહુમુખી (કોઈ અમાન્ય સ્થિતિ નથી)

- **ટોગલ મોડ**: જ્યારે J=K=1, આઉટપુટ દરેક ક્લોક સાયકલમાં ટોગલ થાય છે
- **અનુપ્રયોગો**: કાઉન્ટર્સ, શિફ્ટ રજિસ્ટર્સ, સિક્વેન્શિયલ સર્કિટ્સ

મેમરી ટ્રીક: "J-K: સેટ-રીસેટ-ટોગલ, માસ્ટર આગળ ચાલે સ્લેવ અનુસરે છે"