**Conversions ( Base = R)**

Convert from foreign base to familiar base ( V(x) = )

(14.02)5 to R = 10 🡪 4 x 50 + 1 x 51 + 0 x 5-1 + 2 x 5-2 = 4 + 5 + 0 + 2/25 = 9.08

Convert from familiar base to foreign base ( V(x) = ) where V(x) is known but x is unknown

(125)10 to R = 8 🡪 125/8 = 15 rem = **5** 🡪 15/8 = 1 rem = **7** 🡪 1/8 = 0 rem = **1** so V(x) = 125 and x = 175 (125)10 = (175)8

**Finding unknown:** (211)x = (152)8 🡪 (211)x = (1\*82) + (5 \* 81) + (2 \* 80) 🡪 (2 \* x) + (1 \* x) + (1 \* x) = (106)10 🡪 2x2 + x + 1 = (106)10 🡪plug in 7 to get (211)7 = (152)8

**Binary to Octal:** (1101010.1011011)2 to R = 8 🡪 001 | 101 | 010 | .101 | 101 | 100 🡪 152.554 (1101010.1011011)2 = (152.554)8

**Binary to Hex:** (11101.101011)2 to R = 16🡪 | 0001 | 1101 | .1010 | 1100 🡪 1D.AC (11101.101011)2 = (1D.AC)16

**Binary to Decimal:** 1110101.101 to R = 10 🡪 1 x 20 + 0 x 21 + 1 x 22 + 0 x 23 + 1 x 24 + 1 x 25 + 1 x 26 + 1 x 2-1 + 0 x 2-2 + 1 x 2-3 = (1110101.101)2 = (117.625)10

**Octal to binary:** (35)8 to R = 2 🡪 1st convert octal to decimal 🡪 3 x 81 + 5 x 80 = 29 🡪 2nd convert dec. to binary🡪 29/2 = 14 rem = 1 🡪 14/2 = 7 rem 0 ….. 11101

**Octal to Hex:** (35)8 to R = 16 🡪 1st convert octal to decimal 🡪 (35)8 = (29)10 🡪 convert decimal to binary 🡪 (29)10 = (11101)2 🡪 0001|1101 = 1D 🡪 (1D)16 = (35)8

**Octal to Decimal:** (35)8 to R = 10 🡪 3 x 81 + 5 x 80 = 29 (35)8 = (29)10

**Decimal to Binary:** (35)8 to R = 2 🡪 1st convert to decimal 🡪 🡪 (35)8 = (29)10 🡪 (29)10 = 29/2 = 14 rem = 1 🡪 14/2 = 7 rem = 0 🡪 7/2 = 3 rem 1🡪 3/2 = 1 rem=1🡪1/2 = 1 rem 1🡪 (35)8 = (11101)2

**Decimal to Octal:** (1863)10 to R = 8 🡪 1863/8 = 232 rem = 7 🡪 232/8 = 29 rem = 0 🡪 29/8 = 3 rem = 5 🡪 3/8 = 0 rem = 3 (1863)10 = (3507)8

(0.125)10 to R = 8 🡪 (0.125) x 8 = 1.0 (0.125)10 = (1.0)8

**Decimal to Hex:** (253)10 to R = 16 🡪 253/16 = 15 rem =13(D) 🡪 15/16 = 0 rem 15(F) (253)10 = (FD)16

(0.63671875)10 to R = 16🡪 (0.63671875) x 16 = 10.1875 🡪 10 = A 🡪 (0.1875) x 16 = 3.0 = 3 (0.63671875)10 = (0.A3)16

**Hex to Decimal:** A67E = 14 + 7(16) + 6(16)2 + 10(16)3

(1D.AC)16 to R = 10 🡪 1 x 16 + 13 + 10/16 + 12/162 = 29.672 (1D.AC)16 = (29.672)10

**Hex to Binary:** 1 = 0001 2 = 0010 3 = 0011 4 = 0100….. (A67E)16 = (1010011001111110)2 🡪 1010|0110|0111|1111|0

**Hex to Octal:** (A67E)16 to R = 8 🡪 1st convert to binary 🡪 001|010|011|001|111|110 🡪 (A67E)16 = (123176)8

**Binary Coded Decimal (BCD):** 236 = (0010 0011 0110)BCD

**Complements:** 9’s comp: 123456 = 876543 🡪 subtract 9 from each number. 10’s comp: add 1 to 9’s comp🡪123456 = 876544. 2’s comp🡪10111010 = 01000110

**Boolean expressions:**

Properties of 1 and 0: 1.) X + 0 = X 2.) X + 1 = 1 3.) X \* 0 = 0 4.) X \* 1 = X 5.) X + X’ = 1 6.) X \* X’ = 0

Commutative Rule: 1.) x + y = y + x 2.) x\*y = y\*x

Mealy Machine Sequence Detector

Associative Rule: 1.) x + (y + z) = (x + y) + z 2.) x(y\*z) = (x\*y)z

Distributive Rule: 1.) x + (y\*z) = (x + y)(x + z) 2.) x(y + z) = (x\*y) + (x\*z)

Idempotency: 1.) x + x = x 2.) x\*x = x

Absorption: 1.) x + x\*y = x 2.) x + x’ \* y = x + y 3.) x(x + y) = x 4.) x(x’ + y) = x\*y

De Morgan’s Law: 1.) (x + y)’ = x’ \* y’ 2.) (x \* y)’ = x’ + y’

**Sum of Products (SOP):** Follow the 1’s

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Present State | | | Next State | | Output |
| I | Q1 | Q0 | D1 | D0 | F |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I D1 Q1Q0** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 1 | 1 |

D1 = IQ1’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I D2 Q1Q0** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 0 | 0 |
| **1** | 1 | 1 | 0 | 0 |

D0 = I(Q0 + Q1) F = IQ1

A | B | Cin | Sum AB'Cin'+ A'B'Cin + ABCin + A'BCin'

0 0 0 0 Cin’(ab′+a′b)+Cin(a′b′+ab)

0 0 1 1 Cin’(A ⊕ B) + Cin(a ⊕ b)’

0 1 0 1 A ⊕ B ⊕ Cin

0 1 1 0

1 0 0 1

1 0 1 0

1 1 0 0

1 1 1 1

**Product of Sums (POS):** Follow the 0’s (a’b + ab’+ a’c’)

SOP =

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | c | a’b | ab’ | a’c’ | F |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

POS =

**K-Maps:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **00** | **01** | **11** | **10** |
| **0** | M0 | M2 | M6 | M4 |
| **1** | M1 | M3 | M7 | M5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **00** | **01** | **11** | **10** |
| **00** | **M0** | **M4** | **M12** | **M8** |
| **01** | **M1** | **M5** | **M13** | **M9** |
| **11** | **M3** | **M7** | **M15** | **M11** |
| **10** | **M2** | **M6** | **M14** | **M10** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Min Terms | x | y | z | F |
| M0 | 0 | 0 | 0 | 0 |
| M1 | 0 | 0 | 1 | 0 |
| M2 | 0 | 1 | 0 | 1 |
| M3 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 0 | 0 |
| M5 | 1 | 0 | 1 | 1 |
| M6 | 1 | 1 | 0 | 0 |
| M7 | 1 | 1 | 1 | 1 |

Solve with Boolean algebra

F = x’yz’ + x’yz + xy’z + xyz Using K-maps to solve

F = x’y(z’ + z) + xz (y’ + y) x’yz’ = M2 x’yz = M3 xy’z = M5 xyz = M7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| z xy | **00** | **01** | **11** | **10** |
| **0** | 0 | 1 | 0 | 0 |
| **1** | 0 | 1 | 1 | 1 |

F = x’y + xz

x ⊕ 0 = x x ⊕ 1 = x’ x ⊕ x = 0 x ⊕ ( y ⊕ z) = (x ⊕ y) ⊕ z

x ⊕ y = y ⊕ x

**Logic Circuits** NOR a’b’ = (a + b)’ NAND(ab)’ = a’ + b’ XOR(ab’ + a’b) = a ⊕ b XNOR(a ⊕ b)’ 🡪 a’b’ + ab

|  |  |  |
| --- | --- | --- |
| a | b | C |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| a | b | C |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| a | b | C |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| a | b | C |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| Decimal | Excess Decimal | Binary |
| 0 | 3 | 0011 |
| 1 | 4 | 0100 |
| 2 | 5 | 0101 |

**Excess 3 codes**: Whatever Decimal, add three and convert to binary

**Gray Code**: Adjacent code words only differ by 1-bit

Step 1: 0 🡪 step 2: 00 step 3: 000

1 01 001 🡪 Step 1 : Start with zero and differ by 1 bit (1-bit gray code)

11 011 🡪 Step 2: Pad upper right with 0’s. Pad lower right with 1’s and mirror the rest (2-bit gray code)

10 010 🡪 Step 3: Pad upper right with 0’s. Pad lower right with 1’s and mirror the rest (3-bit gray code)

110

111

101

100

**Decoders:** (N inputs = 2N outputs)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I2 | I1 | I0 | Q7 | Q6 | Q5 | Q4 | Q3 | Q2 | Q1 | Q0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Q0 = I2’I1’I0 ‘ Q1 = I2’I1’I0 Q2 = I2’I1I0’ Q3 = I2’I1I0 Q4 = I2I1I0’ Q5 = I2I1’I0 Q6 = I2I1I0’ Q7 = I2I1I0

I2

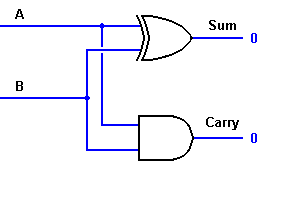
I1

I0

**Encoders:** Inverse of decoders (N inputs = 2N outputs)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | A | B | C |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

D7 = ABC D6 = ABC’ D5 = AB’C D4 = AB’C’ D3 = A’BC D2 = A’BC’ D1 = A’B’C D0 = A’B’C’

**Half Adder (HA):** Two inputs a,b. Two outputs carry, sum

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | CH | Sh |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

CH = ab

HA

SH = a’ \* b + a \* b’ 🡪 a ⊕ b

**Full Adder(FA):** Three inputs, a, b, c, and two outputs carry, sum

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | CF | SF |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Cf = a’bc + ab’c + abc’ + abc

HA

Cf = c(a’b + ab’) + ab(c’ + c)

Cf = c(a ⊕ b) + ab

Cf = SH \* c + CH

SF = a’b’c + a’bc’ + ab’c’ + abc

SF = a’(b’c + bc’) + a(b’c’ + bc)

SF = a’(b ⊕ c) + a(b ⊕ c)’ 🡨 XNOR

SF = (a ⊕ b) ⊕ c

SF = SH ⊕ c

HA

**Half Subtractor(HS):** Two inputs x,y. Two outputs D, B

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | BH | Dh |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

B = x’y

HS

D = x’y + x’y

D = x ⊕ y

Let a = x’ and b = y

B = ab

D = ab + a’b’

D = (a ⊕ b)’ 🡨XNOR

**Full Subtractor(FS):** Three inputs x, y, z. Two outputs D, B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| z xy | **00** | **01** | **11** | **10** |
| **0** | 0 | 1 | 0 | 1 |
| **1** | 1 | 0 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| z xy | **00** | **01** | **11** | **10** |
| **0** | 0 | 1 | 0 | 0 |
| **1** | 1 | 1 | 1 | 0 |

D = x’y’z + x’yz’ + xy’z’ + xyz

Let a = x’ b = y c = z

D = abc’ + a’b’c’ + ab’c + a’bc

D = a(bc’ + b’c) + a’(b’c’ + bc)

D = a(b ⊕ c) + a’(b ⊕ c)’

D = ( a ⊕ b ⊕ c)’ = FA

B = x’y’z + x’yz’ + x’yz + xyz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | y | z | 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 |

B = x’y + x’z + yz

Let a = x’ b = y c = z

B = ab + ac + bc

B = Carry of FA

FA

Moore Machine

**Binary Sequence Detectors:** Detect sequence 111 (input = 1011110101)

States

0 = (00)

1 = (01)

11 = (11)

111 = (10)

0

1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Present State** | | **If input = 0** | | **If input = 1** | |
| **Q1** | **Q0(t)** | **Q1** | **Q1** | **Q1** | **Q0** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
|  |  | **D1** | **D0** | **D1** | **D0** |

11

111

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I D0 Q1Q0** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 0 | 0 |
| **1** | 1 | 1 | 0 | 0 |

D1 Q1

Q1'

IQ1’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I D1 Q1Q0** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 1 | 1 |

F

I(Q0 + Q1)

D0 Q0

Q0'

|  |  |  |
| --- | --- | --- |
| **Q0 F Q1** | **0** | **1**  Q1Q0’ |
| **0** | **1** | **0** |
| **1** | **0** | **1** |

Remember to 2’s comp 2nd number

s

**Carry Propagation:** When M = 0 you add and when M = 1 you subtract

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | Carry | Sum |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | M | A | B |
| a.) | 0 | 0111 | 0110 |
| b.) | 0 | 1000 | 1001 |
| c.) | 1 | 1100 | 1000 |
| d.) | 1 | 0101 | 1010 |
| e.) | 1 | 0000 | 0001 |

Case A(ADD) Case B(ADD) Case C(SUBTRACT) Case D (SUBTRACT) Case E (SUBTRACT)

0111 1000 1100 0101 0000

0110 1001 1000 1010 0001

1101 0001 0100 1011 1111

A.) C4 = 0 C3 = 1 B.) C4 = 1 C3 = 0 C.) C4 = 1 C3 = 1 D.) C4 = 0 C3 = 1 E.) C4 = 0 C3 = 0

(0 ⊕ 1) = 1 (1 ⊕ 0) = 1 (1 ⊕ 1) = 0 (0 ⊕ 1) = 1 (0 ⊕ 0) = 0

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sum | Carry C | Overflow V |
| a.) | 1101 | 0 | 1 |
| b.) | 0001 | 1 | 1 |
| c.) | 0100 | 1 | 0 |
| d.) | 1011 | 0 | 1 |
| e.) | 1111 | 0 | 0 |

D2 Q2

Clk Q2’

**Flip Flops (SR, D, JK, T):** (Set/Reset)(Data)(Jack Kilby)(Toggle)

D1 Q1

Clk Q1’

DFF: Design a synchronous counter that counts in the following way: 000, 010, 011, 101, 110, 000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D2 | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | 0 | x |
| **1** | x | 1 | x | 1  D0 Q0  Clk Q0’ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q2 | Q1 | Q0 | D2 | D1 | D0 |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | x | x | x |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | x | x | x |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | x | x | x |

Q2Q1 Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D1 | **00** | **01** | **11** | **10** |
| **0** | 1 | 1 | 0 | x |
| **1** | x | 0 | x | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D0 | **00** | **01** | **11** | **10** |
| **0** | 0 | 1 | 0 | x |
| **1** | X | 1 | x | 0 |

**D2 =** Q0 D1 = Q2’Q0’ + Q1’ D0 = Q2’Q0

Moore Machine only depends on current state

**JKFF:** A synchronous counter is built using JK flip-flops Q2, Q1, and Q0. It counts the following way: 000, 010, 011, 101, 110, 000… What are the equations for J2, J1, K1, and K0?

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Current State | | | Next State | | |
| J2 | K2 | J1 | K1 | J0 | K0 | Q0 | Q1 | Q2 | Q2 | Q1 | Q0 |
| 0 | x | 1 | x | 0 | x | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | x | x | 0 | 1 | x | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | x | x | 1 | x | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| x | 0 | 1 | x | x | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| x | 1 | x | 1 | 0 | x | 1 | 1 | 0 | 0 | 0 | 0 |

SR Latch

NAND

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **F** |
| **0** | **0** | **0** | **1** |
| **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **0** |

JK

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **J** | **K** |
| **0** | **0** | **0** | **x** |
| **0** | **1** | **1** | **x** |
| **1** | **0** | **x** | **1** |
| **1** | **1** | **x** | **0** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q0 J2 Q2Q1** | **00** | **01** | **11** | **10** |
| **0** | 0 | 0 | x | x |
| **1** | x | 1 | x | x |

All Moore Machine since next state is only dependant on the current state.

Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q0 J1 Q2Q1** | **00** | **01** | **11** | **10** |
| **0** | 1 | x | x | x |
| **1** | x | x | x | 1 |

1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q0 K1 Q2Q1** | **00** | **01** | **11** | **10** |
| **0** | x | 0 | 1 | x |
| **1** | x | 1 | x | x |

Q2 + Q0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Q0 K0 Q2Q1** | **00** | **01** | **11** | **10** |
| **0** | x | x | x | x |
| **1** | x | 0 | X | 1 |

Q2

**Combinational Circuits:** A set of gates with inputs and outputs

**AND-OR Circuit (SOP)** Use k-maps to solve for final answer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Min Terms | x | y | z | F |
| M0 | 0 | 0 | 0 | 0 |
| M1 | 0 | 0 | 1 | 0 |
| M2 | 0 | 1 | 0 | 1 |
| M3 | 0 | 1 | 1 | 1 |
| M4 | 1 | 0 | 0 | 0 |
| M5 | 1 | 0 | 1 | 1 |
| M6 | 1 | 1 | 0 | 0 |
| M7 | 1 | 1 | 1 | 1 |

F = x’yz’ + x’yz + xy’z +xyz

F = M2 + M3 + M5 + M7

M2 + M3 = x’y(z’ + z)

M5 + M7 = xz(y’ + y)

F = x’y + xz

F =

**OR-AND Circuit (POS)** 🡪 Same truth table as SOP (Use K-map but flip every “not” on final answer 🡪 x’yz’ = xy’z)

|  |  |
| --- | --- |
| Check Position |  |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |
| 13 | 1101 |
| 14 | 1110 |
| 15 | 1111 |

F = (x + y + z)( x + y + z’)(x’ + y + z)(x’ + y’ + z)

F = M0 + M1 + M4 + M6

(xyz) + (xyz’) + (x’yz) + (x’y’z)

M0 + M1 = xy

M4 + M6 = x’z

F = (x + y)(x’ + z)

(0, 1, 4, 6)

**Error Correction (Hamming Code):** (M + R + 1) Where M = message bits, R = Check bits

1st parity check: **1** 3 5 7 9 11 13 15 **Message = 1011** 🡪 Positions: 1 2 3 4 5 6 7

2nd parity check: **2** 3 6 7 10 11 14 15 M + R + 1 = 4 + 3 + 1 = 8 \_ \_ 1 \_ 0 1 1

3rd parity check: **4** 5 6 7 12 13 14 15 Error check Parity 1 = 1 3 5 7 = 1 ⊕ 0 ⊕ 1 = **0**

4th parity check: **8** 9 10 11 12 13 14 15 add RC + Positions Parity 2 = 2 3 6 7 = 1 ⊕ 1 ⊕ 1 = **1**

Positions 1 2 3 4 5 6 7 Parity 3 = 4 5 6 7 = 0 ⊕ 1 ⊕ 1 = **0**

0 1 0 0 01 1 Received codeword (RC): 0110011

Error Check = 0100011

Parity Error Check 1: P1 + P3 + P5 + P7 = 0 ⊕ 0 ⊕ 0 ⊕ 1 = **1** (failed)

Parity Error Check 2: P2 + P3 + P6 + P7 = 1 ⊕ 0 ⊕ 1 ⊕ 1 = **1** (failed)

Parity Error Check 3: P4 + P5 + P6 + P7 = 0 ⊕ 0 ⊕ 1 ⊕ 1 = **0** (success)

Error has occurred at 011 (Position 3) 🡪 ~~0~~ ~~1~~ 1 ~~0~~ 0 1 1

Extracted codeword = 1011