darshil solanki

November 11, 2023

computer organization

assignment

Certificate

This is to certify that DARSHILKUMAR MANSUKHBHAI SOLANKI of MCA SEM-1 with PRN NO: 8023057262 has successfully completed the COMPUTER ORGANIZATION AND ARCHITECTURE lab work during academic year 2023-24.

Date of Completion: 04/11/2023

Signature of HOD Signature of Professor

Table of Contents

**Q1 Logic gate3**

AND3

OR3

NOT3

NAND4

NOR4

XOR5

XNOR5

**Q2 Half Adders6**

**Q3 Universal gates8**

Implementation of all gates using NOR8

Implementation of all gates using NAND10

**Q4 Boolean Expression13**

**Q5 Code conversion20**

Binary to GCD20

GCD to Binary22

BCD to GCD24

GCD to BCD26

BCD to EX-329

EX-3 to BCD31

**Q6 Full Adders33**

**Q7 Subtractors34**

**Q8 Micro Programming35**

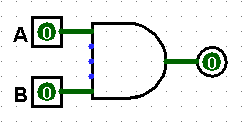
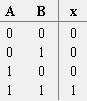
**Q9 8X1 MULTIPLEXER36**

1. To study and analyze logic gates using Logisim Simulator.

### AND Gate

A simple AND gate has two inputs, A and B, and one output, Y. All inputs to an AND gate must be 1 for the output to be 1. If both A and B are 1 then the output is 1. Otherwise, the output is 0.

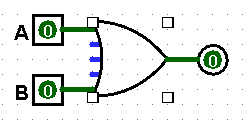
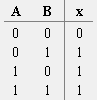
Output is TRUE only if both inputs are TRUE.

### OR Gate

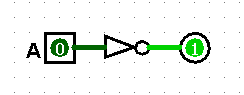
A basic OR gate has two inputs, A and B, and an output Y. The output is 1 if either A or B, or both, are 1, and 0 only when both A and B are 0.

Output is TRUE if either input (or both) is TRUE.

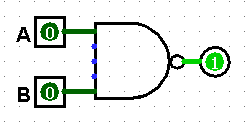
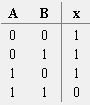
### NOT or Inverter Gate

It produces the NOT, or complement function. If input is 0 output is 1, if input is 1 output is 0.

**** 

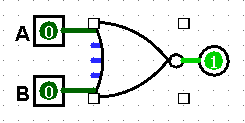
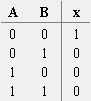
### NAND Gate

The NAND gate is derived from the abbreviation of NOT-AND. The NAND function is the complement of the AND function, is indicated by the symbol, which consists of an AND gate followed by small circle. Output of NAND GATE is inverse of AND GATE.

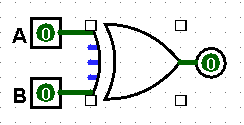
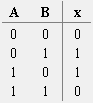
### NOR Gate

The NOR gate is derived from the abbreviation of NOT-OR. The NOR function is the complement of the OR function, is indicated by the symbol, which consists of an OR gate followed by small circle. Output of NOR GATE is inverse of ORGATE.

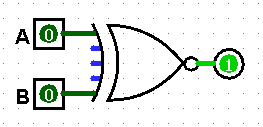
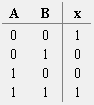
### XOR GATE

The exclusive –OR gate has a graphic symbol similar to the OR gate except for additional curved lined on the input side. The output of this gate is 1 if any input is 1 but excludes the combination when both inputs are1.

### XNOR GATE

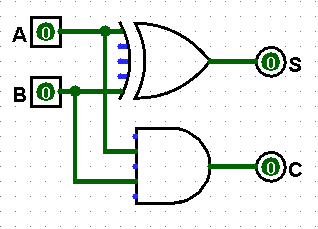
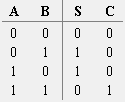
The exclusive –NOR is the complement of the exclusive –OR, as indicated by the small circle in the graphic symbol of XOR. The output of this gate is 1 only if both inputs are equal to 1 or 0.

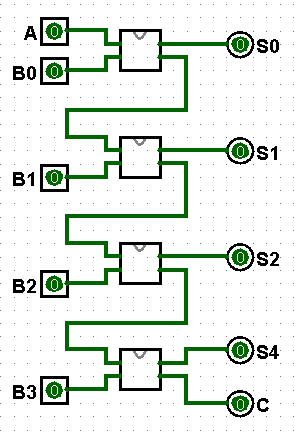
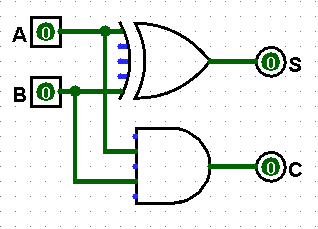
A) To study, design and simulate Half-ADDER Circuit.

### Half adder

### A half adder has two inputs, generally labeled A and B, and two outputs, the [sum](http://en.wikipedia.org/wiki/Sum) S and [carry](http://en.wikipedia.org/wiki/Carry) C. S is the two-bit [XOR](http://en.wikipedia.org/wiki/XOR_gate) of A and B, and C is the [AND](http://en.wikipedia.org/wiki/AND_gate) of A and B. Essentially the output of a half adder is the sum of two one-bit numbers, with C being the most significant of these two outputs.

B) To design 4-bit magnitude adder using half - ADDER circuit.

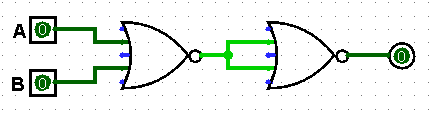
4 bit magnitude adder half-adder

1. Study and implementation of Universal gates.

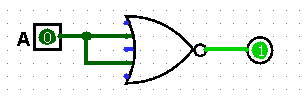
A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

**Implementation of all gates using NOR.**

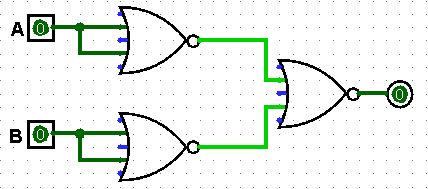
**Or**



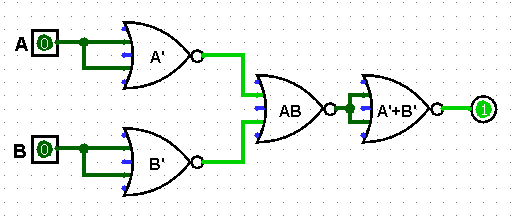
**Not**



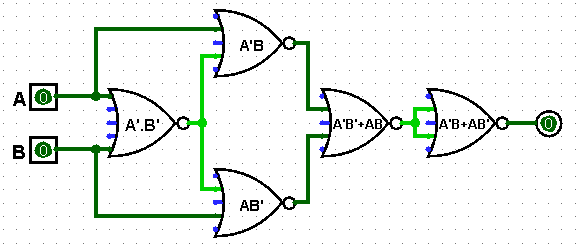
**And**



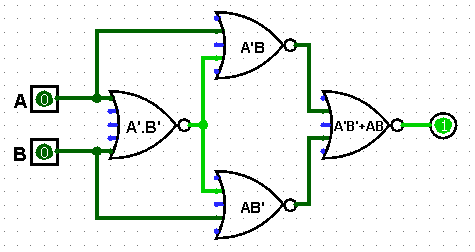
**NAND**



**XOR**

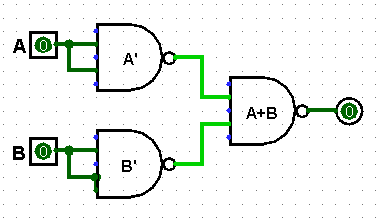
****

**XNOR**

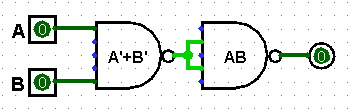


**Implementation of all gates using NAND.**

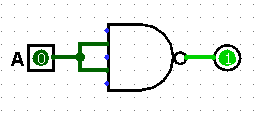
**Or**

****

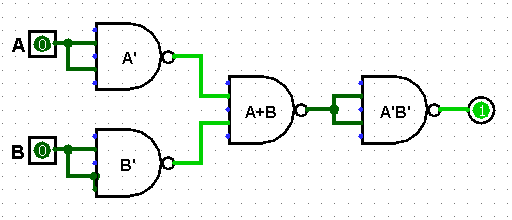
**And**

****

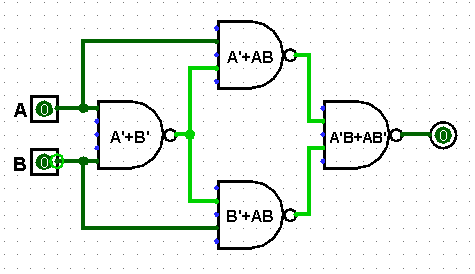
**Not**

****

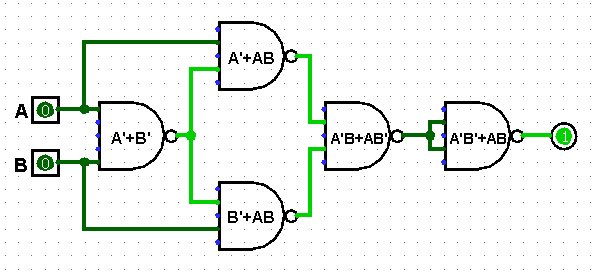
**NOR**

****

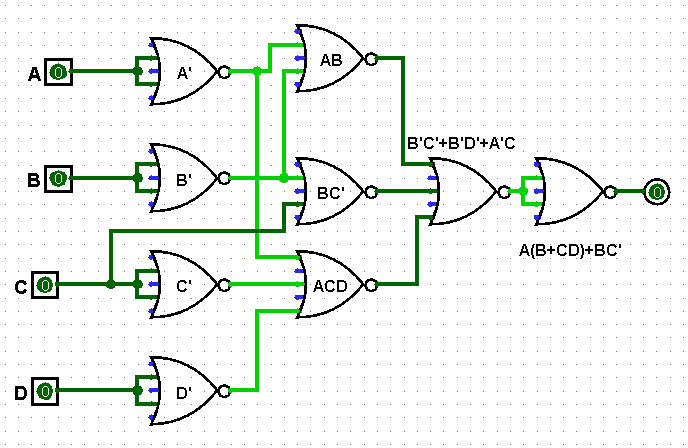
**XOR**

****

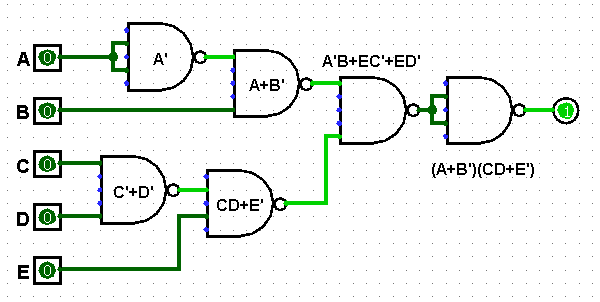
**XNOR**

****

1. Implementation of Boolean expressions and K-maps
2. Implement the following function using only NOR gates F = a(b+cd) + bc’.



1. Implement the following function using only NAND gates G = (a+b’)(cd +e’).



1. Express the following functions as sum of minterms and product of maxterms:
2. F(A, B, C, D) = B’D + A’D + BD

=>B’D(C+C’)+A’D(B+B’)+BD(A+A’)

=>B’DC+B’DC’+A’DB+A’DB’+ABD+A’BD

=>(B’CD+B’C’D)(A+A’)+(A’BD+A’B’D+ABD+A’BD)(C+C’)

=>AB’CD+A’B’CD+AB’C’D+A’B’C’D+A’BCD+A’B’CD+ABCD+A’BCD+A’BC’D+A’B’C’D+ABC’D+A’BC’D

=>AB’CD+A’B’CD+AB’C’D+A’B’C’D+A’BCD+ABCD+A’BC’D+ABC’D

or

=>(A’+B+C’+D’)(A+B+C’+D’)(A’+B+C+D’)(A+B+C+D’)(A+B’+C’+D’) (A’+B’+C’+D’)(A+B’+C+D’)(A’+B’+C+D’)

1. F (x, y, z) = (xy +z) (xz+y)

=>XYXZ+XYY+XZZ+ZY

=>XYZ+XY+XZ+ZY

=>XYZ+XY(Z+Z’)+XZ(Y+Y’)+ZY(X+X’)

=> XYZ+XYZ+XYZ’+XZY+XZY’+XYZ+X’ZY

=> XYZ+XYZ’+XY’Z+X’ZY

Or

=>(X’+Y’+Z’)(X’+Y’+Z)(X’+Y+Z’)(X+Y’+Z’)

1. Reduce the following Boolean expressions using axioms and laws:
2. AB + (AC)’ + AB’C(AB +C)

=>AB+(AC)’+AB’CAB+AB’CC

=>AB+(AC)’+0+AB’C

=>AB+A’+C’+AB’C

=>A(B+B’C)+A’+C’

=>A(B+C)+A’+C’

=>AB+AC+A’+C’

=>A’+B+C’+A

=>A+A’+B+C’

=>1+B+C’

=>1

1. (WX + WY’)(X+W)+WX(X’+Y’)

=> WXX+WXW+WY’+WY’X+WXX’+WXY’

=>WX+WXY’+WY’

=>WXY’+WY’

=>WY’X

1. (A’ +C)(A’+C’)(A’+B+C’D)

=>((A’+C)’+(A’+C’)’+(A’+B+C’D)’)’

=>(AC’+AC+AB’(C’D)’)’

=>(A(C+C’) +AB’(C+D’))’

=>(A +AB’C+AB’D’)’

=>(A(1+B’C+B’D’))’

=>(A.1)’

=>A’

1. X[Y+Z (XY+XZ)’]

=> XY+XZ(XY+XZ)’

=> XY+XZ[ (XY)’(XZ)’ ]

=>XY+XZ( (X’+Y’)(X’+Z’) )

=>XY+XZ(X’+ X’Z’+X’Y’+Y’Z’)   
=>XY+XZX’+X’Z’XZ+X’Y’XZ+Y’Z’XZ

=>XY+0+0+0+0

=>XY

1. Prove the following:
   * 1. ((AB)’ +A’ +AB)’ =0

=>(AB)(A)(AB)’

=>ABA(AB)’

=>AB(AB)’

=>0

* + 1. AB + (AC)’ +AB’C(AB +C) =1

=>AB+(AC)’+AB’CAB+AB’CC

=>AB+(AC)’+0+AB’C

=>AB+A’+C’+AB’C

=>A(B+B’C)+A’+C’

=>A(B+C)+A’+C’

=>AB+AC+A’+C’

=>A’+B+C’+A

=>A+A’+B+C’

=>1+B+C’

=>1

* + 1. (AB’ +AC’)(BC+BC’)(ABC)=0

=> (A(B’+C’))(B(C+C’))(ABC)

=> (A(B’+C’))B(ABC)

=>ABC(B’+C’)

=>ABCB’+ABCC’

=>0+0

=>0

1. Reduce the following expressions using K-MAP
   * 1. ABC’ + A’BC + ABCD + ABD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  | 1 | 1 |
| 11 | 1 | 1 | 1 |  |
| 10 |  |  |  |  |

=>ABC’+A’BC+BCD

* + 1. AB + AB’C + A’BC’ + BC’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bc→  a↓ | 00 | 01 | 11 | 10 |
| 0 |  |  |  | 1 |
| 1 |  | 1 | 1 | 1 |

=>AC+BC’

* + 1. ∑m(5, 6, 7, 9, 10, 11, 13, 14, 15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 15 | 17 | 16 |
| 11 | 12 | 113 | 115 | 114 |
| 10 | 8 | 19 | 111 | 110 |

=>AC+AC’D+A’BC+A’BC’D or => BD+AB’D+BCD’+AB’CD’

* + 1. ∑m(0, 1, 4, 5, 6, 7, 9, 11, 15) + d(10, 14)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 10 | 11 | 3 | 2 |
| 01 | 14 | 15 | 17 | 16 |
| 11 | 12 | 13 | 115 | X14 |
| 10 | 8 | 19 | 111 | X10 |

=>A’C’+AB’D+BC

* + 1. ∏M(3, 6, 8, 11, 13, 14) d(1, 5, 7, 10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | X1 | 03 | 2 |
| 01 | 4 | X5 | X7 | 06 |
| 11 | 12 | 013 | 15 | 014 |
| 10 | 08 | 9 | 011 | X10 |

=> (B+C’+D’)(B’+C’+D)(A’+B’+C+D’)(A’+B+C+D)

Or

=>(B’CD+BCD’+ABC’D+AB’C’D’)’

1. Implement following conversions.

|  |  |  |  |
| --- | --- | --- | --- |
| **GCD** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |

* 1. Binary to GCD

|  |  |  |  |
| --- | --- | --- | --- |
| **Binary** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 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 |
|  |  |  |  |

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 14 | 15 | 17 | 16 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 18 | 19 | 111 | 110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 112 | 113 | 115 | 114 |
| 10 | 18 | 19 | 111 | 110 |

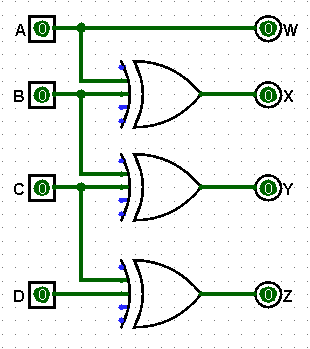
W=>A X=>AB’+A’B

K-map for Y K-map for Z

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 13 | 12 |
| 01 | 14 | 15 | 7 | 6 |
| 11 | 112 | 113 | 15 | 14 |
| 10 | 8 | 9 | 111 | 110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 11 | 3 | 12 |
| 01 | 4 | 15 | 7 | 16 |
| 11 | 12 | 113 | 15 | 114 |
| 10 | 8 | 19 | 11 | 110 |

Y=>BC’+B’C Z=>C’D+CD’



Binary to GCD

* 1. GCD to Binary

|  |  |  |  |
| --- | --- | --- | --- |
| **GCD** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Binary** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 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 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 14 | 15 | 17 | 16 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 18 | 19 | 111 | 110 |

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 112 | 113 | 115 | 114 |
| 10 | 18 | 19 | 111 | 110 |

W=>A X=>AB’+A’B =>A⊕ B

K-map for Y K-map for Z

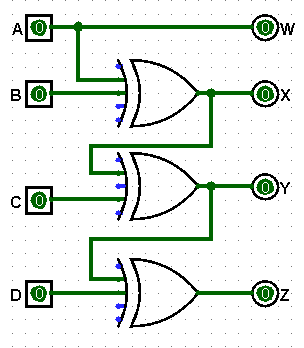
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 13 | 12 |
| 01 | 14 | 15 | 7 | 6 |
| 11 | 12 | 13 | 115 | 114 |
| 10 | 18 | 19 | 11 | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 11 | 3 | 12 |
| 01 | 14 | 5 | 17 | 6 |
| 11 | 12 | 113 | 15 | 114 |
| 10 | 18 | 9 | 111 | 10 |

Y=>A’B’C+A’BC’+ABC+AB’C’ Z=>A⊕ B⊕ C⊕ D

Y=>A’(B’C+BC’)+A(BC+B’C’)

Y=>A⊕ B ⊕ C



GCD to Binary

* 1. BCD to GCD

|  |  |  |  |
| --- | --- | --- | --- |
| **BCD** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **GCD** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |

d(10,11,12,13,14,15) don’t care

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 112 | 113 | 115 | 114 |
| 10 | 18 | 19 | 111 | 110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 14 | 15 | 17 | 16 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 18 | 19 | X11 | X10 |

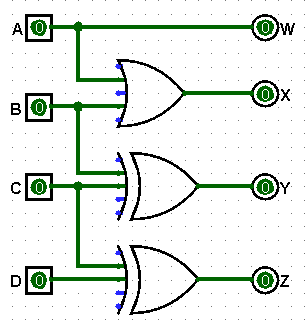
W=>A X=>A+B

K-map for Y K-map for Z

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 13 | 12 |
| 01 | 14 | 15 | 7 | 6 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 8 | 9 | X11 | X10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 11 | 3 | 12 |
| 01 | 4 | 15 | 7 | 16 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 8 | 19 | X11 | X10 |

Y=>B⊕C Z=>C⊕D



BCD to GCD

* 1. GCD to BCD

|  |  |  |  |
| --- | --- | --- | --- |
| **GCD** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **BCD** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 112 | 113 | X15 | X14 |
| 10 | X8 | X9 | X11 | X10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 14 | 15 | 17 | 16 |
| 11 | 12 | 13 | X15 | X14 |
| 10 | X8 | X9 | X11 | X10 |

W=>A X=>A’B

K-map for Y K-map for Z

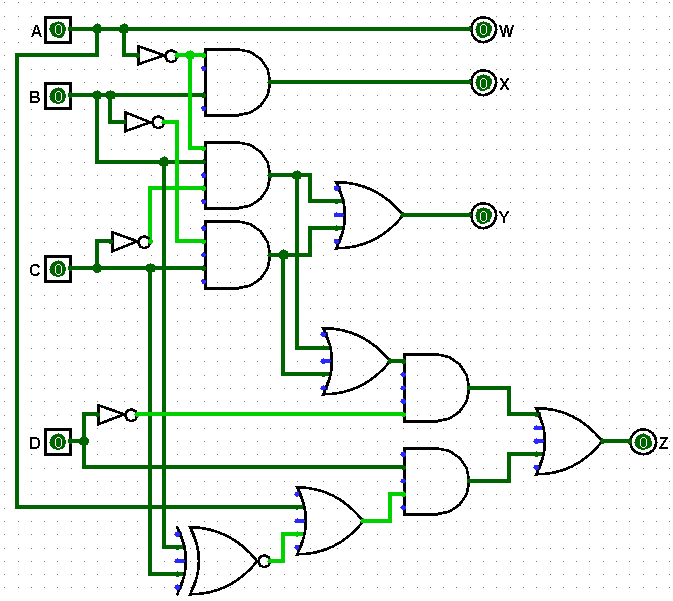
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 13 | 12 |
| 01 | 14 | 15 | 7 | 6 |
| 11 | 12 | 13 | X15 | X14 |
| 10 | X8 | X9 | X11 | X10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 11 | 3 | 12 |
| 01 | 14 | 5 | 17 | 6 |
| 11 | 12 | 113 | X15 | X14 |
| 10 | X8 | X9 | X11 | X10 |

Y=>A’BC’+B’C Z=>AD+BCD+B’C’D+B’CD’+

A’BC’D’

Z=>D(A+B⊙C)+(B’C+A’BC’)D’



GCD to BCD

* 1. BCD to ex-3

|  |  |  |  |
| --- | --- | --- | --- |
| **BCD** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ex-3** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 15 | 17 | 16 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 18 | 19 | X11 | X10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 11 | 13 | 12 |
| 01 | 14 | 5 | 7 | 6 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 8 | 19 | X11 | X10 |

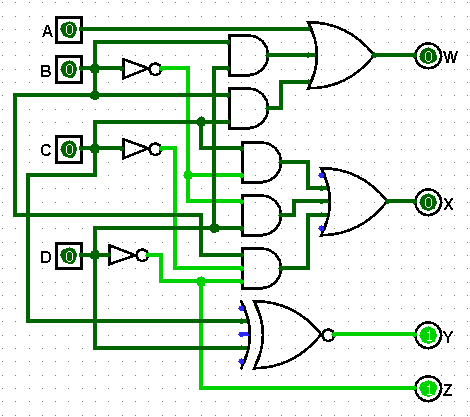
W=>A+BD+BC X=>B’D+B’C+BC’D’

K-map for Y K-map for Z

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 15 | 17 | 16 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 18 | 19 | X11 | X10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | 10 | 1 | 3 | 12 |
| 01 | 14 | 5 | 7 | 16 |
| 11 | X12 | X13 | X15 | X14 |
| 10 | 18 | 9 | X11 | X10 |

Y=>C’D’+CD =>C⊙D Z=>D’



BCD to EX-3

* 1. Ex-3 to BCD

|  |  |  |  |
| --- | --- | --- | --- |
| **Ex-3** | | | |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **BCD** | | | |
| **W** | **X** | **Y** | **Z** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

A

K-map for W K-map for X

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | X0 | X1 | 3 | X2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 112 | X13 | X15 | X14 |
| 10 | 8 | 9 | 111 | 10 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | X0 | X1 | 3 | X12 |
| 01 | 4 | 5 | 17 | 6 |
| 11 | 12 | X13 | X15 | X14 |
| 10 | 18 | 19 | 11 | 110 |

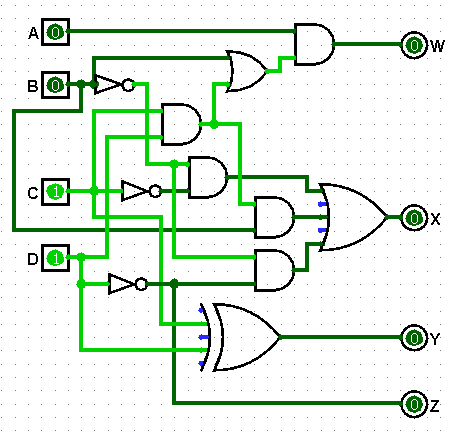
W=>A(B+CD) X=>B’C’+BCD+B’D’

K-map for Y K-map for Z

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | X0 | X1 | 3 | X2 |
| 01 | 4 | 15 | 7 | 16 |
| 11 | 12 | X13 | X15 | X14 |
| 10 | 8 | 19 | 11 | 110 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cd→  ab↓ | 00 | 01 | 11 | 10 |
| 00 | X0 | X1 | 3 | X12 |
| 01 | 14 | 5 | 7 | 16 |
| 11 | 12 | X13 | X15 | X14 |
| 10 | 18 | 9 | 11 | 110 |

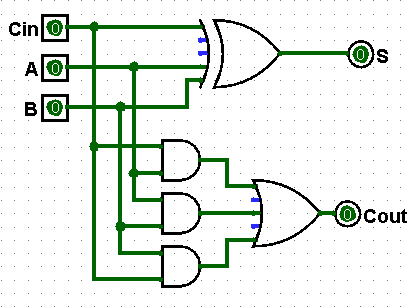
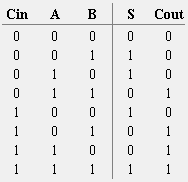
Y=>C⊕D Z=>D’



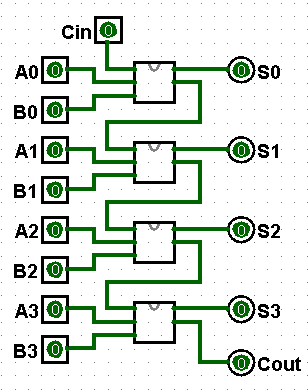
EX-3 to BCD

* 1. To study, design and simulate a **FULL-ADDER Circuit**.

The full adder is used to add three 1-bit binary numbers A, B, and carry C. The full adder has three input states and two output states i.e., sum and carry.

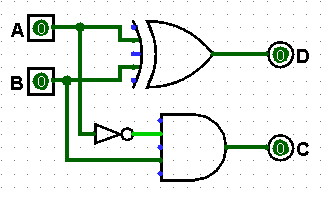
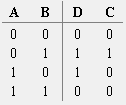
 

* 1. To design **4-bit magnitude parallel adder** using FULL-ADDER circuit.

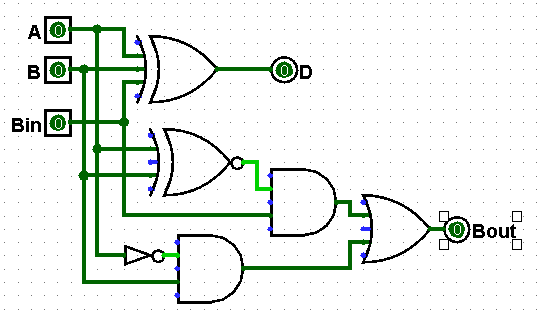
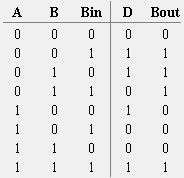


1. To study, design and simulate half subtractor and Full subtractor.

Subtractor circuits take two binary numbers as input and subtract one binary number input from the other binary number input. Similar to adders, it gives out two outputs, difference and borrow.

Half-Subtractor

Full-Subtractor

1. Perform addition of two 8 bit numbers using 8085.

| **Memory Address** | **Mnemonics** | **Comment** |
| --- | --- | --- |
| 2000 | LDA 2050 | A<-[2050] |
| 2003 | MOV H, A | H<-A |
| 2004 | LDA 2051 | A<-[2051] |
| 2007 | ADD H | A<-A+H |
| 2008 | MOV L, A | L←A |
| 2009 | MVI A 00 | A←00 |
| 200B | ADC A | A←A+A+carry |
| 200C | MOV H, A | H←A |
| 200D | SHLD 3050 | H→3051, L→3050 |
| 2010 | HLT |  |

1. **LDA 2050** moves the contents of 2050 memory location to the accumulator.
2. **MOV H, A**copies contents of Accumulator to register H to A
3. **LDA 2051** moves the contents of 2051 memory location to the accumulator.
4. **ADD H**adds contents of A (Accumulator) and H register (F9). The result is stored in A itself. **For all arithmetic instructions A is by default an operand and A stores the result as well**
5. **MOV L, A**copies contents of A (34) to L
6. **MVI A 00**moves immediate data (i.e., 00) to A
7. **ADC A**adds contents of A(00), contents of register specified (i.e A) and carry (1). As ADC is also an arithmetic operation, A is by default an operand and A stores the result as well
8. **MOV H, A**copies contents of A (01) to H
9. **SHLD 3050**moves the contents of L register (34) in 3050 memory location and contents of H register (01) in 3051 memory location
10. **HLT**stops executing the program and halts any further execution
11. To study design and simulate 8X1 Multiplexer.

Multiplexer is a circuit that selects binary information from one of many input lines, and directs the information to a single output line.

The selection of a particular input line is controlled by a set of input variables, called “selection inputs”.

Normally, there are 2n input lines and n selection inputs.

