Department of Computer Science and Engineering
Course: Computer Organization Lab (CSC205)

Location : CSE Lab I

Computer Organization Lab Manual

Experiment 1

AIM:

Realization of basic gates implementation on bread board

Apparatus Required:

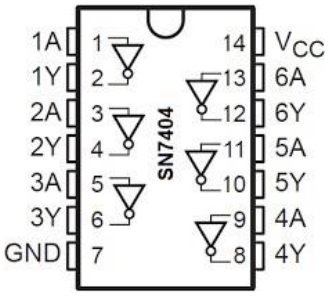
IC 7404, 7408, 7432, LEDs – 4, Breadboard, Connecting wires, and power supply.

Theory:

NOT gate (Truth Table)

i/p	o/p
0	0
0	1
1	0
1	1

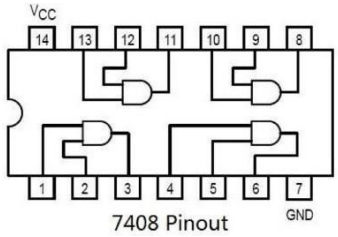
Logic/pin Diagram



AND gate (Truth Table)

A	B	o/p
0	0	0
0	1	0
1	0	0
1	1	1

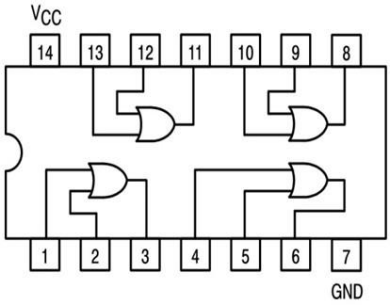
Logic Diagram



AND gate (Truth Table)

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	1

Logic/pin Diagram



Procedure:

For realization

- ICs 7404, 7408,7432 are mounted on breadboard.
- Ground and Vcc pins of each IC is connected accordingly.
- Connections are completed as per the requirements.
- LEDs are connected at each output terminal.
- Inputs as given as per the truth table.

Conclusion:

- Output tested as per truth table through LED.
- Basic gates NOT,AND and OR successfully realized on bread board

Experiment 2

AIM: Design

Half adder and full adder using AND, OR, NOT, XOR gates.
Half subtractor and full subtractor using AND, OR, NOT, XOR gates.
4x1 Multiplexer realization on bread board

Apparatus Required:

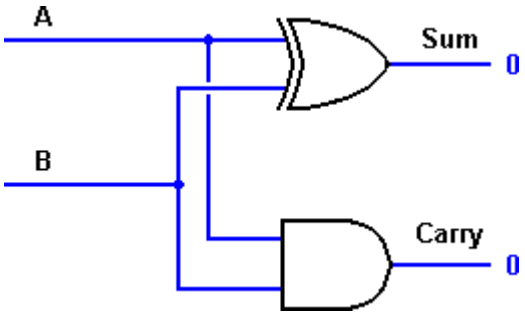
IC 7404, 7408, 7432, 7486, LEDs – 5, Breadboard, Connecting wires, and power supply.

Theory:

Half adder:

It adds two-bit binary digit A & B and yields two outputs Sum (S) and Carry (C).
Here, S=A ⊕B and C=AB

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



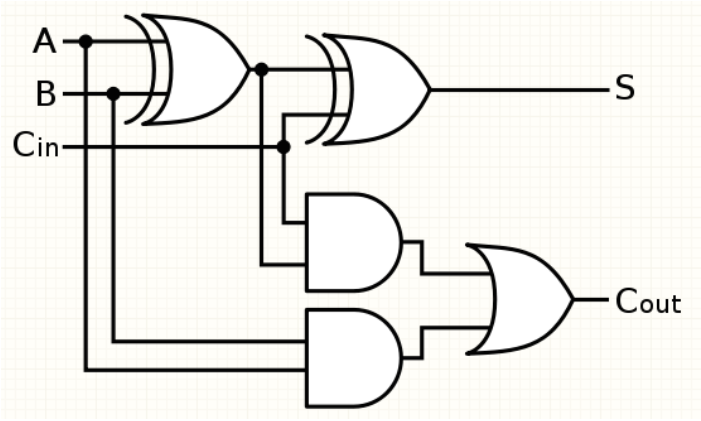
(Half adder)

Full adder:

A full adder adds two-bit numbers A & B, and also adds carry from the previous less significant bit.
Here, S=A ⊕B ⊕Cin and Carry=AB + BC + CA

A	B	Cin	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



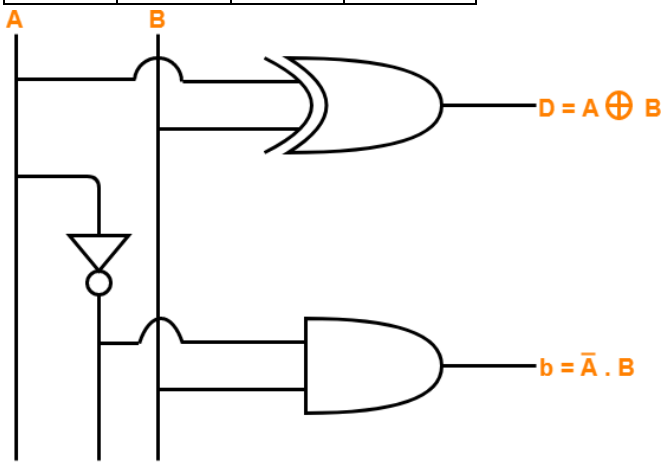
(Full Adder)

Half subtractor:

It subtracts two-bit and produces output as Difference (D) and Borrow (Bo).

D = A ⊕B and Bo = A¯ B

A	B	D	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Half Subtractor Logic Diagram

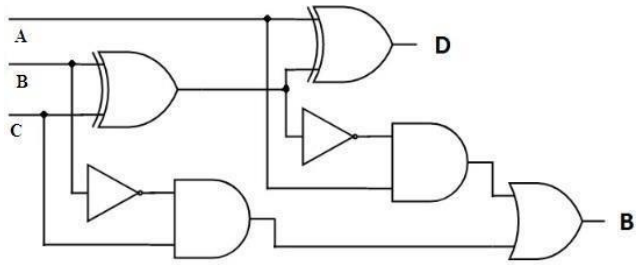
(Half subtractor)

Full subtractor:

A full subtractor subtracts two-bit numbers A &B by consisting the input and borrow (Bo) and generate the output and borrow.

D = A $\oplus$B $\oplus$Bin and Bout = A^- Bin + A^- B + B Bin

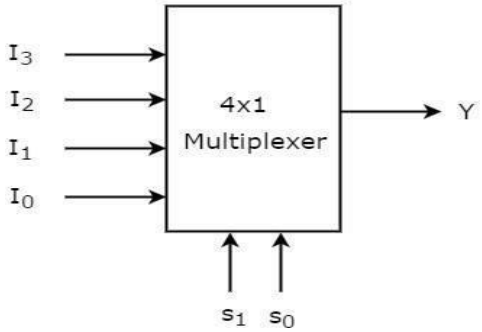
A	B	Bin	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



(Full Subtractor)

4x1 MUX:

It has 4 data inputs (I_3, I_2, I_1 & I_0) and two selection line (S_1 & S_0) that shows which data input has selected.



$Y=S_1^-S_0^- + S_1^-S_0 + S_1S_0^- + S_1S_0$

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Procedure:

For designing half/full adder, subtractor, and MUX.

- ICs 7404,7408, 7486,7432 are mounted on breadboard as per given logic diagram of half/full adder and subtractor
- Ground and Vcc pins of each IC is connected accordingly.
- Connections are completed as per the requirements.
- LEDs are connected at each output terminal.
- Inputs as given as per the truth table.

Conclusion:

- Half adder and full adder are implemented on bread board
- Half and full subtractor are successfully implemented on bread board
- 4x1 Mux are successfully realized on bread board

Experiment 3

AIM:

Realization of basic gates NOT, OR, AND, NOR, XOR, XNOR, Full adder/subtractor using NAND gates

Apparatus Required:

IC 7400, breadboard, LED, Connecting wires and power supply.

Theory:

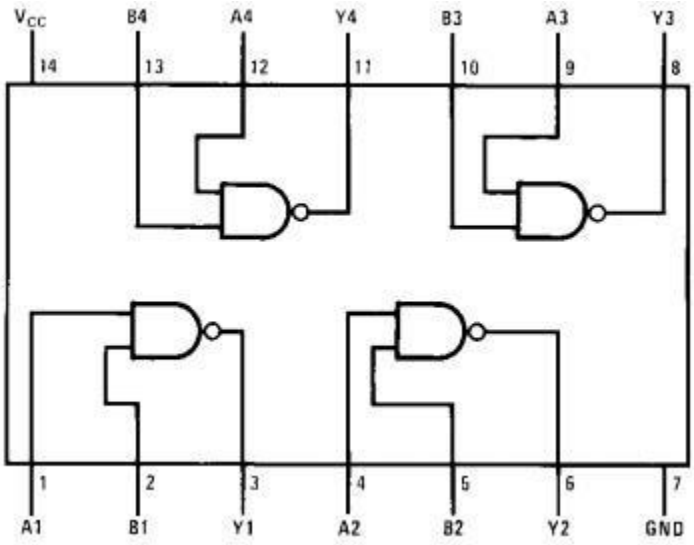
IC 7400: It is a small-scale package that contains dual input NAND gates. Any other logic gates can be made from NAND gate. We can make any logic circuit multiplier of this IC.

NAND gate:

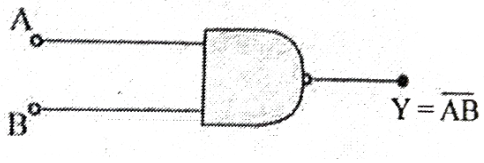
It is made by using transistor.

It is a universal gate.

It is the complement of AND.



(IC 7400 pin diagram)



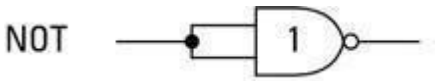
(NAND gate)

A	B	Output
1	1	0

1	0	1
0	1	1
0	0	1

NOT gate:
The NOT gate is made by joining of input of NAND gate together.

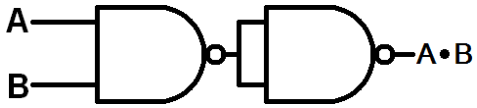
A	Output
0	1
1	0



(NOT gate using NAND gate)

AND gate:
AND gate is realized by following of NAND gate with NOT gate which gives us the complement of NAND.

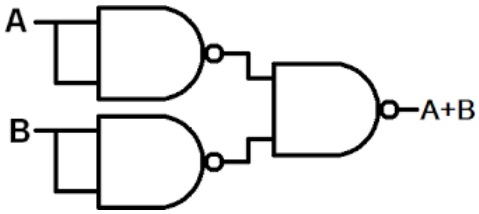
A	B	Output
1	1	1
1	0	0
0	1	0
0	0	0



(AND gate using NAND gate)

OR gate:
An OR gate is realized shorting the inputs and then joining another NAND gate.

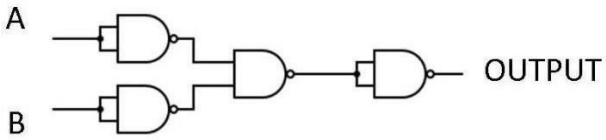
A	B	Output
1	1	1
1	0	1
0	1	1
0	0	0



(OR gate using NAND gate)

NOR gate:
It is another universal gate and realize any logical expression. It is the complement of OR gate.

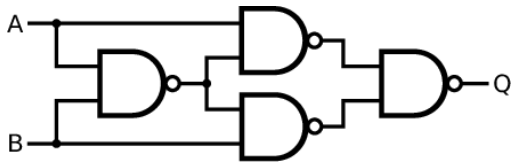
A	B	Output
1	1	0
1	0	0
0	1	0
0	0	1



(NOR using NAND gate)

XOR gate:
It is realized using 4 NAND gates connected is the way as follows.

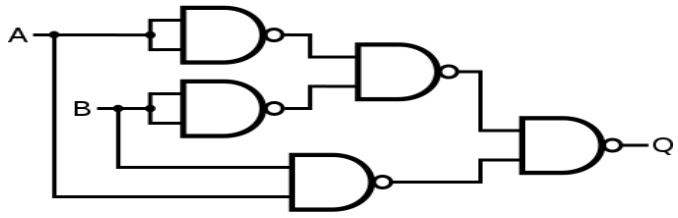
A	B	Output
1	1	0
1	0	1
0	1	1
0	0	0



(XOR using NAND gate)

XNOR gate:
It is negation of XOR gate.

A	B	Output
1	1	1
1	0	0
0	1	0
0	0	1

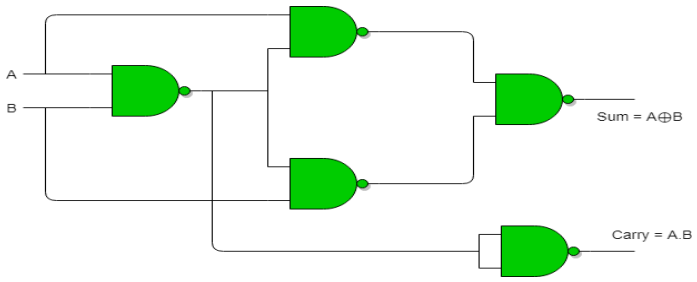


(XNOR using NAND gate)

ADDER:
Adders are digital circuit that perform addition operation on numbers.

Half Adder:
It adds two single binary digits A and B. It adds two outputs Sum (S) and Carry (C).
Here, S=A $\oplus$B and C=AB.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

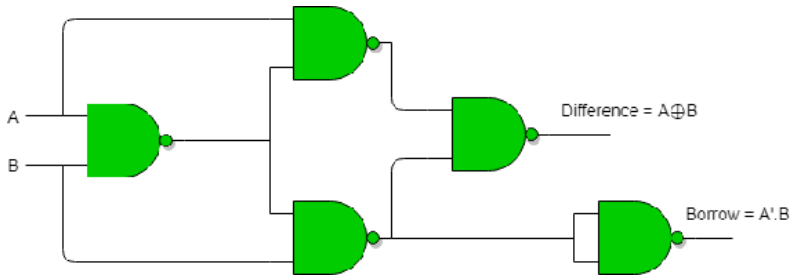


(Half adder using NAND gate)

Subtractor:
It is a digital circuit that perform subtraction.

Half subtractor:
It subtracts two-bit binary digits A & B and produce output difference and borrow.

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



(Half subtractor using NAND gate)

Procedure:

As per logic diagram of selected gate for realization perform the following.

- IC 7400 is mounted on breadboard.
- Pin 7 and 14 and connected to ground and Vcc respectively.
- Connections are made as per circuit/logic diagram of gate/adder/subtractor.
- Inputs are given according to the truth table of circuit.
- LED is connected at output.
- Logic is 1 for ON state of LED and logic 0 for OFF state of LED.

Conclusion:

- All basic gates are successfully realized using universal NAND gate.
- Half adder, full adder, half subtractor and full subtractor are successfully implemented using NAND gate.

Experiment 4

AIM:

Design of 4 bit Incrementor using multiplexers/NAND gates

Apparatus Required:

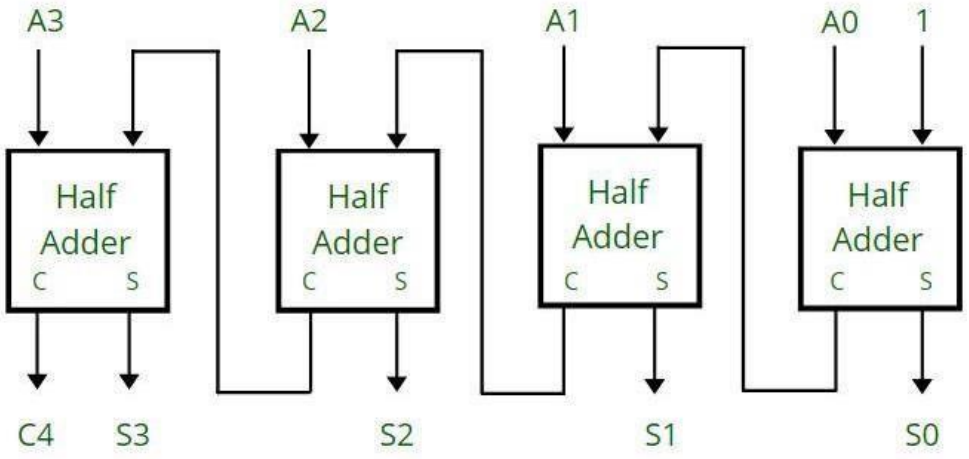
IC 7400, IC 7404, IC 7408, IC 7432, breadboard, LED, Connecting wires and power supply.

Theory:

4-bit incrementor:

It adds the 1 to the current value and stored into a register. A 4-bit binary Incrementor required 4 cascaded half adder circuit.

A_3	A_2	A_1	A_0	S_3	S_2	S_1	S_0	C
0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	1	0
0	0	1	1	0	1	0	0	0
0	1	0	0	0	1	0	1	0
0	1	0	1	0	1	1	0	0
0	1	1	0	0	1	1	1	0
0	1	1	1	1	0	0	0	0
1	0	0	0	1	0	0	1	0
1	0	0	1	1	0	1	0	0
1	0	1	0	1	0	1	1	0
1	0	1	1	1	1	0	0	0
1	1	0	0	1	1	0	1	0
1	1	0	1	1	1	1	0	0
1	1	1	0	1	1	1	1	0
1	1	1	1	0	0	0	0	1



(4-bit Binary incrementor)

Procedure:

For designing incrementor using NAND gates/ Multiplexer following steps are used.

- ICs 7400/ IC 7404, IC 7408, IC 7432are mounted on breadboard as per logic diagram of Half adder given in experiment 2 and 3.
- Ground and Vcc pins of each ICs arr connected accordingly.
- Connections are completed as per the requirements.
- LEDs are connected at each output terminal.
- Inputs as given as per the truth table.

Conclusion:

- Incrementor circuit successfully implemented using Multiplexer designed using basic gates on bread board.
- Incrementor circuit successfully implemented using NAND gates on bread board.

Experiment 5

AIM: To realize 4 bit logical and arithmetic circuits separately and then integrating them into a single arithmetic and logical unit (ALU).

Apparatus Required: Breadboard, connecting wires, light emitting diode, battery eliminator and integrated circuits: IC7400 (NAND), IC7486(XOR), IC7408(AND), IC7432(OR), IC7404 (NOT), IC7483 (full address), IC74153 (4×1 MUX).

Theory:

1) Logical circuits: It takes n-bits of two numbers and performs bitwise logical AND, OR, NOT, XOR operations and produces the output. The output from various logic gates is fed to 4×1 multiplexer and the required output is chosen via select lines.

Truth table: (Logical circuit)

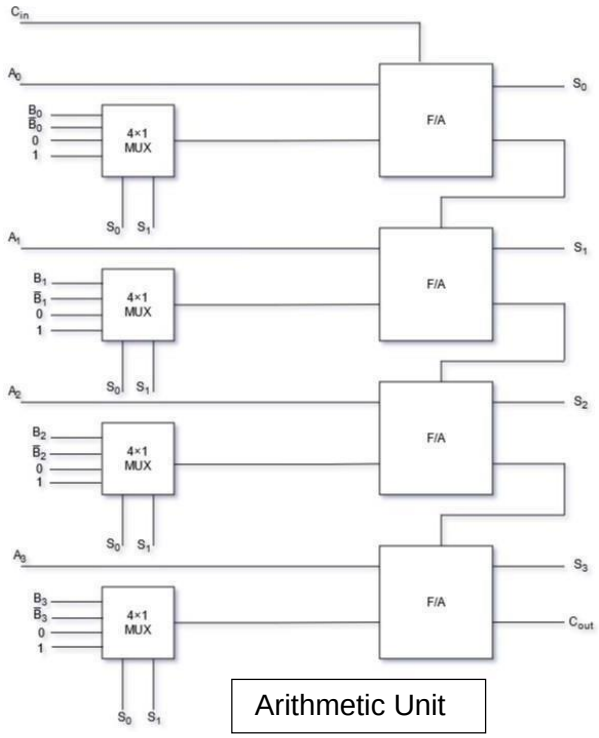
A- 1111, B- 1010, taken during the selection.

S_1	S_0	Operation	Output
0	0	AND	1010
0	1	OR	1111
1	0	XOR	0101
1	1	NOT	0000

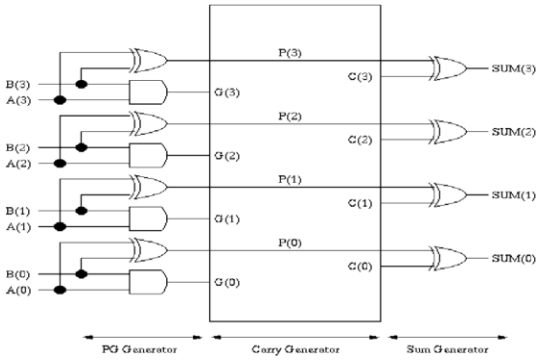
2) Arithmetic circuit: It is a circuit that takes two n-bits words and performs arithmetic operations on it.

Data input	Input/output Central system			Data input	Output
	S_1	S_0	C_{in}		
A	0	0	0	B	A+B
A	0	0	1	B	A+B+1
A	0	1	0	B	A+$\bar{B}$
A	0	1	1	B	A+$\bar{B}$ +1
A	1	0	0	B	A
A	1	0	1	B	A+1
A	1	1	0	B	A-1
A	1	1	1	B	A

3) Arithmetic and logical unit: It is a combinational circuit that can perform various arithmetic and logical operation over two n-bit words as input. The operation to be performed is selected through the select lines.



Arithmetic Unit



Logic Unit

Note: Integration of Arithmetic and Logic unit can be implemented by either of the following

(i)Using multiplexer (i.e. output will be selected based on arithmetic/logic control variable

(ii)Modification of Boolean expressions of full adder inputs. (Refer class notes)

Procedure:

1) IC's for respective logical and arithmetic circuits were mounted on different set of breadboards.
2) The circuit (ALU) was verified for different combinations of select lines and data inputs.

Precautions:

1) All the connections were right and tight.
2) IC's were biased properly.
3) All the apparatus were handled with care.
4) All LED's were tested beforehand.

Conclusion:

The arithmetic and logical unit was released and cross verified from different combinations of data inputs and select lines.

Applications:
1) The ALU is a fundamental building block of the CPU of a modern computer.
2) ALU is responsible for computer mathematical calculations such as floating point math.
3) With the help of an efficient and versatile software, the ALU can also be used to perform advanced scientific computing.

Experiment 6

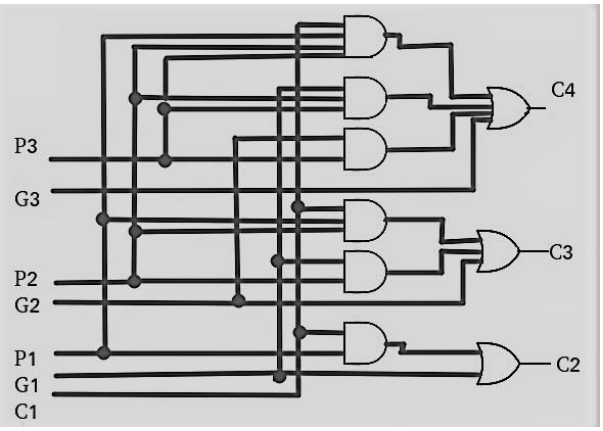
AIM: To realize Carry Look Ahead adder using AND/OR/XOR gates.

Apparatus Required: Breadboard, connecting wires, light emitting diode, integrated circuit-IC7486 (XOR), IC7408(AND), IC7432(XOR).

Theory: A carry look ahead (CLA) or fast adder is a type of adder used in digital logic. A carry look ahead adder improves speed by reducing amount of time required to determine carry bits. Two variations are defined on the basis of truth table carry generate and carry propagate.

$P_i = A_i \oplus B_i$
$G_i = A_iB_i$

The sum output and carry output can be expressed in terms of G_i and P_i as
$C_{i+1} = G_i + P_iC_i$
$C_2 = G_i + P_1C_1 = G_1 + P_1G_0 + P_1P_0G_{in}$
$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0G_{in}$
$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{in}$



Procedure & precautions:

1) ICs are mounted on breadboard as per logic diagram drawn
2) Setup of power supply and ground connection on bread board.
3) IC's were biased properly.
4) All LED's were tested beforehand.
5) All the apparatus were handled with ease.

6) All IC's were mounted on the breadboard and biased properly.
7) LED's were mounted at the end of required output pins.
8) All IC's were mounted on the breadboard and biased properly.
9) Connecting wires were used to make the connections between the pins as per the circuit in theory.

Conclusion:

The carry look ahead adder was realized by using 2 input AND/OR/XOR gates and cross verified for different combinations of input.

Applications: The circuit is used in place of ripple carry adder for faster computations examples include ALU, PC, and timers.

Experiment 7

AIM: To realize

a) 4-bit carry save adder using full Adder/ANO/XOR/OR

b) 4-bit shift register using J-K Flip Flop

Apparatus Required:

Breadboard connecting wires light emitting diodes(LED),battery eliminator, IC 7486(XOR), IC 7408(ANO), IC 7432(OR), IC 7483 14 bit full adder, IC 7476

Theory:

1) Carry Save Adder- A **carry-save adder** is a type of digital adder used in computer micro architecture to efficiently compute the sum of three or more n bit binary numbers. It differs from other digital adders in that it outputs two (or more) numbers of the same dimension as the inputs, one which is a sequence of partial sum bit and the another which is a sequence of carry bit. These two sets of bits are added via normal full adder circuit.
2) Shift Register- A **shift register** is a type of digital circuit using a cascade of flip flops where the output of one flip-flop is connected to the input of the next in the chain resulting in a circuit that shift by one position the bit array stored in it "shifting in" the data present at its input and "shifting out" the least bit in the array, at each transition of the clock input.

Procedure:

1) All IC's were mounted on the breadboard & biased properly.
2) LED's were mounted at the end of the required output pins.
3) Finally the circuit was tested by feeding it with different combinations of output.

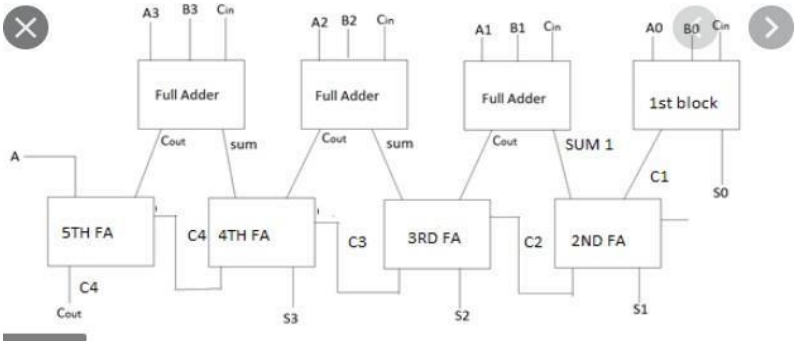


Fig: 4-bit carry save adder

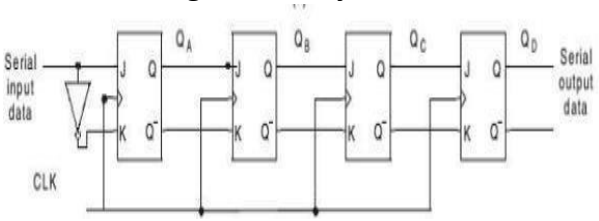


Fig: 4 bit shift register

Precautions:

1) All connections were tight.
2) All the apparatus were handled with care.
3) IC's were properly biased.
4) All the LED's were tested beforehand.

Conclusion:

1) 4-bit carry save address using full adder/ AND/ XOR/OR gates was realized.
2) 4 bit shift operator using JK flip flop was realized.

Application:

1) Carry save adder: It is the most efficient choice to perform fast multiplication as multi-operand operations are required too after.
2) Shift Register: It is used as parallel to serial convertors and vice versa, it, along the combinational logic generates sequence hence, used as sequence generator. They are also used in counter circuit.

Experiment 8

AIM:

To implement the Binary Code Decimal (BCD) adder.

Apparatus Required:

2 Half Adder, IC 74136, OR gate IC, AND gate IC

Theory:

Binary Coded Decimal is a binary encoding of decimal numbers where each decimal digit is represented by a fixed number of 4 bits.

Binary Sum

k	Z8	Z4	Z2	Z1
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1

BCD Sum

c	s8	s4	s2	s1
0	0	0	0	1
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0

0	1	0	0	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1

K=0

Z8 Z4 \ Z2Z1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

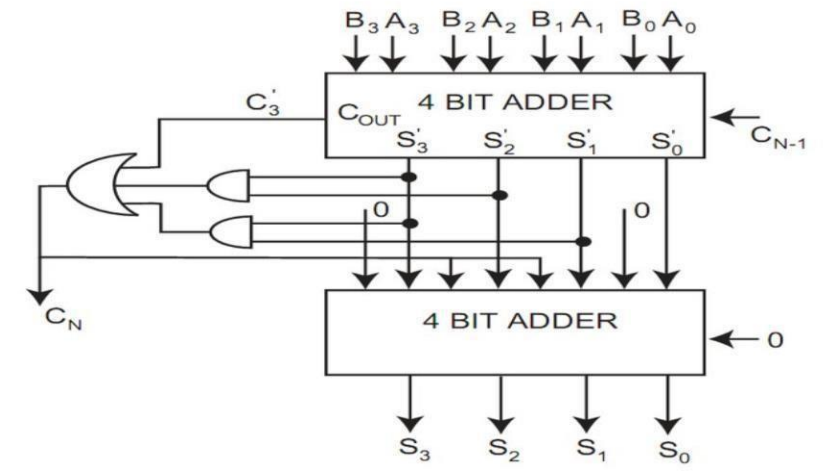
K=1

Z8 Z4 \ Z2Z1	00	01	11	10
00	1	1	1	1
01	x	x	x	x
11	x	x	x	x
10	x	x	x	x

C= K+Z8Z4+Z8Z2

Procedure:

Set up the circuit as shown in the figure. The inputs are given to BCD adder and the output BCD is obtained as shown in the figure.



4 bit BCD adder

Conclusion:

We have successfully verified the 4 bit BCD adder.

Sources of error:

1) Connections may be loose
2) IC's may be faulty.
3) LED's may be fused

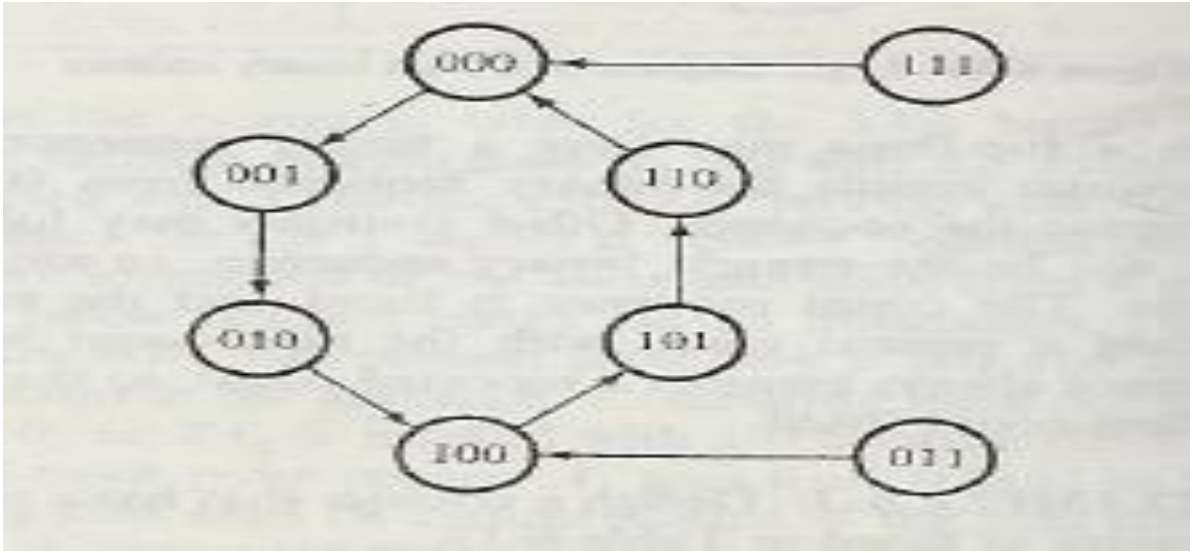
Experiment 9

AIM:
To implement a counter that has a repeated sequence of given six states using J-K flip flop

Apparatus Required:
IC 7476 , Breadboard, connecting wires, clock generator, power supply, LED bulbs.

Theory:

Given six repeated count sequences are 0,1,2,4,5,6. The state diagram is shown below.



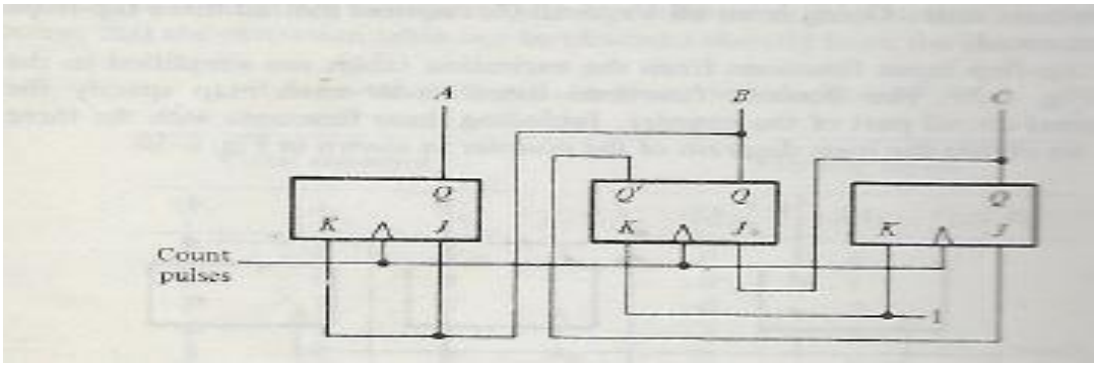
For above diagram, the state/excitation table can be derived as follows.

TABLE 6-13 Excitation table for Example 6-3

Count sequence			Flip-flop inputs					
A	*B*	*C*	*JA*	*KA*	*JB*	*KB*	*JC*	*KC*
0	0	0	0	*X*	0	*X*	1	*X*
0	0	1	0	*X*	1	*X*	*X*	1
0	1	0	1	*X*	*X*	1	0	*X*
1	0	0	*X*	0	0	*X*	1	*X*
1	0	1	*X*	0	1	*X*	*X*	1
1	1	0	*X*	1	*X*	1	0	*X*

Procedure

Set up the circuit as shown below on bread board The inputs are given to flip flops. The clock pulse generated from signal generator will be used to observe the working of counter.



Conclusion: Given counter implemented using J-K flip flop successfully.

Experiment 10

AIM:

To design a combinational circuit that squares, a 3 bit binary number using ROM.

Apparatus Required:

IC 7476, IC 74138(Decoder), IC 7432(OR gates), IC 7408(AND), IC 7407(NOT), Breadboard, Connecting wires, clock generator, power supply, LED bulbs.

Theory:

ROM (Read only Memory): It refers to computer memory chips containing permanent and semi-permanent data. Unlike RAM, ROM is non –volatile even after you turn off your computer, the content of ROM will return.

The truth table to find squares of members:

INPUT OUTPUT

A2	A1	A0	B5	B4	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

Procedure:

- All the IC's are mounted on breadboard according to need.
- Connections are made as per circuit diagram.
- Verify the O/P for each input combination.
- Give Inputs according to the truth table.
- Check for any loose connections.
- Switch off power supply after finishing.

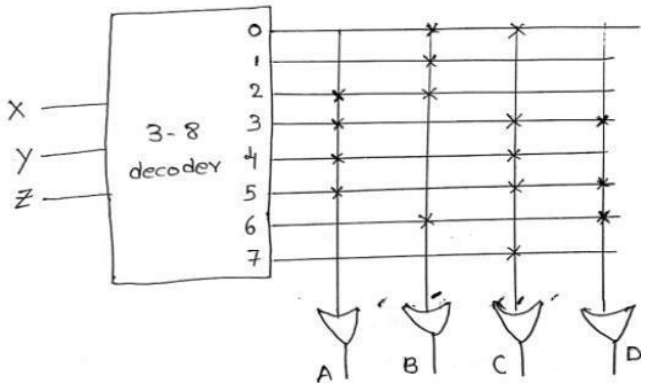


Fig: Block Diagram ROM

Note: Decoder can be designed using basic gates (refer class notes)

Conclusion:

We have successfully implemented combinational circuit using ROM and IC's.

SOURCES OF ERROR

- Connections may be loose
- IC's may be faulty
- LED's may be faulty

Experiment 11

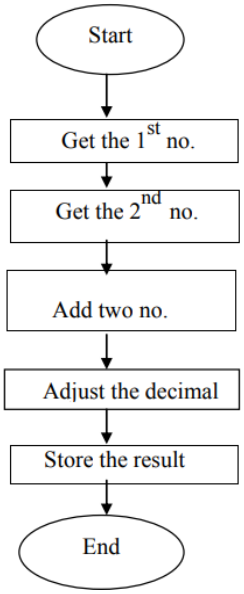
AIM:
Write a program to add two decimal numbers.

Apparatus Required:
8085 Microprocessor programming kit, instruction coding sheet.
Power supply

Theory:

Steps:

1. Initialize HL Reg. pair with address where the first number is lying.
2. Store the number in accumulator.
3. Get the second number.
4. Add the two numbers and store the result in 200B.
5. Go back to Monitor



Procedure:

```
LXI H, 2009 ; Point 1st no.
MOV A, M ; Load the acc.
INX H ; Adv Pointer
ADD M ; ADD 2nd NO.
DAA ; Adjust the decimal
INX H ; Adv Pointer
MOV M, A ; Store Result
RST 5
```

Conclusion:
Thus the numbers at 2009H and at memory are added.

Experiment 11

AIM:
Write a program to add two hexadecimal & decimal numbers.

Apparatus Required:
8085 Microprocessor programming kit, instruction coding sheet.
Power supply

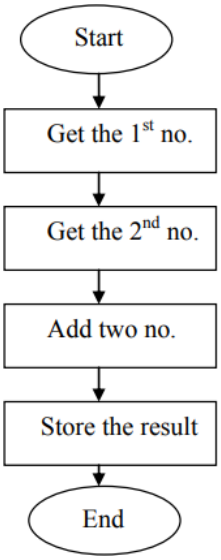
Theory:

Hexadecimal Addition: The program takes the content of 2009, adds it to 200B & stores the result
back at 200C.

Steps:

1. Initialize HL Reg. pair with address where the first number is lying.
2. Store the number in accumulator.
3. Get the second number.
4. Add the two numbers and store the result in 200B.
5. Go back to Monitor

Let:(2009 H) = 80 H
(200B H) = 15 H
Result = 80 H + 15 H = 95 H
(2009 H) → A
A → B
(200B H) → A
A + B → A
A → (200C H)



Procedure:

LXI H, 2009 ; Point 1st no.
MOV A, M ; Load the acc.
INX H ; Adv Pointer
ADD M ; ADD 2nd NO.
INX H ; Adv Pointer
MOV M, A ; Store Result
RST 5

Conclusion:
Thus the numbers at 2009H and at memory are added.

Experiment 11

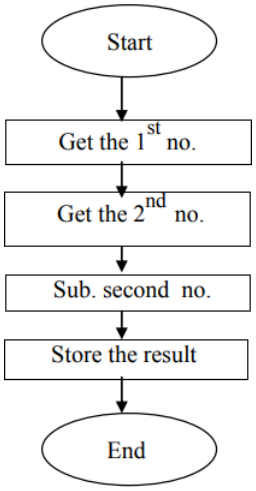
AIM:
Write a program to subtract two hexadecimal.

Apparatus Required:
8085 Microprocessor programming kit, instruction coding sheet.
Power supply

Theory:
Steps:- 1. Initialize HL Reg. pair with address where the first number is lying. 2. Store the number in accumulator. 3. Get the second number. 4. Subtract second no from ACC and store the result in 200B. 5. Go back to Monitor

Procedure:

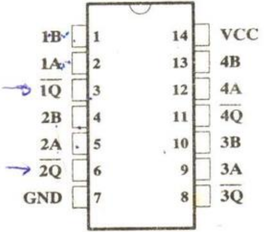
```
LXI H, 2009 ; Point 1st no.
MOV A, M ; Load the acc.
INX H ; Adv Pointer
SUB M ; Subtract IIND NO.
INX H ; Adv Pointer
MOV M, A ; Store Result
RST 5
```

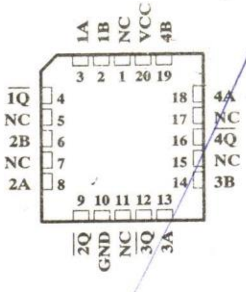


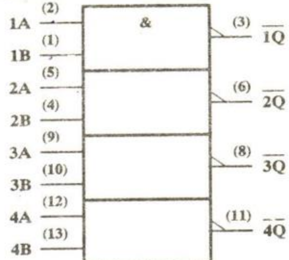
Conclusion:
Thus the subtraction operation is taken out using assembly language.

INPUTS		OUTP.
A	B	$\overline{Q}$
H	H	L
X	L	H
L	X	H

$Q = \overline{A\,B}$







7400 Typ Type	Hersteller Production Fabricants Produttori	Gehäuse Case Boîtier Carcassa	T_U°C	I_S &I_R	Pin	t_{PD} E → Q ns_{typ}	t_{PD} E → Q ns_{max}	Bemerk. Note f_T f_Z &f_E

7400 Typ Type	Hersteller Production Fabricants Produttori	Gehäuse Case Boîtier Carcassa	T_U°C	I_S &I_R	Pin	t_{PD} E → Q ns_{typ}	t_{PD} E → Q ns_{max}	Bemerk. Note f_T f_Z &f_E

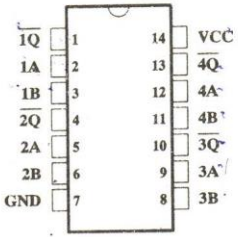
7402

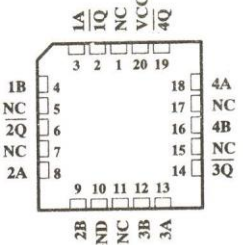
Output: TP

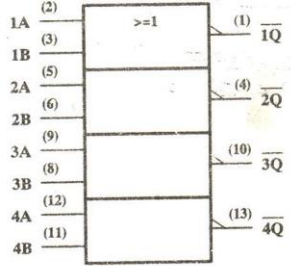
4 NOR-Gatter mit je 2 Eingängen
Quad 2-Input NOR Gate
4 porte NOR à 2 entrées
3 porte NOR con rispettivamente 2 ingressi
Puerta cuádruple NOR cada una con 2 entradas

INPUTS		OUTP.
A	B	$\overline{Q}$
H	X	L
X	H	L
L	L	H

$Q = \overline{A + B}$







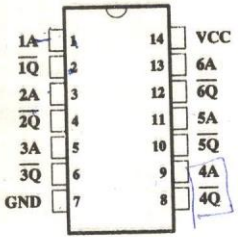
7404

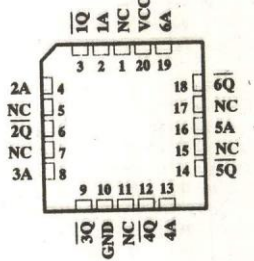
Output: TP

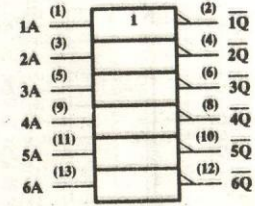
6 Inverter
Hex Inverter
6 Inverseur
Invertitore esstuplo
Inversor séxtuple

INPUT	OUTP.
A	$\overline{Q}$
H	L
L	H

$Q = \overline{A}$







Hersteller	Gehäuse	t_{PD}	t_{PD}	Bemerk.

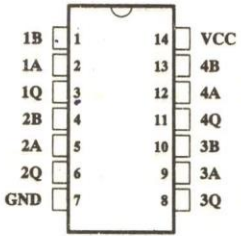
7408

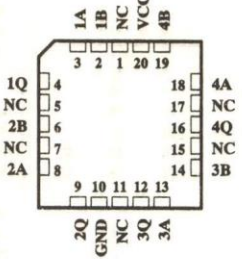
Output: TP

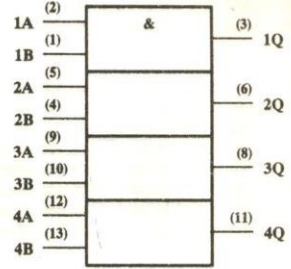
4 AND-Gatter mit je 2 Eingängen
Quad 2-Input AND Gate
4 porte AND à 2 entrées
4 porte AND con rispettivamente 2 ingressi
Puerta cuádruple AND cada una con 2 entradas

INPUTS		OUTP.
A	B	Q
H	H	H
X	L	L
L	X	L

Q = A B







Hersteller	Gehäuse		Anz.	Anz.	Remark

Hersteller	Gehäuse			Remark

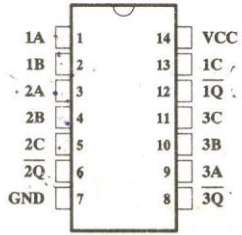
7410

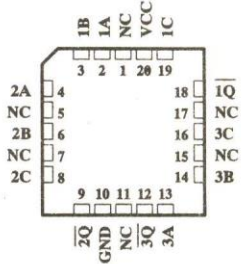
Output: TP

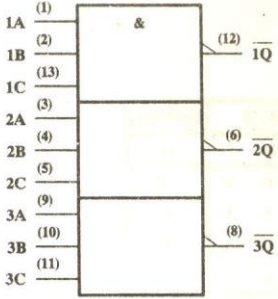
3 NAND-Gatter mit je 3 Eingängen
Triple 3-input NAND Gate
3 porte NAND à 3 entrées
Porta triplo NAND con rispettivamente 3 ingressi
Puerta triple NAND cada una con 3 entradas

INPUTS			OUTPUT
A	B	C	$\overline{Q}$
H	H	H	L
L	X	X	H
X	L	X	H
X	X	L	H

$Q = \overline{A\,B\,C}$







Hersteller	Gehäuse					Inp.	Inp.	Bemerk.

Hersteller	Gehäuse					Inp.	Inp.	Bemerk.

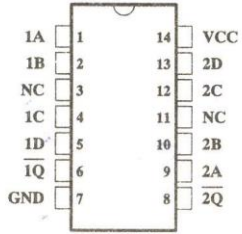
7420

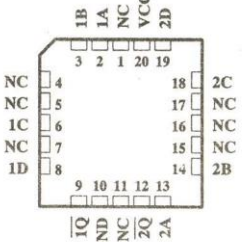
Output: TP

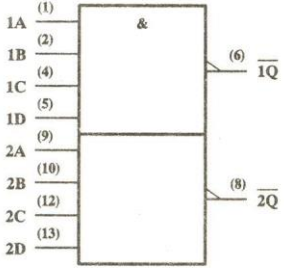
2 NAND-Gatter mit je 4 Eingängen
Dual 4-Input NAND Gate
2 porte NAND à 4 entrées
2 porte NAND con rispettivamente 4 ingressi
Puerta doble NAND, cada una con 4 entradas

INPUTS				OUTPUT
A	B	C	D	$\overline{Q}$
H	H	H	H	L
L	X	X	X	H
X	L	X	X	H
X	X	L	X	H
X	X	X	L	H

$Q = \overline{A\,B\,C\,D}$







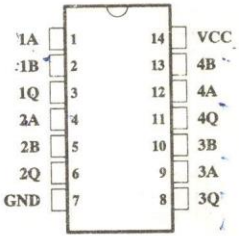
7432

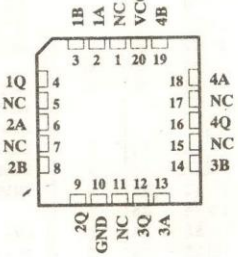
Output: TP

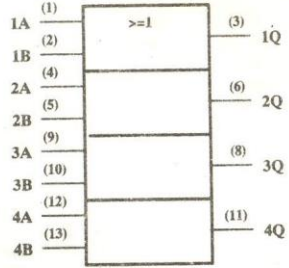
4 OR-Gatter mit je 2 Eingängen
Quad 2-Input OR Gate
4 porte OR à 2 entrées
4 porte OR con rispettivamente 2 ingressi
Puerta cuádruple OR cada una con 2 entradas

INPUTS		OUTPUT
A	B	Q
H	X	H
X	H	H
L	L	L

Q = A + B







Hersteller	Gehäuse	t_{PD}	I_{PD}	Bemerk.

Hersteller	Gehäuse	t_{PD}	t_{PD}	Bemerk.

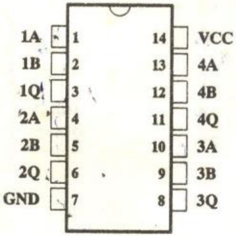
7486

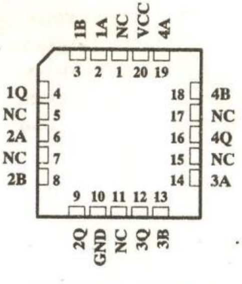
Output: TP

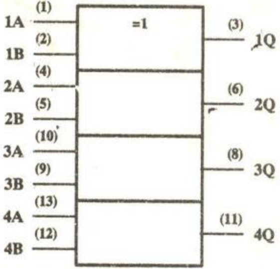
4 EX-OR-Gatter mit je 2 Eingängen
Quad 2-Input EX-OR Gate
4 étages porte EX-OR à 2 entrées
4 porte EX-OR con rispettivamente 2 ingressi
4 Puertas EX-OR, cada una con 2 entradas

INPUTS		OUTPUT
A	B	Q
L	L	L
L	H	H
H	L	H
H	H	L

$Q = A \oplus B$







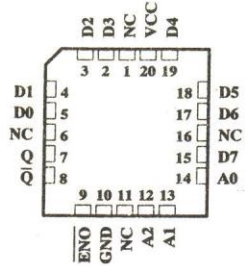
Hersteller Gehäuse tpd tpd Bemerk

Hersteller Gehäuse

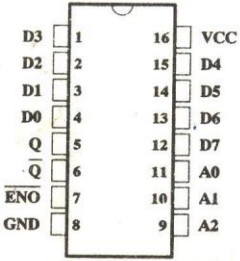
74151

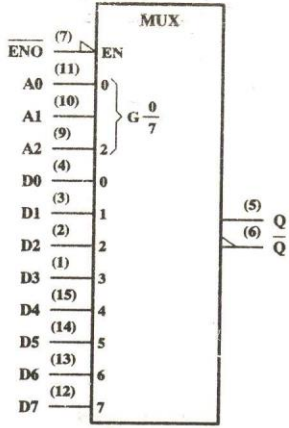
Output: TP

8-zu-1-Datenselektor/Multiplexer mit Strobe
8-to-1 Data Selector/Multiplexer with Strobe
Sélecteur de données/multiplexeur 8/1 avec Strobe 8/1
Selettore dati/multiplexer 8 a 1 con Strobe
Selector de datos/multiplexor de 8 a 1 con activación (Strobe)



INPUTS				OUTP.	
A2	A1	A0	ENO	Q	Q̄
X	X	X	H	L	H
L	L	L	L	D0	D0̄
L	L	H	L	D1	D1̄
.	.	.	.	.	.
H	H	L	L	D6	D6̄
H	H	H	L	D7	D7̄





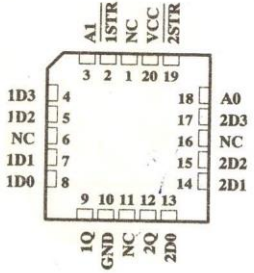
Hersteller	Gehäuse	typ. t_{PLH}	typ. t_{PHL}	Bemerk.

Hersteller	Gehäuse	typ. t_{PLH}	typ. t_{PHL}	Bemerk.

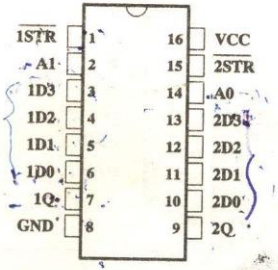
74153

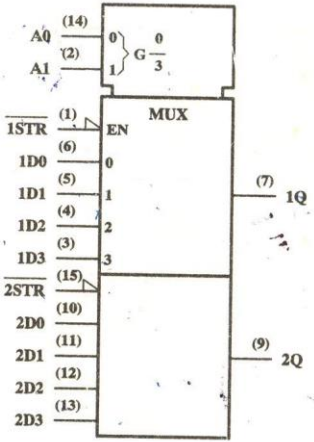
Output: TP

2 4-zu-1-Datenselektoren/Multiplexer
Dual 4-to-1 Data Selector/Multiplexer
2 sélecteurs de données/multiplexeurs 4/1
2 selettori dati/multiplexer 4 a 1
2 Selectores de datos/multiplexores de 4 a 1



INPUTS			OUTP.
$\overline{STR}$	A1	A0	Q
H	X	X	L
L	L	L	D0
L	L	H	D1
L	H	L	D3
L	H	H	D4



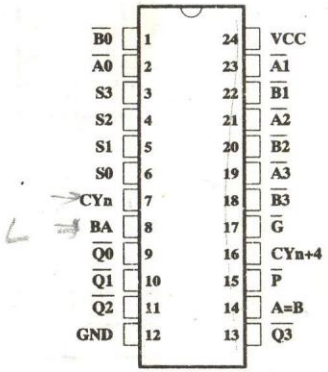


Hersteller	Gehäuse	I_{DD}	t_{pd}	Remark

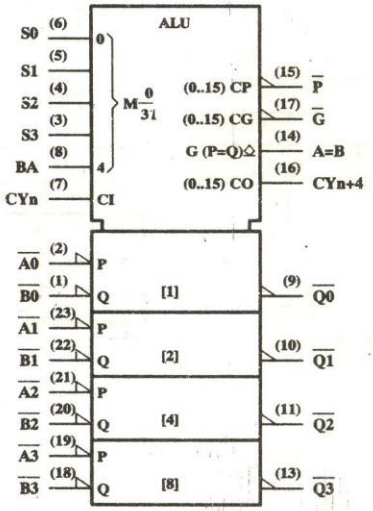
74181

Output: TP

4-Bit arithmetische/logische Einheit, Funktionsgenerator
4-Bit Arithmetic/Logic Unit, Function Generator
Unité arithmétique/logique 4 bits, générateur de fonctions
Unità aritmetica/logica 4 bit, generatore di funzioni
Unidad aritmético/lógica de 4 bits, generador de funciones



MODE				DATA OUTPUTS		
				BA=H: LOG. FUNC.	BA=L: ARITHMETIC FUNCTION	
S3	S2	S1	S0	CYn = X	CYn = H	CYn = L
L	L	L	L	$\overline{A}$	A	A plus 1
L	L	L	H	$\overline{A+B}$	A + B	(A + B) plus 1
L	L	H	L	$\overline{A}B$	$A+\overline{B}$	$(A+\overline{B})$ plus 1
L	L	H	H	L	minus 1	zero
L	H	L	L	$\overline{A\overline{B}}$	A plus $(A\overline{B})$	A plus $(A\overline{B})$ plus 1
L	H	L	H	$\overline{B}$	$(A+B)$ plus $(A\overline{B})$	$(A+B)$ plus $(A\overline{B})$ plus 1
L	H	H	L	$A \oplus B$	A minus B minus 1	A minus B
L	H	H	H	$A\overline{B}$	$(A\overline{B})$ minus 1	$A\overline{B}$
H	L	L	L	$\overline{A}+B$	A plus $(A\overline{B})$	A plus (A B) plus 1
H	L	L	H	$\overline{A \oplus B}$	A plus B	A plus B plus 1
H	L	H	L	B	$(A+\overline{B})$ plus (A B)	$(A+\overline{B})$ plus (A B) plus 1
H	L	H	H	AB	(A B) minus 1	AB
H	H	L	L	H	A plus A	A plus A plus 1
H	H	L	H	$A+\overline{B}$	(A + B) plus A	(A + B) plus A plus 1
H	H	H	L	A + B	$(A+\overline{B})$ plus A	$(A+\overline{B})$ plus A plus 1
H	H	H	H	A	A minus 1	A



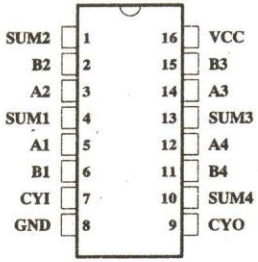
Hersteller	Gehäuse	tpn	tpn	Bemerk.

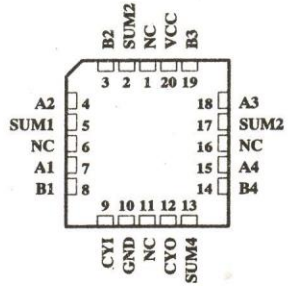
Hersteller	Gehäuse	tpn	tpn	Bemerk.

74283

Output: TP

4-Bit Volladdierer
4-Bit Full Adder
Additionneur complet 4 bits
Addizionatore totale di 4 bit
Sumador completo de 4 bits





INPUTS				OUTPUTS					
				CYI = L			CYI = H		
				CYIN = L			CYIN = H		
A1 / A3	B1 / B3	A2 / A4	B2 / B4	SUM1 / SUM3	SUM2 / SUM4	CYIN / CYO	SUM1 / SUM3	SUM2 / SUM4	CYIN / CYO
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

Note: input conditions at A1, A2, B1, B2 and CYI are used to determine outputs SUM1 and SUM2 and the value of the internal carry CYIN. The values at CYIN, A3, A4, B3 and B4 are then used to determine outputs SUM3, SUM4 and CYO.