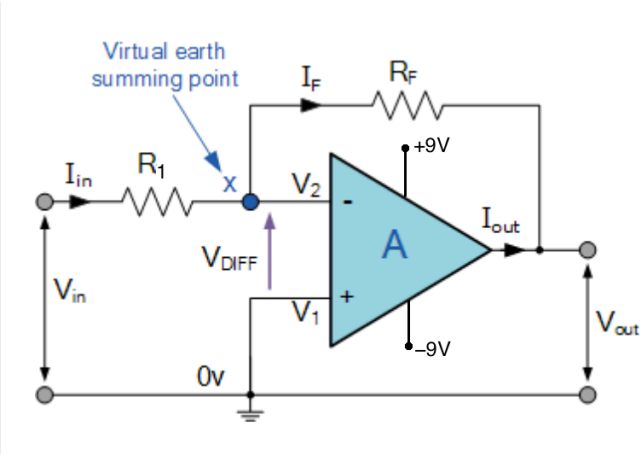
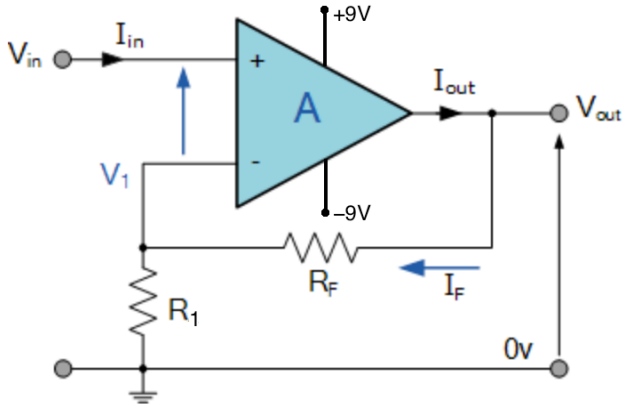
PHY 307: Electronics II
Experiments

Experiment 1: IC 741: Inverting Amplifier, Non-Inverting Amplifier

Inverting Amplifier:



Non-inverting Amplifier:



Try R_F = 10kΩ, R_1 = 4.7kΩ.
Voltage gain for ideal scenario:

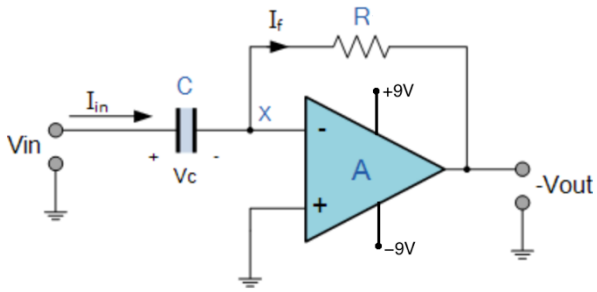
Inverting amplifier: $A_v = V_{out}/V_{in} = -R_F/R_{in}$
Non-inverting amplifier: $A_v = V_{out}/V_{in} = 1+R_F/R_{in}$

Input various signals (sinusoidal, triangular etc.) using the function generator with Vpp=250mV, 500mV, and see both the input and output signals using DSO. Find the voltage gain and compare with the theoretical value.
See what happens to the output signal if you input signal has Vpp $\gtrsim$ 2.5V (Don't make it too large though.).

Sources:
https://www.electronics-tutorials.ws/opamp/opamp_1.html
https://www.electronics-tutorials.ws/opamp/opamp_2.html
https://www.electronics-tutorials.ws/opamp/opamp_3.html

Experiment 2: IC 741: Differentiator, Integrator

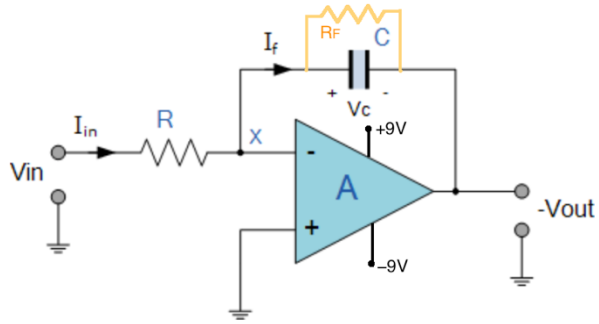
Differentiator:



$$V_{out}(t) = -RC\frac{dV_{in}(t)}{dt}$$

Try R=10kΩ, C=10nF.
Try different waveforms as input (with Vpp ~ 2V and frequency ~ 1kHz) and compare with the output waveforms. See, if you get the expected result, as given by the above formula.

Integrator:



$$V_{out}(t) = -\frac{1}{RC}\int V_{in}(t)dt$$

Try R=10kΩ, C=10nF.
In case of too much noise, add R_F~100kΩ in parallel to C.
Try different waveforms as input (with Vpp ~ 2V and frequency ~ 5kHz) and compare with the output waveforms. See, if you get the expected result, as given by the above formula.

Sources:

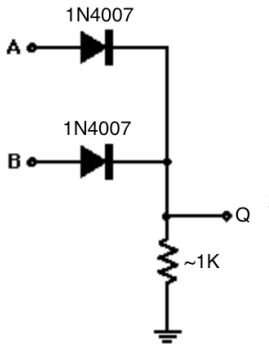
https://www.electronics-tutorials.ws/opamp/opamp_6.html
https://www.electronics-tutorials.ws/opamp/opamp_7.html

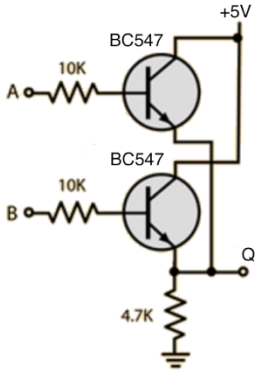
Experiment 3: Realizing Logic gates using diodes and transistors

In all the circuits below, consider +5V as the level 1, and 0V (circuit-ground) as level 0. Verify the truth table for each. You may choose to observe the output using a multimeter (used as voltmeter) or a LED with about 470Ω resistor in series. What are the differences between the diode based circuit and transistor based circuit in terms of working?

(i) OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

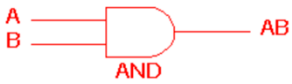




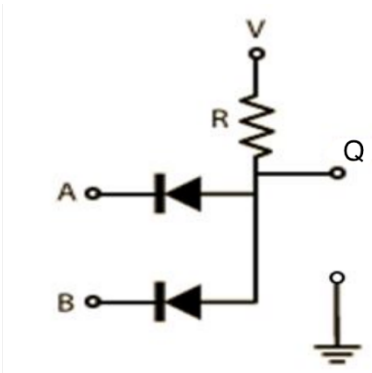
Diode based

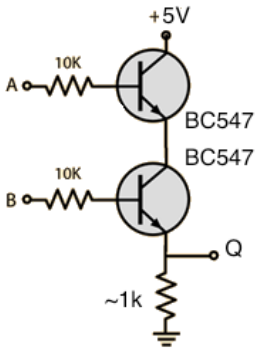
Transistor based

(ii) AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

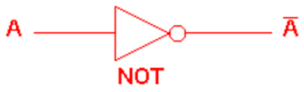




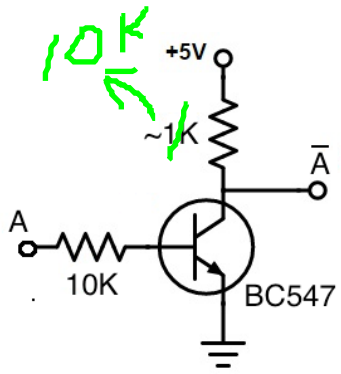
Diode based

Transistor based

(iii) NOT gate



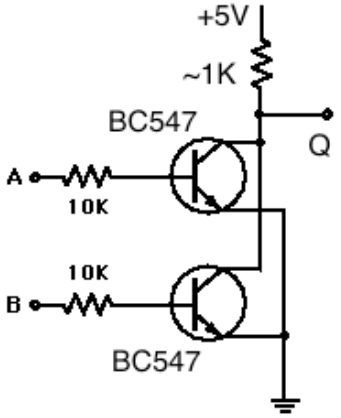
NOT gate	
A	$\overline{A}$
0	1
1	0



Question: Can NOT gate be realized only using diode(s)?

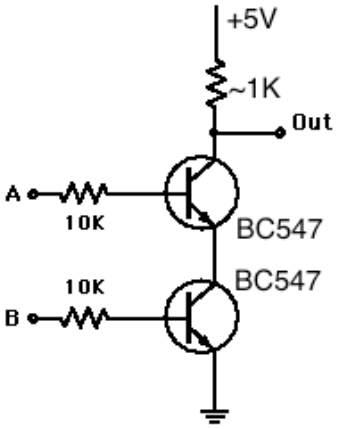
(iv) NOR gate

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



(v) NAND gate

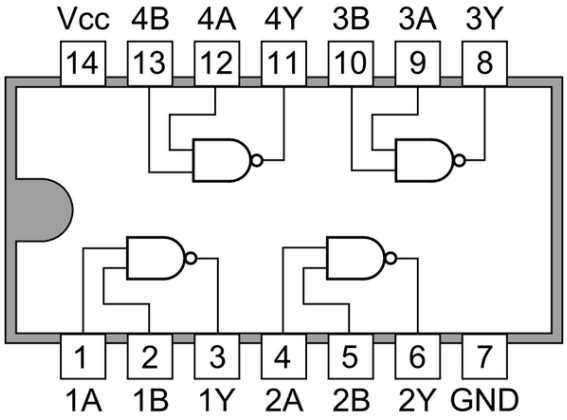
2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



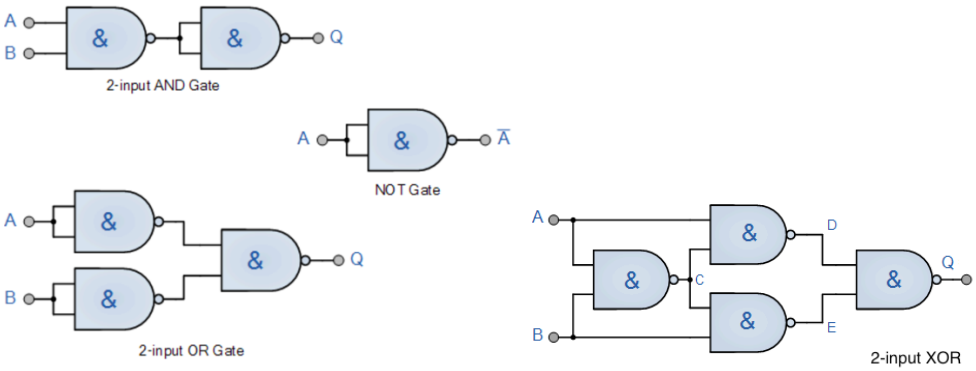
Experiment 4: Use of universal gates (NAND, NOR) for constructing other gates

Use IC 7400 for NAND gates. Give power supply to the IC using pin 14 (+5V) and pin 7 (0V). Use the four in-built NAND gates, as shown below, for your experiment. Verify the truth table in each case.

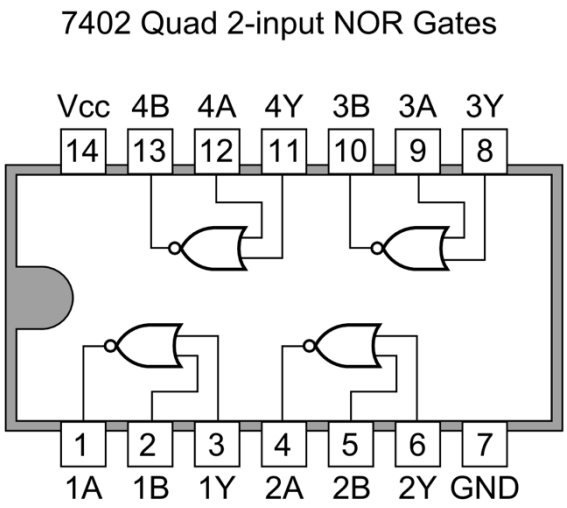
7400 Quad 2-input NAND Gates

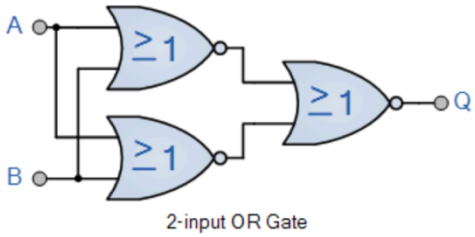


Various Logic Gates using only NAND Gates

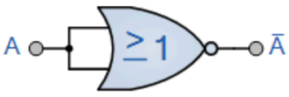


Use IC 7402 for NOR gates. Give power supply to the IC using pin 14 (+5V) and pin 7 (0V). Use the four in-built NAND gates, as shown below, for your experiment. Verify the truth table in each case.

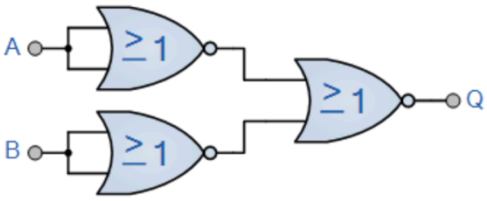




2-input OR Gate



NOT Gate



2-input AND Gate

Tasks: Try to construct XOR and XNOR gates using NOR gates.

Sources:

https://www.electronics-tutorials.ws/logic/logic_5.html
https://www.electronics-tutorials.ws/logic/logic_6.html

Experiment 5: Verification of Boolean algebra using logic gates

Verify the following Boolean equalities by implementing the LHS and RHS using logic gates, i.e., construct the circuits realizing the two sides, and show that their outputs are identical for given set of inputs.

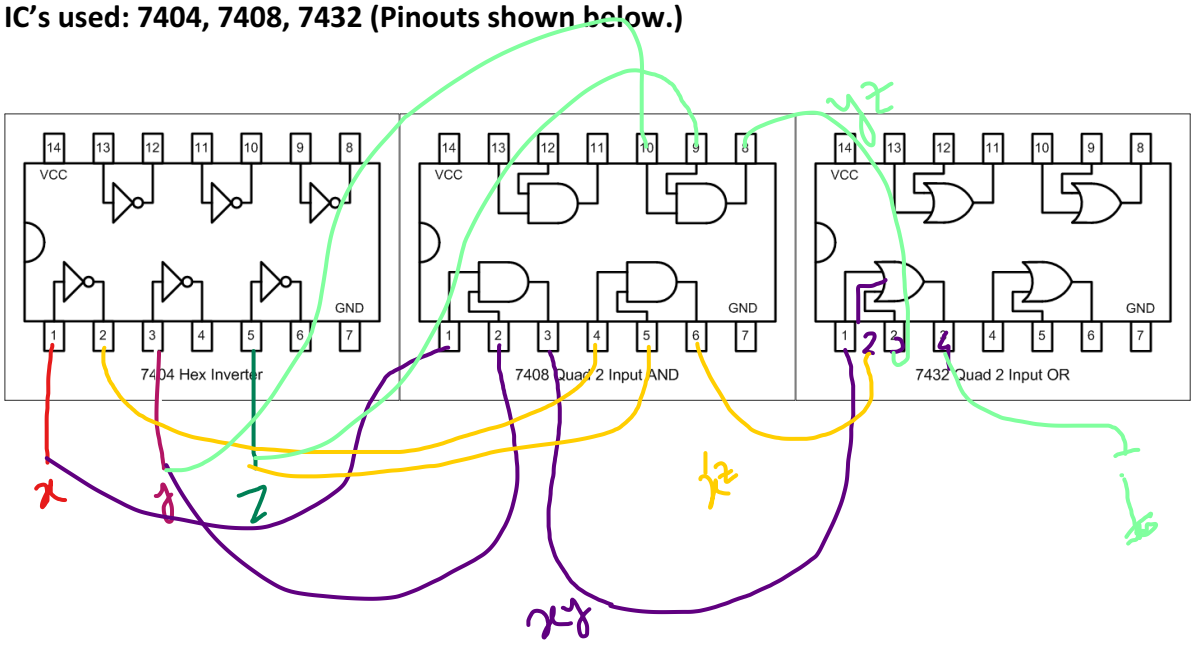
(i) $\mathbf{A}' \cdot \mathbf{B} + \mathbf{A} \cdot \boldsymbol{B}' = (\mathbf{A} \cdot \boldsymbol{B})' \cdot (\mathbf{A} + \mathbf{B})$

Gates needed: 2 NOT, 2 AND, 1 OR

(ii) $\mathbf{A} \cdot \mathbf{B} + (\mathbf{A} + \mathbf{B}) \cdot (\mathbf{B} + \mathbf{C}) = \mathbf{A} \cdot \mathbf{C} + \mathbf{B}$

Gates needed: 3 OR, 2 AND

IC's used: 7404, 7408, 7432 (Pinouts shown below.)



Experiment 6: Verification of SOP, POS, and Karnaugh-simplified Boolean expressions using logic gates

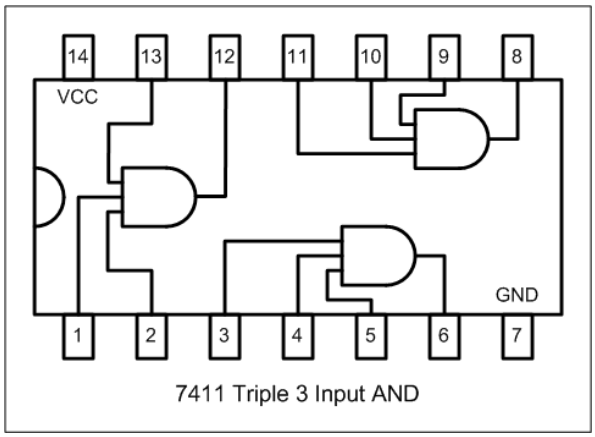
Verify by constructing circuits using logic gates that each of the following expressions are equivalent:

SOP: $\mathbf{A' \cdot B \cdot C + A' \cdot B \cdot C + A \cdot B' \cdot C + A \cdot B \cdot C}$ $xy + x'z + yz$

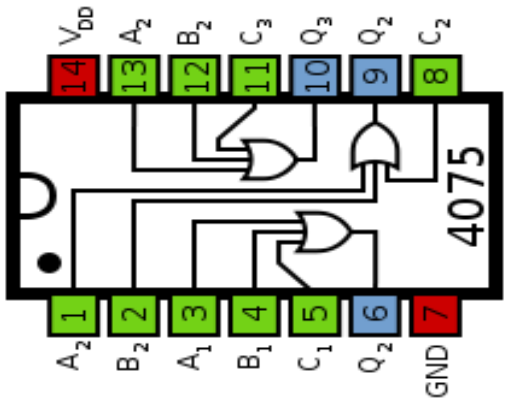
POS: $(\boldsymbol{A + B + C}) \cdot (\boldsymbol{A + B + C'}) \cdot (\boldsymbol{A' + B + C}) \cdot (\boldsymbol{A' + B' + C})$

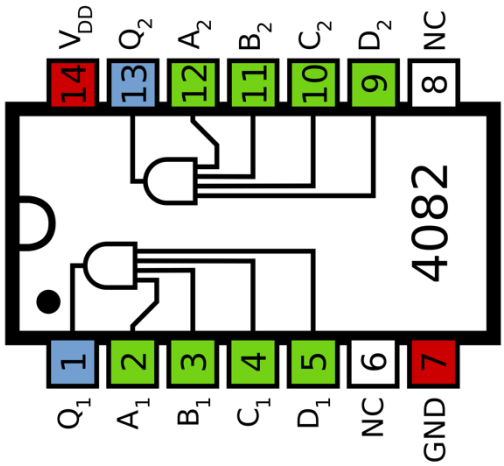
Karnaugh-simplified: $\mathbf{A' \cdot B + A \cdot C}$

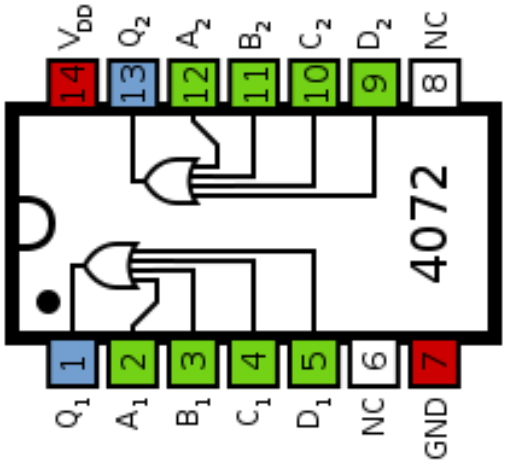
Gates needed: 3-input AND, 3-input OR, 4-input AND, 4-input OR, NOT, 2-input AND, 2-input OR



7411 Triple 3 Input AND







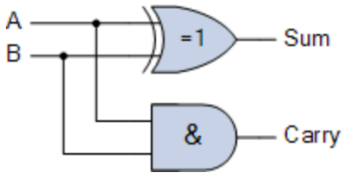
(For other ICs refer to the last experiment.)

Experiment 7: Verification of 2/3 bit binary addition/subtraction: Half adder, Full adder using logic gates

In each case implement the circuits using ICs and verify the truth table.

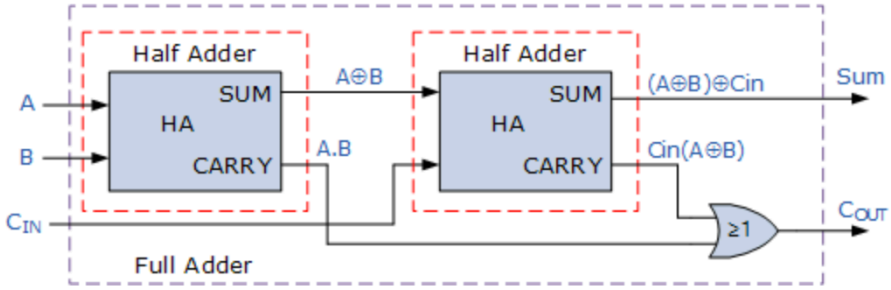
Half adder (Adds two 1-bit binary numbers):

Half Adder Truth Table with Carry-Out

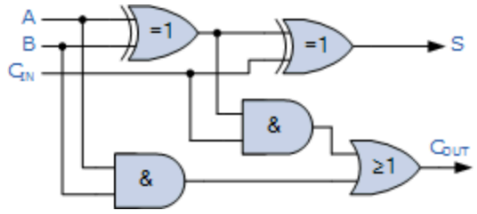
Symbol	B	A	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1

Full adder (Adds three 1-bit binary numbers, typically one of the bits is the carry from previous stage):

Full Adder Logic Diagram

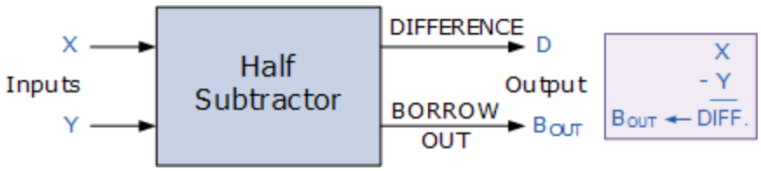


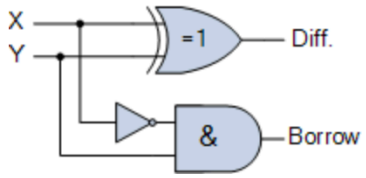
Full Adder Truth Table with Carry

Symbol	Truth Table				
	C-in	B	A	Sum	C-out
	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

Half-subtractor:

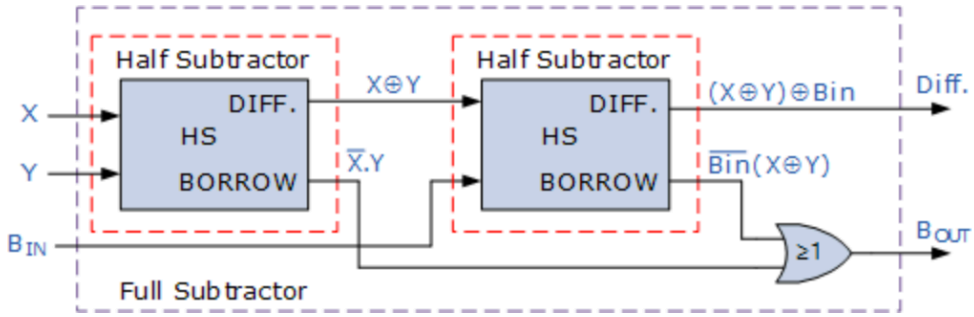
Half Subtractor with Borrow-out



Symbol	Truth Table			
	Y	X	DIFFERENCE	BORROW
	0	0	0	0
	0	1	1	0
	1	0	1	1
	1	1	0	0

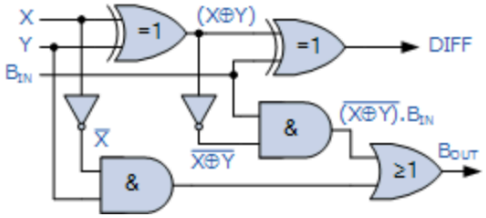
Full Subtractor:

Full Subtractor Logic Diagram



Full Subtractor Truth Table

Symbol	Truth Table				
	B-in	Y	X	Diff.	B-out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	1
	0	1	1	0	0
	1	0	0	1	1
	1	0	1	0	0
	1	1	0	0	1
	1	1	1	1	1



Sources:

https://www.electronics-tutorials.ws/combination/comb_7.html
https://www.electronics-tutorials.ws/combination/binary-subtractor.html

Experiment 8

Aim: To exercise BCD decoding to 7-segment display and realize circuits for displaying any three digits from 0-9.

Apparatus required: IC's (7447, 7432, 4075 and 7404), FND 7-segment panel, resistors of values 220Ω or higher, connecting wires, breadboard, power supply,

Theory:
The basic idea of this exercise is to take binary information to logical display. A particular binary number of 4-bit BCD systems can be used to energize a LED circuit arranged in systematic manner to show decimal equivalent. For this purpose, a set of 7 LED's arranged in a sequence known as 7-segment display, Fig.1 is used. This display unit is provided dc electric power through various Boolean circuits so that a group of LED segments glow to indicate desired decimal number. For example, if a 0000 is input fed to the circuit, six of available 7 LED segments on the seven-segment display should glow except the LED segment available at the center (horizontal). These segments are popularly assigned alphabets like a, b, c, d, e, f and g (Fig.1).

Fig.1 7-segment display

Various examples of different glowing LED segments indicating different decimal equivalents are shown below, Fig.2. Considering 4-bit BCD input flow to drive 7-segment a truth table is used for designing distinct boolean circuit for glowing each segment. For example- segment ***a*** should glow for BCD inputs corresponding to decimal numbers 0, 2, 3, 5, 6, 7, 8 and 9 (considering that BCD input may be used to indicate alphanumeric information where display conflict between numeral '***6***' and alphabet '***b***' must be planned to avoid. Therefore the circuit designed for '***a***' must produce logic ***1*** for above numbers and logic '***0***' for numbers 1 and 4. It is known that BCD numbers use only ten of the sixteen possible combinations of four bits (0-9). Therefore, using *don't care states* (10-15) is an advantage to reduce circuits for LED segments as minimum as possible.

Truth table for designing circuits for powering each LED segment:

	BCD input				7-LED segments						
Decimal	**b3**	**b2**	**b1**	**b0**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

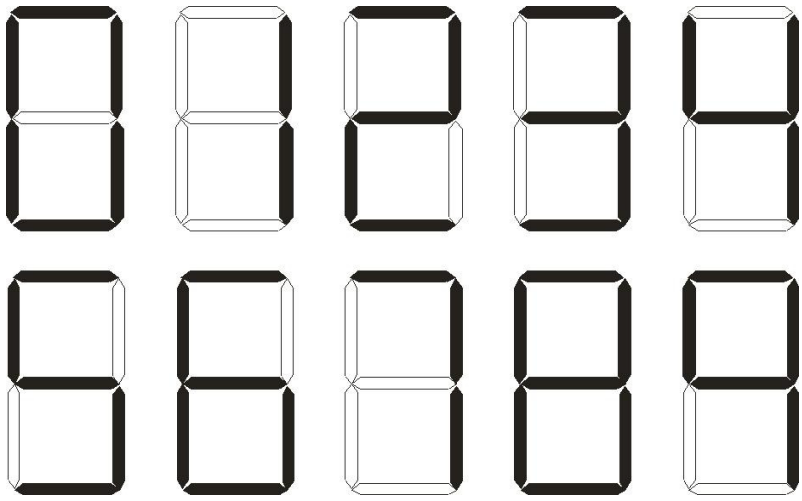


Fig.2 Seven-segment display decimal indication

From the truth table given on previous page, Karnaugh map for designing Boolean circuit for powering segment '***a***' can be drawn like,

b1b0 / b3b2	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	d	d	d	d
10	1	1	d	d

Simplifying above K-map, following Boolean function is obtained:

$$\boldsymbol{a = b3 + b1 + b2b0 + \overline{b2}\,\overline{b0}}$$

Similarly simplifying K-map for other segments ***b-g*** following Boolean function is obtained as:

$$\boldsymbol{b = \overline{b2} + \overline{b1}\,\overline{b0} + b1b0}$$

$$\boldsymbol{c = b0 + \overline{b1} + b2}$$

$$\boldsymbol{d = \overline{b2}\,\overline{b0} + \overline{b2}b1 + b1\overline{b0} + b0\overline{b1}b2}$$

$$\boldsymbol{e = b1\overline{b0} + \overline{b2}\,\overline{b0}}$$

$$\boldsymbol{f = b3 + \overline{b1}\,\overline{b0} + b2\overline{b1} + b2\overline{b0}}$$

$$\boldsymbol{g = b3 + b2\overline{b1} + b1\overline{b0} + \overline{b2}b1}$$

Similarly Boolean functions for other segments can be written. In real practice, building all distinct Boolean circuits is not necessary as BCD-7 segment decoder IC 7447 has all these individual circuits summarized on one platform. However, in order to use this IC platform to energize seven-segment display pin numbers must be well recognized and labeled.

7-segment panel are constructed usually in two types: a) common anode and b) common cathode. Common anode is the type in which anodes of each LED are connected together and provided +5V dc power. Common cathode is the type in which all cathodes are connected together and extended to ground. These two types are indicated as shown below to indicate '**0**':

Common Cathode

a	b	c	d	e	f	g
1	1	1	1	1	1	0

Common Anode

a	b	c	d	e	f	g
0	0	0	0	0	0	1

Procedure:

1) Arrange circuit for energizing a particular segment on breadboard.
2) Every single LED segment must get power through a resistor to limit the current. *If you connect the LED between +5V and ground without the resistor, the LED will momentarily glow bright and then* **never** *glow again*. For this display use, a 220Ω resistor is popularly used with conventional 7-segment panel.
3) Improper connections to the seven-segment display can destroy it. Double check connections **before** applying power.
4) Inside the seven-segment display, one end of each LED is connected to a common point. This common point is tied either to ground or to the positive supply, depending on the specific device. If your seven segment display is designed to have the common connection tied to the positive supply, +5V, it is called a *common anode* configuration as shown below, Fig.3. To light these LED segments, the inputs must be a logic low.

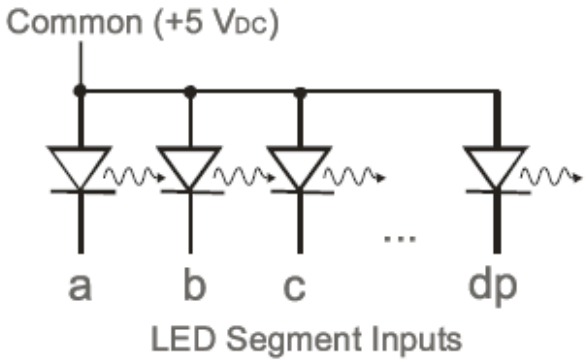


Fig.3 Common anode 7-segment display circuit

5) If Fa, Fb, etc... are +5 V, Fig.4, there is no voltage drop across the LED and resister resulting in no current flow through them, (and the LED remains dark.) If the inputs are 0 Volts, a current is produced and the diode glows.

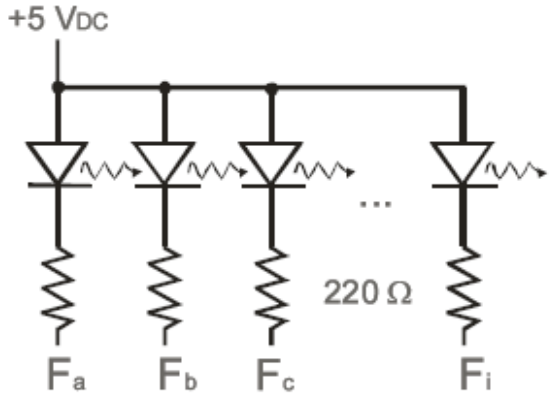


Fig.4 Resistors used with a common anode 7-segment display

6) If your lab kits contain a *common cathode* display, Fig.5, the common point is *ground* instead of +5 V as shown below. To light these segments, logic '**1**' must be supplied.

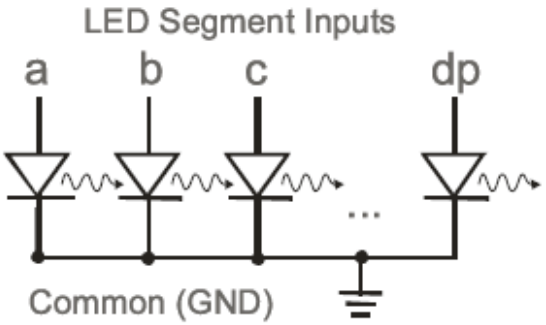


Fig.5 Common cathode 7-segment display circuit

7) Using IC7447: It has common collector outputs which can sink much more current than they can withdraw from source (supply). Therefore, the 7447 is designed for a common anode type display which needs logic '**0**' to turn on the segments. If you have a common cathode display, then you must use an inverter on each input. The inverter should be able to supply enough current to light a segment. Be sure to include a description of the circuit operation in your report.

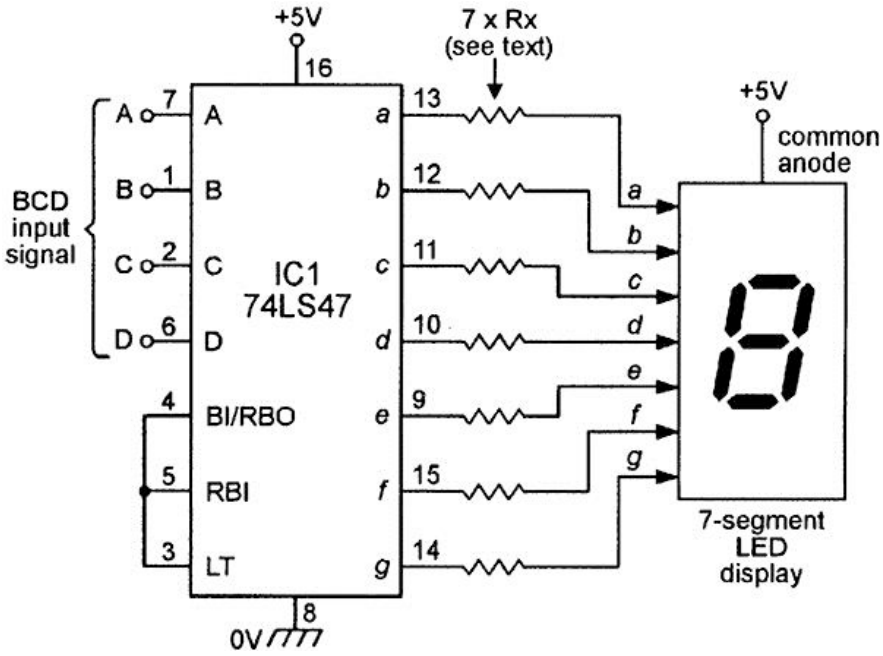


Fig.6 IC7447 BCD to 7-segment display

8) Make sure that no resistor leads are shorted anywhere and that the power to the seven-segment display is connected to the correct pin. **Do not connect** any pin of the common anode seven-segment display directly to ground. This will short out the segment forever. Likewise, do not connect any pin of the common cathode display to + 5 V or it to will be shorted out forever. Check the six unused input combinations (1010 through 1111) and report which segments light up. Does this match what you would expect from the seven equations you got for the decoder? If not, can you think of one reason why the output might not match your equations?

Result: As discussed in lecture

Experiment 9

Aim: To design a 4-bit odd parity generator and checker

Apparatus required: IC's 7402, 7486, 4077, 74266, connecting wires, breadboard, power supply etc.

Theory: Parity Generator: An extra bit is assigned to any bit-binary number for safe communication between different channels. The purpose of this extra bit is to make whole binary number as an even or odd. At the receiver end the bits are counted with respect to known nature of sender circuit, i.e., even or odd. If somehow there is an error in the communicated binary than output does not match as per need of an even parity checker or odd-parity checker circuit. For example- 1011 is communicated with even parity generator circuit than the flow is 1011 **1** where, last bit is the parity bit making whole number indicating even **1**'s. Now if some short-circuit converts it into 1111 than parity bit association 1111 **1** makes it to read like **odd** however, parity bit is high '**1**' in even parity circuit. That counts for a problem! One has to demand sender to again communicate this set of information. Following truth table simplifies the process of designing 4-bit odd-parity generator circuit through simplification via K-map:

BCD input				Parity bit
b3	b2	b1	b0	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

The Karnaugh Map for 4-bit odd parity generator circuit is shown as,

b1b0 / b3b2	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

Procedure:

1) **Parity Generator**

In this first part of the experiment, a 4-bit odd parity generator will be designed. Such a circuit will take four input bits (*b0, b1, b2, b3*) and produce one output bit which is parity bit (P). Following is the first Boolean function written using 8 high values from K-map:

$$P = b3b2b1b0 + \overline{b3}\,\overline{b2}\,\overline{b1}\,\overline{b0} + b3b2\overline{b1}\,\overline{b0} + \overline{b3}\,\overline{b2}b1b0 + b3\overline{b2}\,b1\overline{b0} + b3\overline{b2}\,\overline{b1}\,b0 + \overline{b3}\,b2\,b1\overline{b0} + \overline{b3}\,b2\,\overline{b1}\,b0$$

Above Boolean expression is simplified to:

$$P = (\overline{b1 \oplus b0})\ (\overline{b3 \oplus b2}) + (b1 \oplus b0)\,(b3 \oplus b2)$$

$$P = \overline{(b0 \oplus b1) \oplus (b2 \oplus b3)}$$

The circuit diagram for this parity generator is shown below, Fig.1:

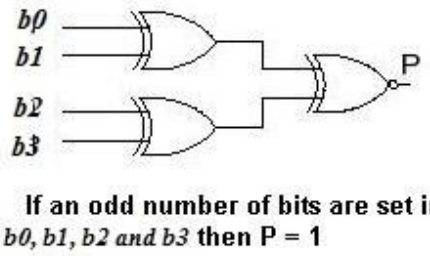


Fig.1 Odd parity generator for 4-bit signal

2) **Parity Checker:** This will be a 5-bit even parity generator circuit where 5-bit input contains 4 already propagated bits (*b0, b1, b2, b3*) along with generated 1 parity bit (P). The output bit is an indicative of error if occurs in between initial and final stage referring to parity bit. Whenever output error bit is high '1' there is an error in the communicating the right bit. The circuit diagram for this parity checker circuit is shown below in Fig.2.

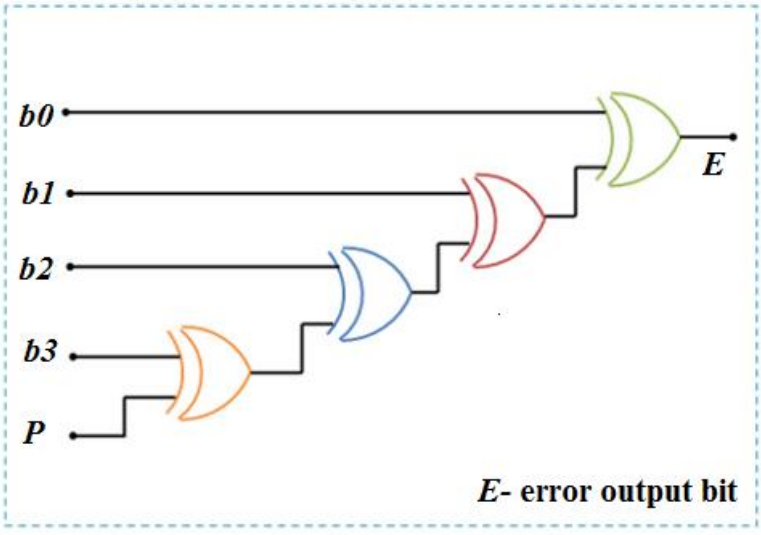


Fig.2 Even parity checker for 4-bit signal

Truth table for Parity checker 4-bit data communication with 5th bit parity input:

BCD input				Parity bit	Types of Checker	
					Error bit	Error bit
b3	b2	b1	b0	P	E (even)	E (odd)
0	0	0	0	1	1	0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	0	1	1	0	1
0	0	1	0	0	1	0
0	0	1	0	1	0	1
0	0	1	1	1	1	0
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	0	1	0	0	1
0	1	1	0	1	1	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0
0	1	1	1	1	0	1
1	0	0	0	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	0	0	0	1
1	1	0	1	0	1	0
1	1	0	1	1	0	1
1	1	1	0	0	1	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0
1	1	1	1	0	0	1
		Correct data				Error in data

Critical survey of truth table dictates that if anything happens to a particular data, the parity generation circuit will change parity bit and stage-2 parity checker receives a wrong parity and Error bit is high '***1***'. Someone may prefer to have parity checker stage-2 like even parity generation circuit or odd parity generation circuit. Just there will be a change in output function from XOR or XNOR. Knowing that XNOR is odd parity function and XOR is even parity function.

Result: Submit verified truth table for parity generator and checker.

Experiment 10

Aim: To exercise fabrication and working of Set-Reset latch and Set-Reset Flip-Flop.

Apparatus required: Bread board, IC's 7400, 7402, LED's, power supply, connecting wires

Theory: A latch is a basic memory circuit that can store 1 or 0 corresponding to static input followed by a feedback. Such a circuit based on single input inverter NOT gates is shown in Fig.1. Latches are useful in storing a particular state as memory like 0 or 1 or both by taking two different latches as one latch can store one state. The digital memories are of two types: static and dynamic. Static memories are those which keep a particular memory state retained (with applied electric power) forever till it is lost due to thermal or mechanical fluctuation. Dynamic memories are those which retain a particular state for stipulated duration (with applied electric power) and become ready for re-use after a particular time for storing earlier or a new state. These memories are also affected by thermal or mechanical fluctuation. Latches and gates are the basic components of static memory and are also the building blocks for sequential logic circuits, Fig.2. Flip-Flops are latches with enable/clock input, Fig.3. These are basic logic circuits element used for the storing both state high '1' as well as state low '0'. Some flip-flops are operated asynchronously; most are operated under clock control in a synchronous system.

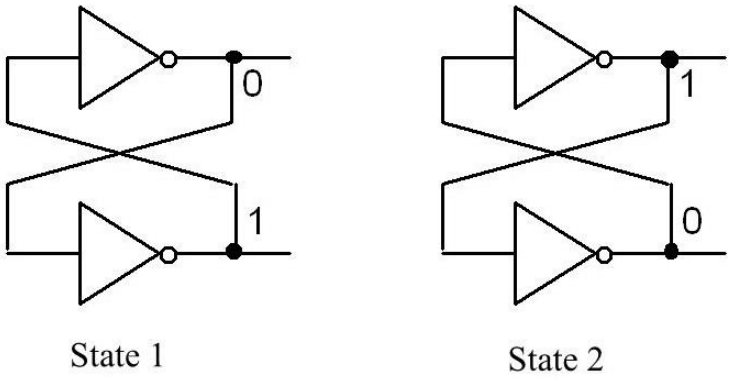


Fig.1 Symbol for NOT Latch

The truth tables for designing and operating Set-Reset latches and Set-Reset Flip-Flop are shown further. The term Set is abbreviated as '**S**' and term Reset is abbreviated as '**R**'. Knowing input variables S- and R- can be seen NOT to be sufficient for operating a particular latch as it needs another unknown input. For confirming second unknown input the variables must be changed and truth table of basic NAND or NOR gates suggest an optimum condition when a particular output state can be actually confirmed. For example- in case of a NAND latch, when at least one input is low '0' the output of that gate will surely be high '1' similarly in case of a NOR gate when at least one input is high '1' than output of that gate will surely be low '0', see truth tables of NAND and NOR gates.

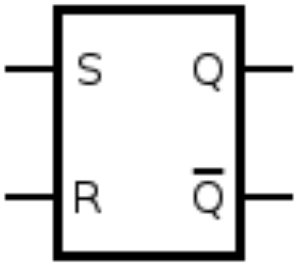


Fig.2(a) Symbol for S-R Nand Latch

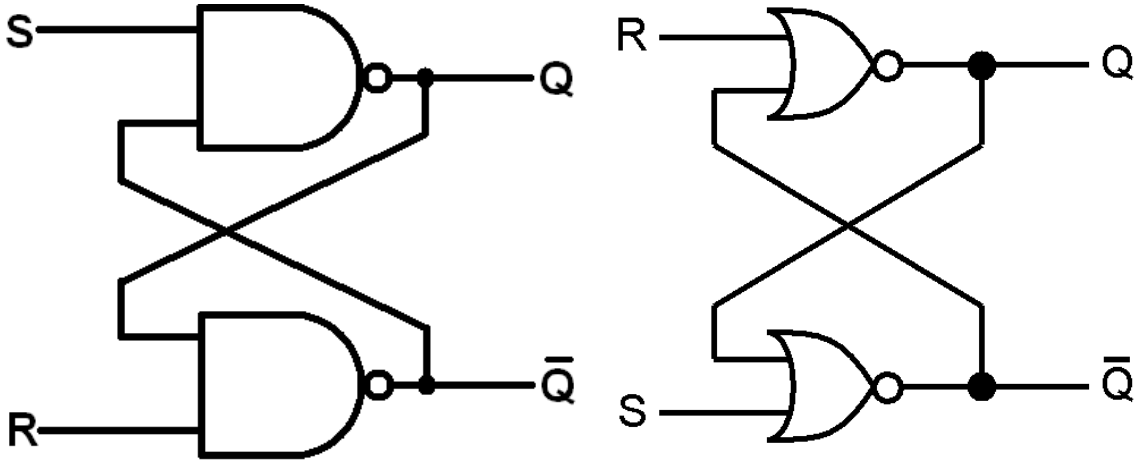


Fig.2(b) S-R NAND and NOR Latches

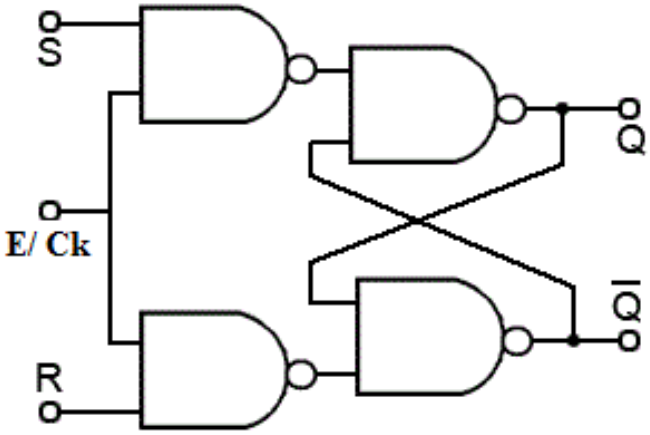


Fig.3 S-R NAND Flip-Flop

Truth tables of NAND and NOR gates:

NAND gate		
A	**B**	**Y**
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate		
A	**B**	**Y**
0	0	1
0	1	0
1	0	0
1	1	0

Characteristic Table-1 to be verified for S-R NOR Latch (SET Latch):

S	**R**	**Q**	**$\bar{Q}$**
0	1	0	1
0	0	Memory as before	
1	0	1	0
0	0	Memory as before	

Characteristic Table-2 to be verified for S-R NAND Latch (RESET Latch):

S	**R**	**Q**	**$\bar{Q}$**
0	1	1	0
1	1	Memory as before	
1	0	0	1
1	1	Memory as before	

Characteristic Table-3 to be verified for S-R NAND Flip-Flop:

E or Ck	**S**	**R**	**Q**	**$\bar{Q}$**
0	X	X	Memory	
1	0	0	Memory	
0	0	1	0	1
1	1	0	1	0
1	1	1	Indeterminate	

PROCEDURE and PRECAUTIONS:

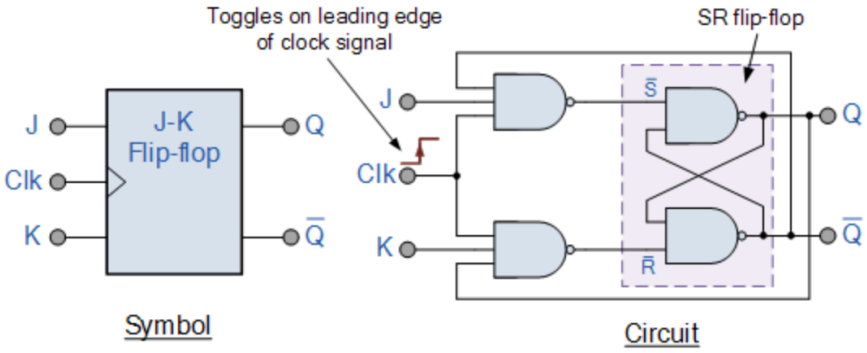
1. Arrange necessary connections on bread board corresponding to each of the tables 1-3 and circuits as shown in Figs.2(b) and 3.
2. Provide necessary input to the gate and check for the output bits by choosing LED's.
3. **Do not exercise indeterminate state as first step.**
4. If a pre-existing high value appears corresponding to expected low '0' than short the terminal to ground before re-use.
5. Use enable input as static dc pulse first than attempt with low frequency square pulse secondly.

RESULT: Set-Reset latch and Flip-Flop working is verified.

Experiment 11: To exercise working of J-K flip-flop with clock input

Implement the JK flip flop using logic gates, and verify its working. For clock pulses you may use the function generator set to produce square wave of desired frequency (~ 1 to 4 Hz).

The Basic JK Flip-flop



The Truth Table for the JK Function

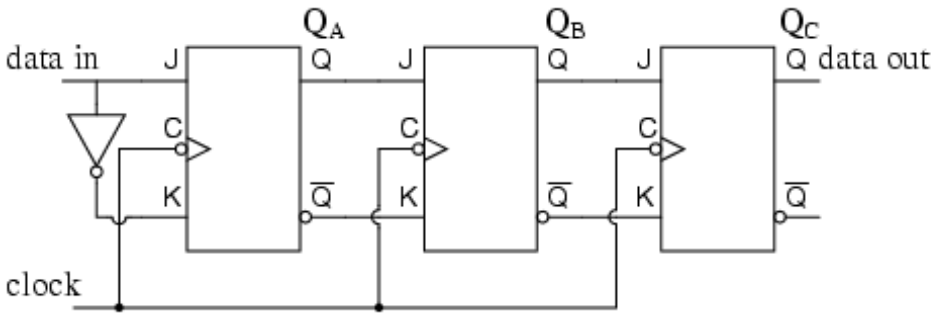
	Input		Output		Description
	J	K	Q	$\overline{Q}$	
same as for the SR Latch	0	0	0	0	Memory no change
	0	0	0	1	
	0	1	1	0	Reset Q » 0
	0	1	0	1	
	1	0	0	1	Set Q » 1
	1	0	1	0	
toggle action	1	1	0	1	Toggle
	1	1	1	0	

Source:

https://www.electronics-tutorials.ws/sequential/seq_2.html

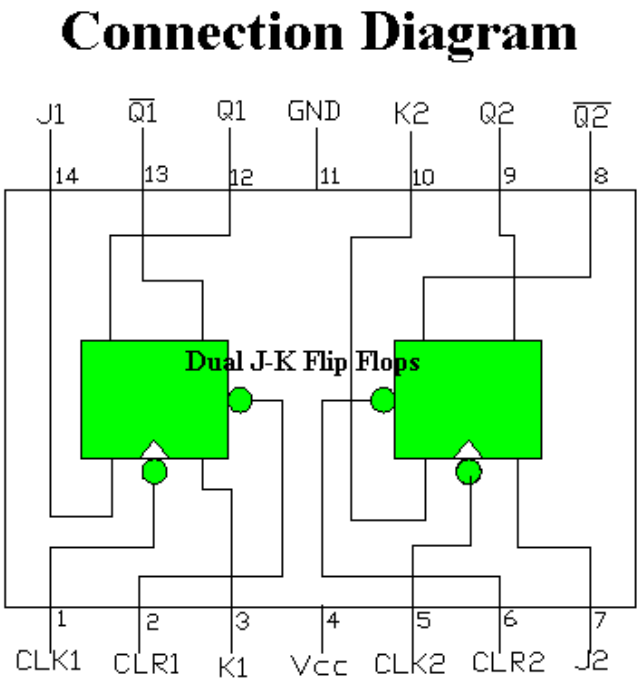
Experiment 12: To design shift register using J-K flip-flop and verify its working

Implement the following circuit using IC 7473 (Dual JK flip-flop). Use 5V supply for the IC (Pin4: 5V, Pin11: 0V). To see the outputs Q_A, Q_B use LEDs with ~1K resistors.



Serial-in, serial-out shift register using type "JK" storage elements

IC 7473 pinout and function:



Function Table

Inputs				Outputs	
CLR	CLK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	⎍	L	L	Q_0	$\overline{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	Toggle	

H = HIGH Logic Level
L = LOW Logic Level
X = Either LOW or HIGH Logic Level
⎍ = Positive pulse data. the J and K inputs must be held constant while the clock is HIGH. Data is transferred to the outputs on the falling edge of the clock pulse.
Q_0 = The output logic level before the indicated input conditions were established.
Toggle = Each output changes to the complement of its previous level on each HIGH level clock pulse.

Sources:

https://www.ibiblio.org/kuphaldt/electricCircuits/Digital/DIGI_12.html
http://makeyourownchip.tripod.com/7473.html

Experiment 13

Aim: To exercise fabrication and working of 4-bit UP/DOWN Asynchronous counter based on J-K Flip Flop for routine counting and selective counting: a) counting ***up*** from 1-16, b) counting ***down*** from 16-1 and c) counting ***up*** from 3-8.

Apparatus required: Bread board, IC's 7400, 7410, 7414, 7473, 7476, LED's, power supply, DSO, connecting wires

Theory: Counting system utilizes the toggling feature of J-K Flip-Flop by synchronizing delay time of individual flip-flops with applied clock pulse. The counting systems can be derived in two ways on the basis of method of applying clock pulse for driving the flip-flops stages. : First one is based on applying clock only at first flip-flop and using output of this as clock for next flip flop and so on for all other necessary stages. The second one is based on applying clock pulse simultaneously to all present necessary flip-flop stages. The first one is known as Asynchronous counter and second one is known as Synchronous counter. Asynchronous counters are also known as Ripple counters. These two varieties differ from each other on number of issues like speed, complication in circuit and need for flow of counting. For all counting operations modified form of race around condition known as toggling is used. Toggling in J-K flip-flop is obtained by cascading another flip-flop stage. The cascading arranges increased propagation delay to match clock pulse and racing gets converted to toggling in a controlled manner. For this operation the default input state is J=1 and K=1. A typical 4-bit asynchronous counter circuit (negative edge triggered FF system) is shown in Fig.1:

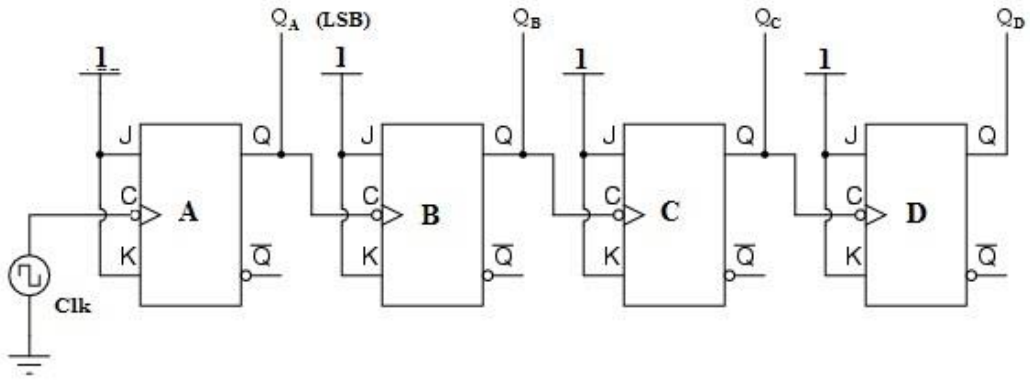


Fig.1 4-bit Asynchronous UP counter

These counters can also be used for reverse counting, i.e., from 15-0 or 16-1. For this, there are two possible approaches either collect outputs from $\overline{Q_A}$ not from Q_A as shown in Fig.1 or collect outputs similarly in up counter after arranging clock pulse from $\overline{Q_A}$ for driving all future flip-flop stages. This can be further understood by looking at the output stage flow table no.1 along with wave form picture as Fig.2. The first possibility of circuit is also shown in Fig.3:

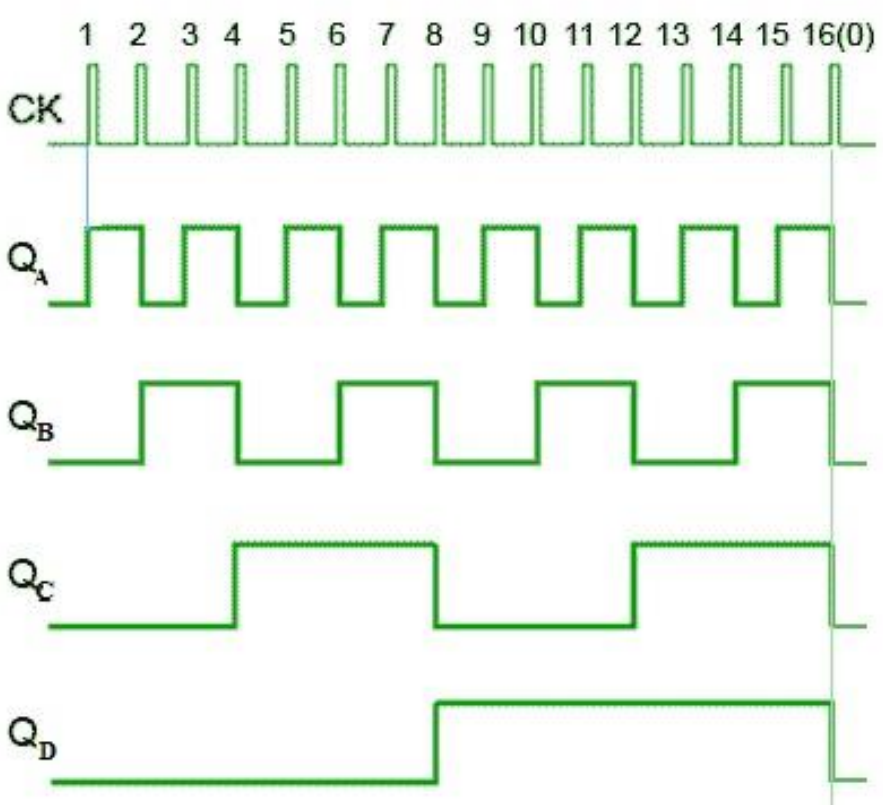


Fig.2 Flow of output stages in positive edge triggered 4-bit asynchronous counter

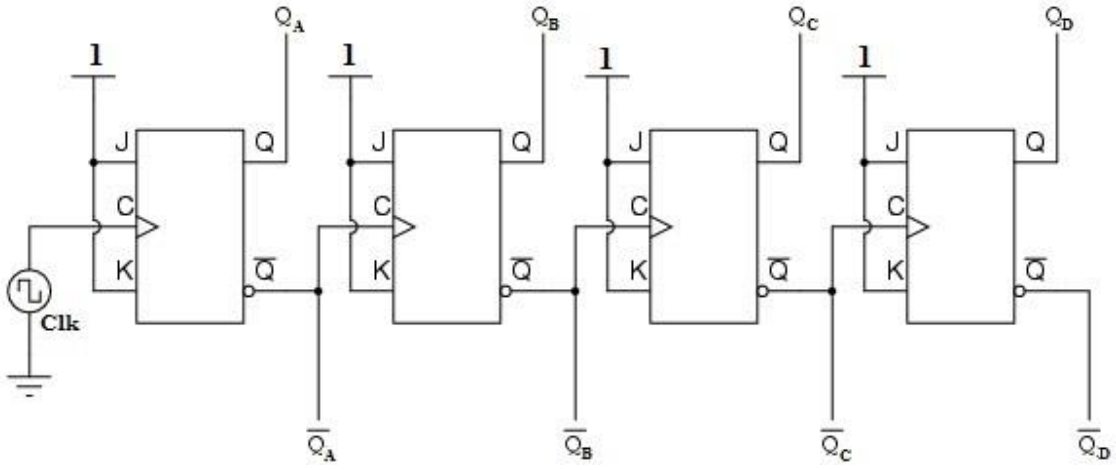


Fig.3 Circuit for 4-bit Asynchronous DOWN counter

Table-1 Flow of output stages in 4-bit asynchronous Up/Down counters taking pulse-1 for 0000

Clk	Q_A	Q_B	Q_C	Q_D	$\overline{Q_A}$	$\overline{Q_B}$	$\overline{Q_C}$	$\overline{Q_D}$
1	0	0	0	0	1	1	1	1
2	1	0	0	0	0	1	1	1
3	0	1	0	0	1	0	1	1
4	1	1	0	0	0	0	1	1
5	0	0	1	0	1	1	0	1
6	1	0	1	0	0	1	0	1
7	0	1	1	0	1	0	0	1
8	1	1	1	0	0	0	0	1
9	0	0	0	1	1	1	1	0
10	1	0	0	1	0	1	1	0
11	0	1	0	1	1	0	1	0
12	1	1	0	1	0	0	1	0
13	0	0	1	1	1	1	0	0

14	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
15	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
16	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**

Two additional inputs are used in counters compared to flip-flops; these are ***Pre-set (PR)*** and ***Clear (CLR)***. The use of these inputs is to provide flip-flops stages a freedom to become ready for any suddenly appearing output stage for example- a usual 4-bit counter capable of counting from 0-15 or 1-16 can also count in the middle anywhere like 3-8 and so on. For such an operation, two actions are desired, a) the counting must go up to 8 and as 9 appears it should remove all 4 outputs immediately to 0,0,0,0 this is known as ***clearing*** the stages and b) output must be forced to become 0,0,1,1 rather than 0,0,0,0 which is actually 3 in decimal, this is known as ***pre-setting*** output stages. The integrated circuit pin diagram having these two additional inputs is shown in Fig.4:

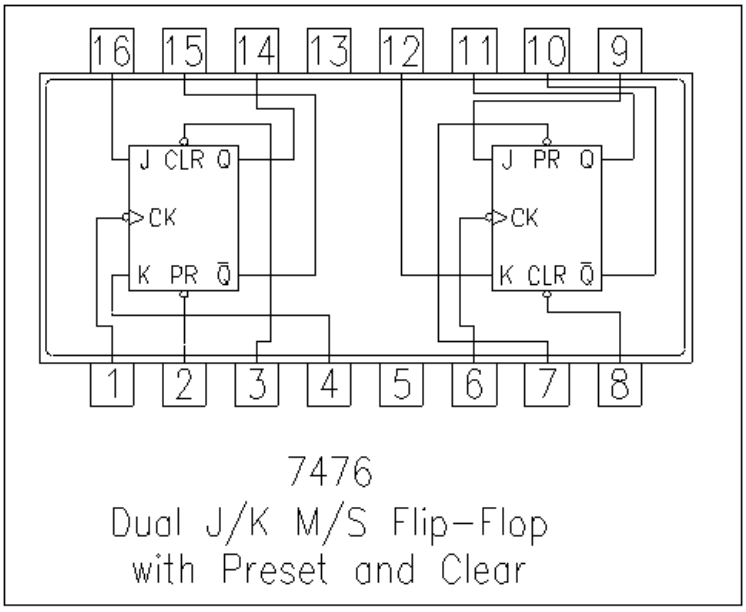


Fig.4: J-K flip-flop IC with Pre-set and Clear inputs

A typical output waveform picture of a negative edge triggered 4-bit asynchronous counter is shown in Fig.5. The selective counting utilizes pre-set and clear inputs and output stages are according to a designed sequence 3-8 or any other as planned. The sequence of ***up*** counting from 3-8 is highlighted in Table-2. That shows the need of clear input at stage 1001 to arrange all 4 flip-flop stages to 0000 and then to over write these 4 flip-flop stages with outputs 0011 using pre-set input. The pre-set input demands an inverted value provided through a 2-input NAND gate driven primarily by output stages Q_A and Q_B.

Table-2 Flow of output stages for selective counting from 3-8 (taking pulse-1 for output 0000 stage)

Clk	Q_A	Q_B	Q_C	Q_D	$\overline{Q_A}$	$\overline{Q_B}$	$\overline{Q_C}$	$\overline{Q_D}$
1	0	0	0	0	1	1	1	1
2	1	0	0	0	0	1	1	1
3	0	1	0	0	1	0	1	1
4	1	1	0	0	0	0	1	1
5	0	0	1	0	1	1	0	1
6	1	0	1	0	0	1	0	1
7	0	1	1	0	1	0	0	1
8	1	1	1	0	0	0	0	1
9	0	0	0	1	1	1	1	0
10	1	0	0	1	0	1	1	0
11	0	1	0	1	1	0	1	0
12	1	1	0	1	0	0	1	0
13	0	0	1	1	1	1	0	0

14	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
15	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**
16	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**

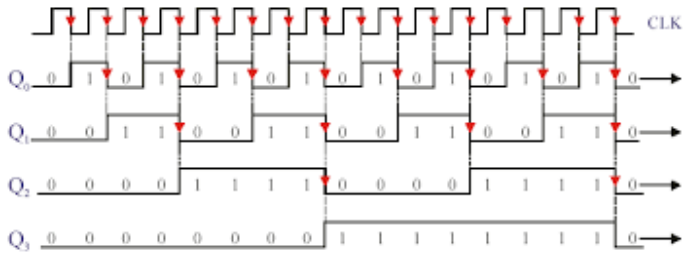


Fig.5 Output stages in 4-bit asynchronous counter driven by negative edge triggering

Procedure:

1. Connect circuit as shown in Fig.1 and take utmost care for introducing LED's for $\mathbf{Q_{n+1}}$ and $\mathbf{\overline{Q}_{n+1}}$.
2. Arrange ***J*** and ***K*** as static input values and dynamic clock input from DSO of the same or slightly lower magnitude.
3. Confirm the output waveform as shown in Fig.2 and save it for result submission.
4. Experience Up/down counting as per circuits shown in Fig.1 and 3.
5. Switch off the circuit and arrange dynamic ***J*** and ***K*** inputs from DSO along with pre-existing clock.
6. Confirm the selective counting outputs as shown in Table-2 using 7476 and a NAND gate IC.

RESULT: Save obtained wave forms for all three targets and submit result analysis.

Experiment 14

Aim: To fabricate and realize 2- and 3-bit Synchronous UP counter.

Apparatus required: Bread board, IC's 7400, 7408, 7432, 7404, 7402, 7473, LED's, power supply, connecting wires.

Theory: In synchronous counters the clock inputs are provided to all the flip-flops parallel and simultaneously triggered also. As a result, all flip-flop stages become ready to deliver output and there is no propagation delay. Thus, the maximum operating frequency for this counter will be significantly higher than for the corresponding ripple counter. In a typical 2-bit synchronous counter at a time four possible output states get generated, as shown below in table-1. These states follow routine sequence and change very fast compared to asynchronous counting. Table-2 is the excitation table for this system and simplifying K-maps for known parameters Q1, Q2, Q1,n and Q2,n four unknowns J1, K1, J2 and K2 can be obtained. A 3-bit synchronous counter is shown in Fig.2 with its excitation table as Table-3. The J and K inputs of FF-1 are connected to HIGH. FF-2 has its J and K inputs connected to the output of FF-1, and the J and K inputs of FF-3 are connected to the output of an AND gate that is fed by the outputs of FF-1 and FF-2. After the 3rd clock pulse both outputs of FF-1 and FF-2 are HIGH. The positive edge of the 4th clock pulse will cause FF-3 to change its state due to the AND gate.

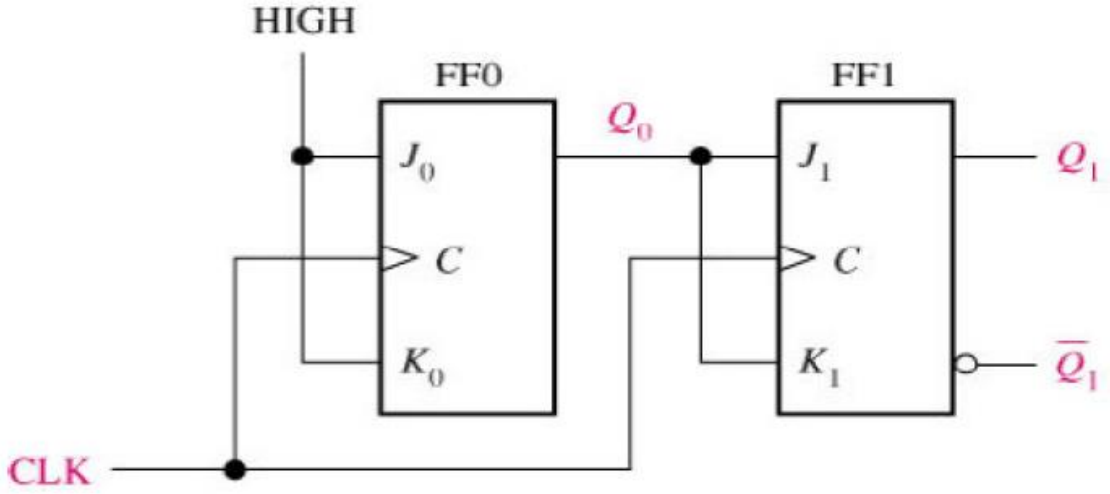


Fig.1: Logic diagram of 2-bit Synchronous Counter (source: Google)

Table-1 Excitation table for 2-bit synchronous counter

	Q1	Q2	Q1,n	Q2,n	J1	K1	J2	K2
State-0	**0**	**0**	**0**	**1**	**0**	**d**	**1**	**d**
State-1	**0**	**1**	**1**	**0**	**1**	**d**	**d**	**1**
State-2	**1**	**0**	**1**	**1**	**d**	**0**	**1**	**d**
State-3	**1**	**1**	**0**	**0**	**d**	**1**	**d**	**1**

Solving four K-maps for four unknowns following results are obtained:

J1=Q2 K1=Q2 J2=1 K2=1

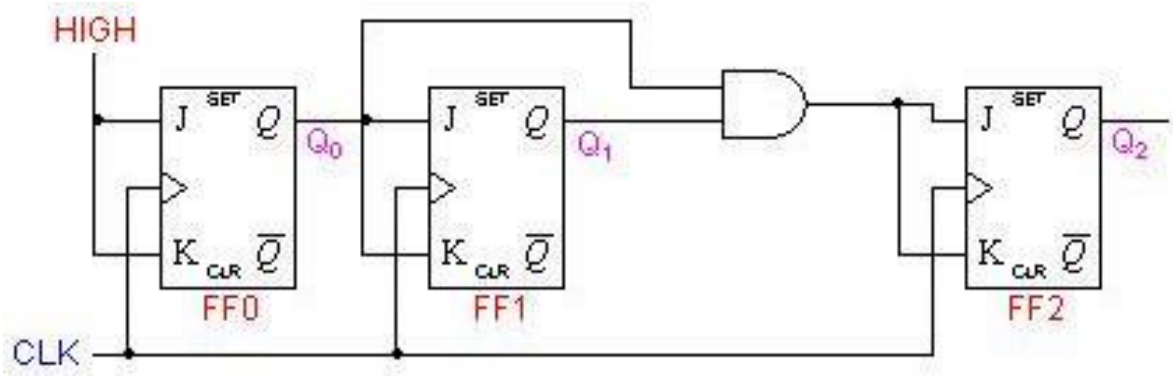


Fig.2: Logic diagram of 3-bit Synchronous Counter (source: Google)

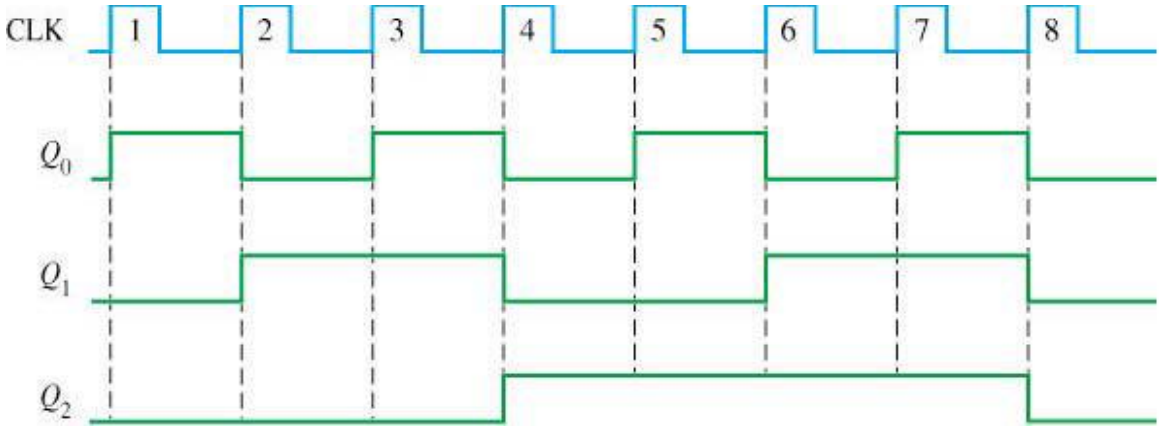


Fig.3: Timing Diagram of 3-bit Counter (source: Google)

Table-2 Excitation table for 3-bit Synchronous Counter

Present State			Next State			T3		T2		T1	
Q3	Q2	Q1	Q3,n	Q2,n	Q1,n	J3	K3	J2	K2	J1	K1
0	0	0	0	0	1	0	0	0	d	1	d
0	0	1	0	1	0	0	0	1	d	d	1
0	1	0	0	1	1	0	0	d	0	1	d
0	1	1	1	0	0	1	d	d	1	d	1
1	0	0	1	0	1	d	0	0	d	1	d
1	0	1	1	1	0	d	0	1	d	d	1
1	1	0	1	1	1	d	0	d	0	1	d
1	1	1	0	0	0	d	1	d	1	d	1

Simplifying K-maps for three unknowns T1, T2, and T3 using above excitation table following expressions are obtained:

T1=1 T2=Q1 T3=Q2Q1

Procedure:

1. Construct the logic circuit as shown in Fig.1 and 2.
2. Use the specific output for selecting process of counting either up or down.
3. Use dc pulse though switch to verify role of clock and count demonstrated through LED's should propagate.
4. Verify dynamic clock pulse response as shown in Fig.3.

Result: Thus the 2- & 3-bit synchronous up counters are designed and verified.