

Department of Electronic Engineering

City University of Hong Kong

EE 2000 Logic Circuit Design

Laboratory Manual

Laboratory Sessions

LAB1	Design a two-digit BCD adder with Multimedia Logic	(1 week)
LAB2	Implementation and Analysis of an Arithmetic Logic Unit	(1 week)
LAB3	Implementation of a sequence generator and recognizer	(2 weeks)

Log Book

Students are required to prepare their own log book (A4-size). Lab tutors and lab demonstrators will check and sign the log book at the end of each laboratory session. Students could use the log book to record all findings, data, analysis, preparation materials, discussions, and experimental results.

Check point

There are some check points in lab exercises. Students are required to demonstrate their own results to the lab tutors and the lab demonstrators.

The Lab Report

Students are required to submit an individual lab report in Week 13 (to their lab tutors). All reports should be typed and A4-size blinded. Contents of the lab report should include:

1. Objectives:

State the purposes of each exercise and the achievements in this lab. Please do not copy the objectives verbatim from the lab handout.

2. Design

Describe your design with the circuit diagram. Explain the functionalities of each circuit. Please make sure all figures and tables are consistent, legible and well labeled.

3. Results and Discussions:

Report and describe what you have observed in the lab. Present your results in a clear and concise manner.

4. Conclusions:

In this section you should attempt to answer the questions: What did you learn from this lab? What did you do wrong (or what went wrong)? How could you have improved upon your design procedures? Were your results as expected or did you find something unusual. Try not to include information that you have included in previous sections. Present the significance of your results conceptually, if applicable.

EE2000 Logic Circuit Design

Experiment 1: Design a two-digit BCD adder with Multimedia Logic

Software Tool: Multimedia Logic

Download website: <http://www.softronix.com/logic.html>

Download the “Setup Kit” and install the software.

Watch the tutorial video “*Multimedia Logic - Tutorial 1 - Simple Circuits*” from youtube:

<http://www.youtube.com/watch?v=hJq2gECXYWc>

You may also read the *Introduction of the Multimedia Logic* for more thorough information about this tool.

Objective: Design and implement a 2-digit BCD adder using Multimedia Logic

Principle: Detect the 4-bit ALU output for the condition of {10,11,12,13,14,15}. Add “0” to the result if “false”, or add “6” to the result if “true”.

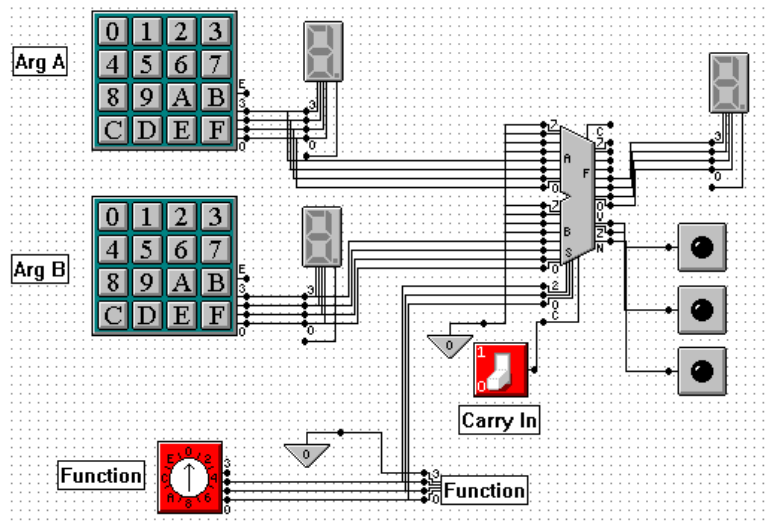
An example ALU.LGI contains circuit for performing the arithmetic and logic operations on two 4-bit inputs.

Open the file: c:\Program file\Softronics\Multimedia Logic\Examples\Devices\ALU.LGI

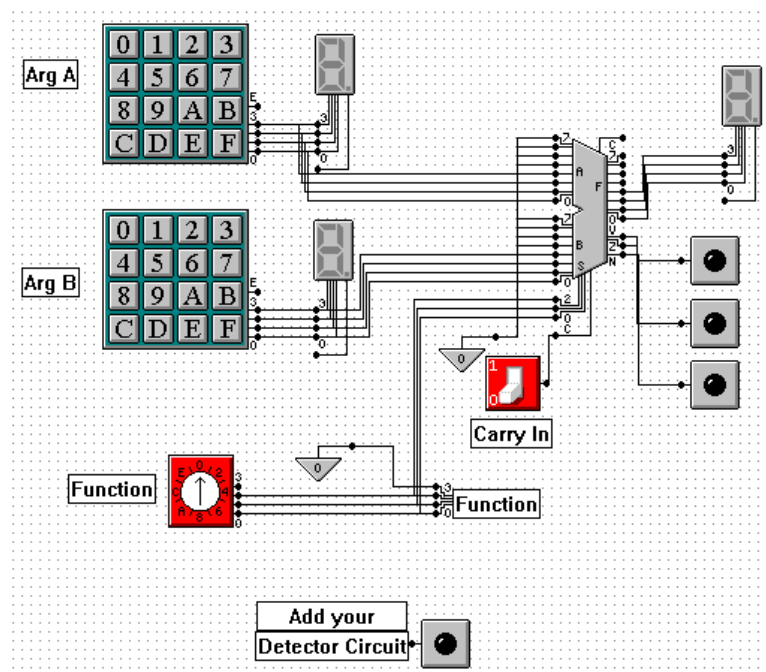
This example uses an arithmetic logic unit to perform various arithmetic and logic operations on two 8-bit inputs. The objective of this experiment is to modify this circuit to perform addition for 2 BCD digits.

Procedures:

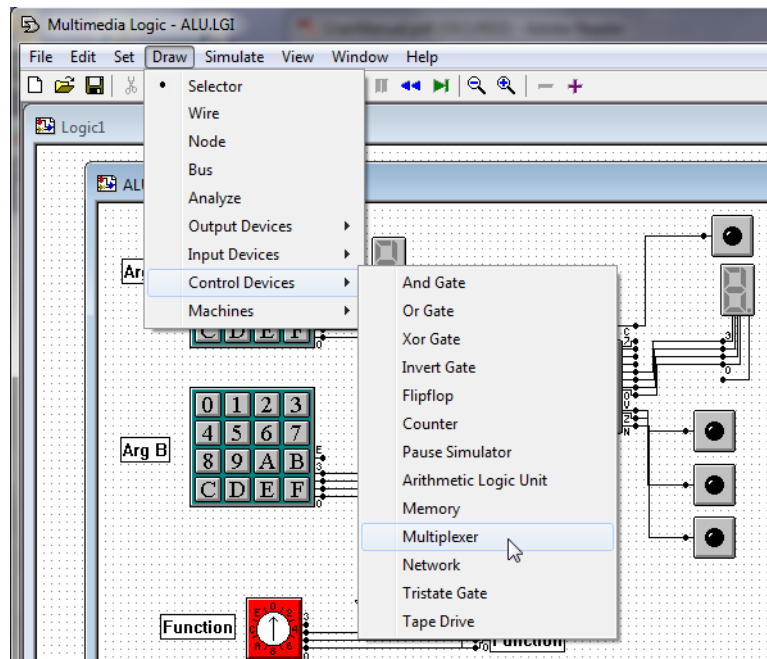
- Step 1: Tabulate the detector output to check the ALU output, and use K-map to determine the logic function for the detector.
- Step 2: Delete the un-used components from the ALU.LGI example to obtain the basic circuit as shown in the following diagram:



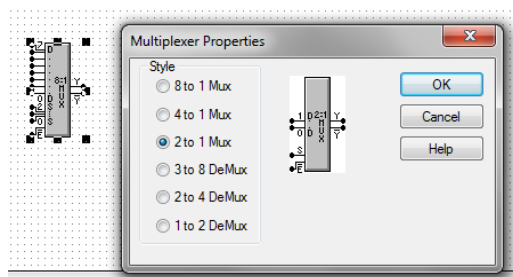
Step 3: Add the detector circuit to the circuit. Verify the function of the detector (use a LED-ON to indicate the “true” condition). **(This is a check point, proceed if complete this part.)**



Step 4: Use 4 multiplexers to select “add-0” or “add-6” to the 4-bit result.



Modify the properties of the device to a 2-to-1 Mux as shown below:

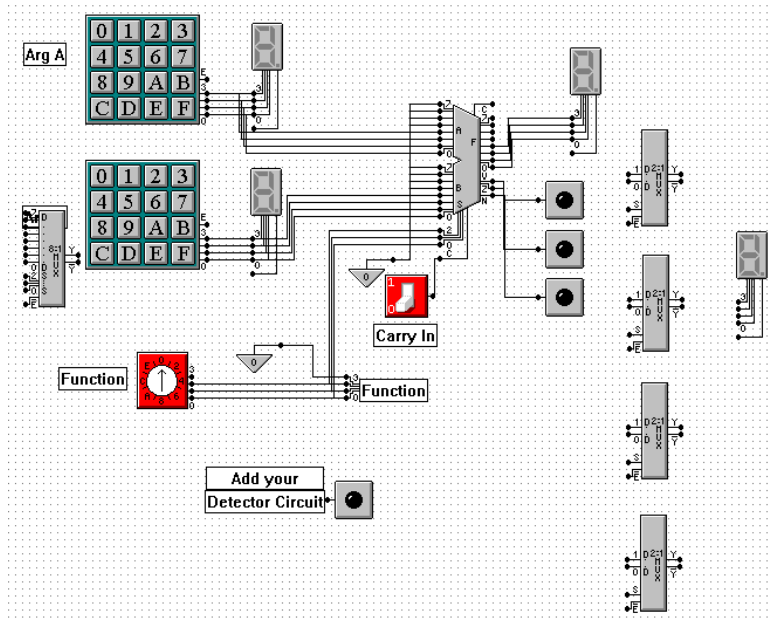


The operation of the multiplexer is as follows:

E	S	Y
0	0	D_0
0	1	D_1
1	X	Hi-Z
1	X	Hi-Z

The detector output is used as the selection input “S” to the multiplexers, i.e. the four “ D_0 ” inputs are “0” and the four “ D_1 ” inputs are “6”.

Add 4 multiplexers, a 7-segment LED to the circuit and complete the connections. Verify the 7-segment LED output, i.e. a “0” for “false” and a “6” for “true” condition. **(This is a check point, proceed if complete this part.)**



Step 5: Add an ALU to the circuit and complete the connections. A group of 4-bit input is from the original ALU and another group of 4-bit input is from the 4 multiplexers output. Add 2 LEDs to indicate the final SUM of the BCD adder.

Verify the circuit by testing with different BCD (0,1,2,...,9 only) inputs with ranges of sum varied from 0 to 15.

EE2000 Logic Circuit Design

Experiment 2: Implementation and Analysis of an Arithmetic Logic Unit

Objective: Analysis and implementation an Arithmetic and Logic Unit (ALU) for BCD addition.

An ALU is circuit for performing specific arithmetic and logic functions.

The experiment requires 4 students to achieve the final result. Each student is required to implement a 1-bit ALU to form the basis of the final circuit. Four 1-bit units form a 4-bit ALU unit. The result will be displayed on a 7-segment LED.

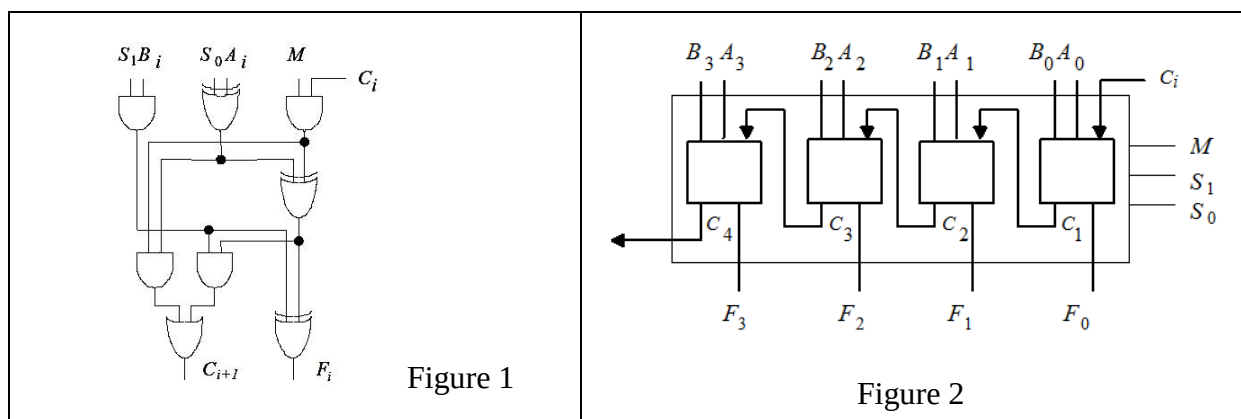
With additional detection and ALU (74LS181) circuits, the 4-bit ALU can be modified to perform addition of 2 BCD digits. The result will be displayed on two 7-segment LEDs.

Group size: 4 students

Components:

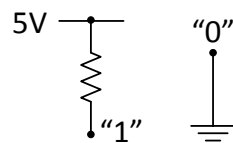
5V USB transformer		4
USB power cable		4
Breadboard		4
2-input AND gate	74LS08	5
2-input OR gate	74LS32	4
2-input XOR gate	74LS86	4
2-to1 Multiplexer	74LS157	1
4-bit ALU	74LS181	1
7-segment display driver	74LS47	3
7-segment LED display (Common Anode)		3
LEDs		10
1 k Ω resistor		10
330 Ω resistor (for 7-segment LED)		21

Figure 1 shows a 1-bit ALU circuit and Figure 2 shows a 4-bit ALU based on four 1-bit ALU modules. The 1-bit ALU module performs 4 logical functions and 8 arithmetic functions.



Procedures:

Use the following circuit to produce logic “1” and logic “0” as input to the logic circuit.



IMPORTANT: Never direct connect to 5V to obtain logic “1”, the logic gate IC may easily be damaged.

(Steps 1 and 2 are individual work)

Step 1: Analyse the operation of the one-bit ALU module by completing the following truth table.

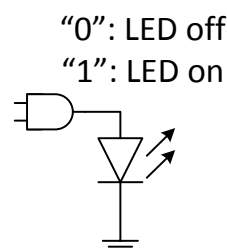
$M = 0$ Logical function				
F_i				
$S_1 S_0$				
$A_i B_i$	00	01	10	11
0 0				
0 1				
1 0				
1 1				

$M = 1$ Arithmetic function								
F_i					C_{i+1}			
$S_1 S_0$					$S_1 S_0$			
$C_i A_i B_i$	00	01	10	11	00	01	10	11
0 0 0								
0 0 1								
0 1 0								
0 1 1								
1 0 0								
1 0 1								
1 1 0								
1 1 1								

From the above truth table, deduce the ALU operation by completing the following table. (You may need to observe more bits to determine the functions.)

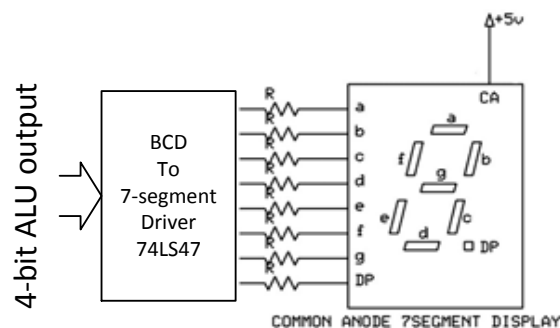
	$M = 0$ Logical function	$M = 1$ Arithmetic function	
$S_1 S_0$	F_i	$F_i (C_i = 0)$	$F_i (C_i = 1)$
0 0	$F_i = A_i$	$F = A$	$F = A \text{ plus } 1$
0 1	$F = \bar{A}$	$F = \bar{A}$	$F = \bar{A} \text{ plus } 1$
1 0	$F_i = A_i \oplus B_i$	$F = A \text{ plus } B$	$F = A \text{ plus } B \text{ plus } 1$
1 1	$F_i = \bar{A_i} \oplus B_i$	$F = \bar{A} \text{ plus } B$	$F = \bar{A} \text{ plus } B \text{ plus } 1$

Step 2: Each student is responsible for implementing a one-bit ALU module and verify their circuit operation against the truth table. Connect a LED to the output for easily observing the logic output. (You may connect the LED directly to the output circuit.)



(Steps 3 and 4 are group work)

Step 3: 4 students combine their own 1-bit ALU to form a 4-bit ALU circuit. The four outputs of the 4-bit ALU should be connected to a 7-segment LED driver (74LS47) with its output connected a 7-segment LED. Verify the circuit operation against the Function Table 1.



Step 4: According to the result of Experiment 1 and with the use of a standard 4-bit ALU (74LS181), design a 1-digit BCD adder circuit. Draw the circuit diagram. Implement the adder and verify its operation.

Step 5: Combine two (or more) 1-digit BCD adders (with other groups) to form a multi-digit BCD adder. Verify the circuit operation.

EE2000 Logic Circuit Design

Experiment 3: Implementation of a sequence generator and recognizer (2 weeks)

Objective: (i) To design and implement a sequential logic circuit to generate a pre-defined 4-bit repeated sequence (1st week), and (ii) to design and implement a sequential logic circuit for recognizing the sequence pattern generated by the circuit in part (i).

Group size: 1 student. Each student will be assigned with a different a 4-bit pattern for this experiment.

Components:

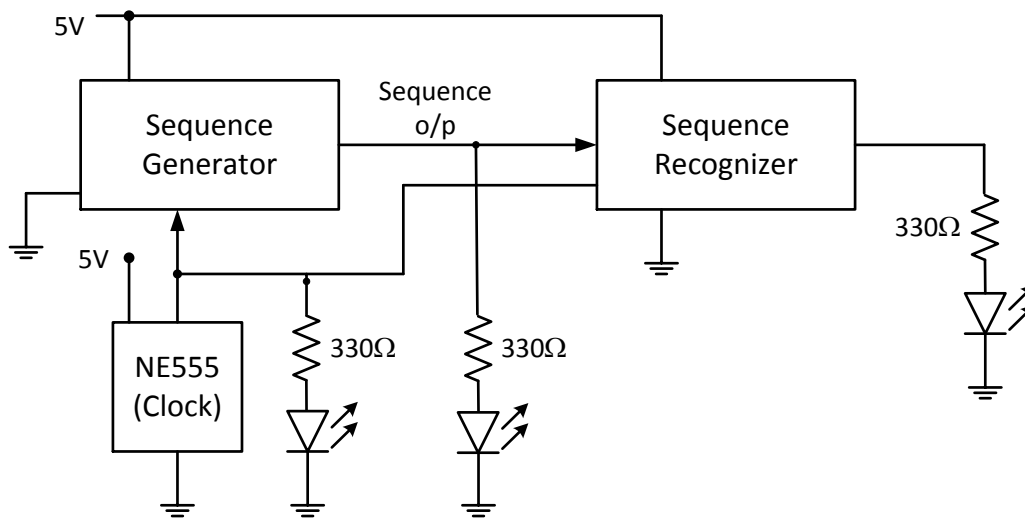
5V USB transformer		1
USB power cable		1
Breadboard		1
Timer IC	NE555	1
D-FF	74LS74	1
LEDs		3
1 k Ω resistor		1
220 k Ω resistor		2
2.2 μ F capacitor		1
Other logic gates	(subject to different design	
(See Appendix 2)	– obtained from the component trays in the laboratory)	

(You may use 74LS175 – quad D-FF IC instead of the 74LS74, the dual D-FF IC)

Part 1 – Sequence Generator (1st week)

- Step 1: The sequence generator design is based on D-FF. According to the sequence given by the laboratory supervisor, follow the standard procedures: (i) tabulate the state table, (ii) binary state assignment, (iii) tabulate the design table; and (iv) determine the FF input logic function.
- Step 2: Draw the circuit diagram.
- Step 3: Implement the circuit on the breadboard.
- Step 4: The sequence generator synchronize with a clock signal which is produced by a timer IC NE555. The clock signal generator circuit is given in Appendix 1. Implement this circuit on the same breadboard.
- Step 5: Connect a LED to the output of NE555 and a LED to the sequence generator output so that the LED will be “ON” for logic “high” and OFF for logic “low”. In this way, the output state change for each clock period can be observed easily and to correctness of circuit operation can easily be verified. An illustration of the system block diagram is shown in the following diagram.

Demonstrate your result to laboratory supervisor !!



Part 2 – Sequence Recognizer (2nd week)

- Step 1: The sequence recognizer design is based on D-FF and the sequence is the one used in Part 1. Follow the standard procedures to design the circuit: (i) draw the state diagram, (ii) tabulate the state table, (iii) simplify the state table by means of state reduction, (iv) binary state assignment, (v) tabulate the design table; and (vi) determine the FF input logic function.
- Step 2: Draw the circuit diagram.
- Step 3: Implement the circuit on the breadboard.
- Step 4: Connect the sequence generator output to the sequence recognizer input and verify the correctness of its operation.

Demonstrate your result to laboratory supervisor !!

- Step 5: Test your sequence recognizer by connect its input to other sequence generator (implemented by other students) output to verify its operation.

Appendix 1

astable operation

As shown in Figure 12, adding a second resistor, R_B , to the circuit of Figure 9 and connecting the trigger input to the threshold input causes the timer to self-trigger and run as a multivibrator. The capacitor C will charge through R_A and R_B and then discharge through R_B only. The duty cycle may be controlled, therefore, by the values of R_A and R_B .

This astable connection results in capacitor C charging and discharging between the threshold-voltage level ($\approx 0.67 \cdot V_{CC}$) and the trigger-voltage level ($\approx 0.33 \cdot V_{CC}$). As in the monostable circuit, charge and discharge times (and therefore the frequency and duty cycle) are independent of the supply voltage.

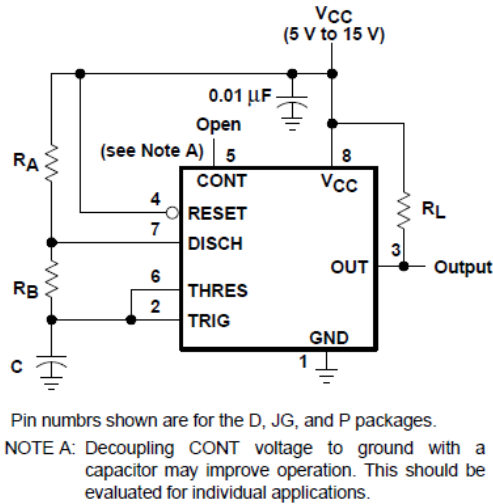


Figure 12. Circuit for Astable Operation

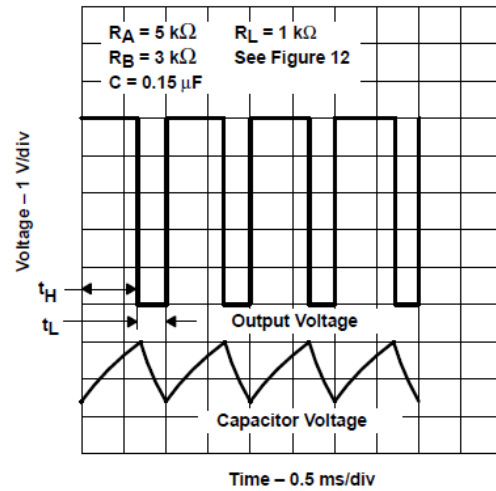


Figure 13. Typical Astable Waveforms

Figure 13 shows typical waveforms generated during astable operation. The output high-level duration t_H and low-level duration t_L may be calculated as follows:

$$t_H = 0.693 (R_A + R_B) C$$

$$t_L = 0.693 (R_B) C$$

Other useful relationships are shown below.

$$\text{period} = t_H + t_L = 0.693 (R_A + 2R_B) C$$

$$\text{frequency} \approx \frac{1.44}{(R_A + 2R_B) C}$$

$$\text{Output driver duty cycle} = \frac{t_L}{t_H + t_L} = \frac{R_B}{R_A + 2R_B}$$

$$\text{Output waveform duty cycle} = \frac{t_H}{t_H + t_L} = 1 - \frac{R_B}{R_A + 2R_B}$$

$$\text{Low-to-high ratio} = \frac{t_L}{t_H} = \frac{R_B}{R_A + R_B}$$

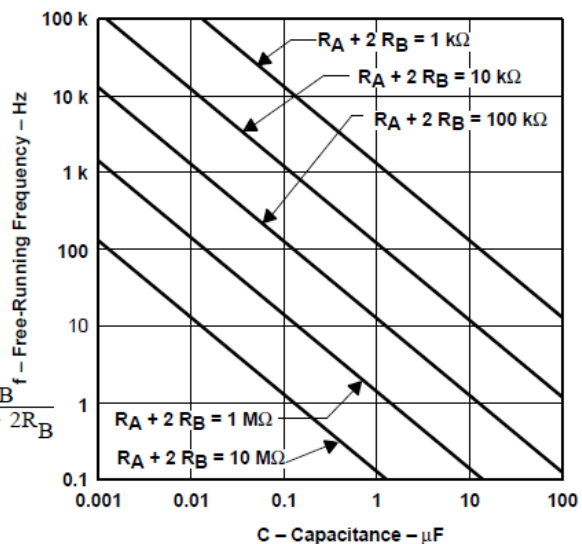


Figure 14. Free-Running Frequency

In order to allow an easy observation of the sequence, the frequency is set to around 1 Hz. The value of the component as listed in the following:

$$R_L = 1 \text{ k}\Omega, \quad R_A = 220 \text{ k}\Omega \quad R_B = 220 \text{ k}\Omega \quad C = 2.2 \text{ }\mu\text{F (electrolytic capacitor)}$$

Appendix 2

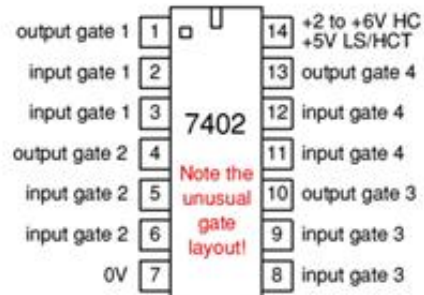
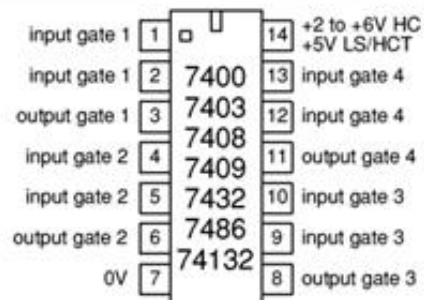
For datasheet, use keyword “TTL 74LS?? pdf” to find them from google.

Quad 2-input gates

- 7400 quad 2-input NAND
- ~~7402 quad 2-input NAND with open collector outputs~~
- 7408 quad 2-input AND
- ~~7409 quad 2-input AND with open collector outputs~~
- 7432 quad 2-input OR
- 7486 quad 2-input EX-OR
- ~~74132 quad 2-input NAND with Schmitt trigger inputs~~

The 74132 has [Schmitt trigger](#) inputs to provide good noise immunity. They are ideal for slowly changing or noisy signals.

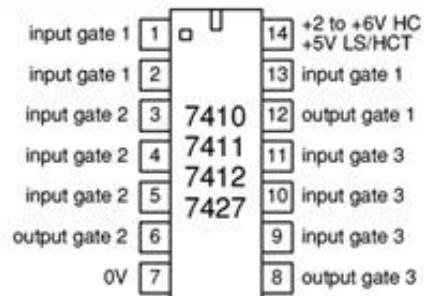
- 7402 quad 2-input NOR
Note the unusual gate layout.



Triple 3-input gates

- 7410 triple 3-input NAND
- 7411 triple 3-input AND
- ~~7412 triple 3-input NAND with open collector outputs~~
- 7427 triple 3-input NOR

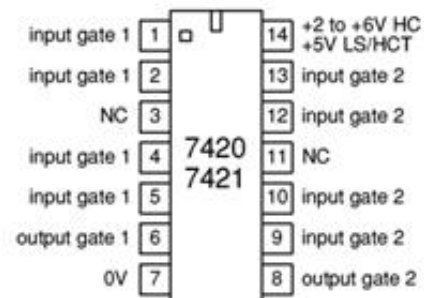
Notice how gate 1 is spread across the two sides of the package.



Dual 4-input gates

- 7420 dual 4-input NAND
- 7421 dual 4-input AND

NC = No Connection (a pin that is not used).



Hex NOT gates

- 7404 hex NOT
- ~~7405 hex NOT with open collector outputs~~
- ~~7414 hex NOT with Schmitt trigger inputs~~

The 7414 has [Schmitt trigger](#) inputs to provide good noise immunity. They are ideal for slowly changing or noisy signals.

