LABORATORY MANUAL

Analysis and Design of Digital Circuits Lab (21EC34)
SEMESTER: III

Name: _____

USN : _____

Compiled by
1. Dr. Hemalatha J N
2. Mrs. Raja Vidya

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
RV COLLEGE OF ENGINEERING®
(AUTONOMOUS UNDER VTU)
BENGALURU-560059.

RV College of Engineering®, Bengaluru-560059

(Autonomous Institution affiliated to VTU, Belgaum)

Department of Electrical and Electronics Engineering

VISION

Attain technical excellence in Electrical and Electronics Engineering through graduate programs and interdisciplinary research related to sustainability in power, energy and allied fields.

MISSION

1. To provide technical education that combines rigorous academic study and the excitement of innovation enabling the students to engage in lifelong learning.

2. To establish Center of Excellence in sustainable electrical energy, smart grids, and systems.

3. To establish tie-ups with industries and institutions of repute and to foster building up of a wide knowledge base to keep in tune with upcoming technologies.

4. To motivate commitment of faculty and students to collate, generate, disseminate, preserve knowledge and to work for the benefit of society.

5. To develop simple, appropriate and cost-effective inclusive technologies which are instrumental in the up-liftment of rural society.

RV College of Engineering®
Bengaluru-560059

(Autonomous Institution affiliated to VTU, Belgaum)

Department of Electrical and Electronics Engineering

Laboratory Certificate

This is to certify that Mr. / Ms._____ has satisfactorily completed the course of Experiments in Practical **Analysis and Design of Digital Circuit(21EC34)** prescribed by the Department of Electrical and Electronics Engineering during the Academic year 2022-2023.

Name of the Candidate: _____

USN No.: _____Semester:_____

Marks	
Maximum	**Obtained**
50	

Marks in words	

Signatures:
Staff in-charge
1.
2.

Head of the Department

RV College of Engineering®, Bengaluru-560059

(Autonomous Institution affiliated to VTU, Belgaum)

Department of Electrical and Electronics Engineering

Analysis and Design of Digital Circuits (21EC34)

SCHEME OF CONDUCTION AND EVALUATION

CLASS: III SEMESTER CIE MARKS: (Max.): 50
YEAR: 2022 SEE: 3hrs SEE MARKS: (Max):50

Note:
a) Out of Twelve experiments, for eight experiments manual will be provided. Each of these would also include practice experiments. Last four experiments are Innovative Lab Experiments and Virtual lab activity and are compulsory.
b) Practice questions: Students should design the experiment in advance and demonstrate in the lab.

Particulars of the Experiments Performed

CONTENTS

Sl. No.	Date	EXPERIMENTS	Viva (03)	Data Sheet (02)	Results and Observations (05)	Total: (10)	Signature
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
Total							
TEST							
Total Marks for Laboratory:							

Experiment-01
Realization of Binary Adder and Subtractor

Aim: To design and implement half/full adder and half/full subtractor using NAND gates and IC-7483.

Components Required: ICs -74LS00,74LS86, 74LS83, Digital trainer kit, patch chords.
Note: Data Sheet should be referred for pin details.

Binary Adders/Subtractors: A digital binary adder is a digital device that adds two binary numbers and gives its sum in binary format along with output carry. There are two types of adders: half adder and full adder. Half adder is a single bit adder without input carry, whereas full adder is a single bit adder with input carry. Binary subtractors are used to find the difference between two binary numbers. Digital adders are mostly used in computer's ALU (Arithmetic Logic Unit) to perform addition. Digital calculators use adders for athematic addition. Micro controllers use adders in arithmetic additions, PC (program counter) and timers etc. Every device that uses some kind of increment or arithmetic process contains adders.

PART-1

1.a) Realization of Binary adder/ subtractor using universal gates and IC-7483.

(a) Half adder (HA):

(i) Design:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 **1**

Inputs		Outputs	
A	**B**	**S_O**	**C_O**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) 1-bit addition **(b) HA Function Table (TT)**

Fig. 1.1: Design of Half Adder

Simplified (K-map) design equations for the outputs Sum(S_O) and Carry(C_O) are in Eq. 1.1

$$S_O = \Sigma_m(1,2) = \overline{A}\,B + A\,\overline{B} = A \oplus B \text{ and } C_O = \Sigma_m(3) = A\,B \qquad \textbf{Eq. 1.1}$$

(ii) Implementation using NAND gates:

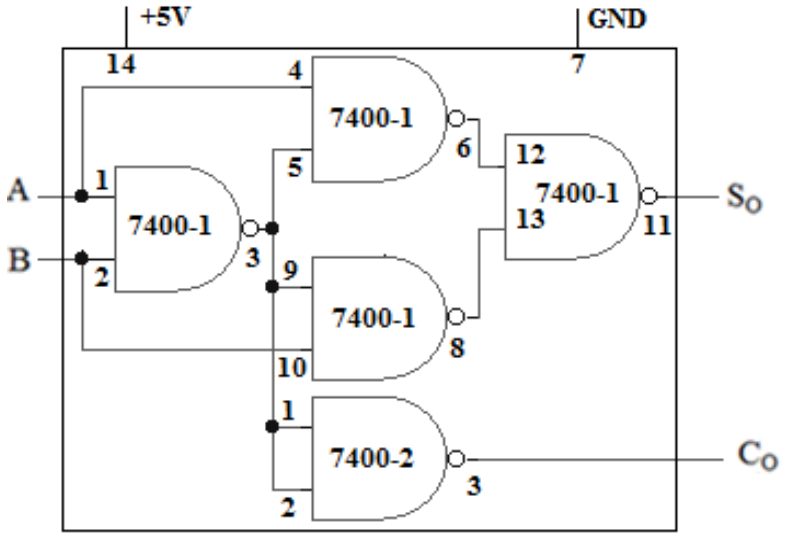


Fig. 1.2: Logic diagram of Half Adder using NAND gates

(iii) Observations:

Inputs		Outputs	
A	B	S_O	C_O
0	0		
1	0		
0	1		
1	1		

(b) Full Adder (FA):

(i) Design:

$0 + 0 + 0 = 0$
$0 + 0 + 1 = 1$
$0 + 1 + 0 = 1$

$0 + 1 + 1 = 0\ \ 1$
$1 + 0 + 0 = 1$
$1 + 0 + 1 = 0\ \ 1$
$1 + 1 + 0 = 0\ \ 1$
$1 + 1 + 1 = 1\ \ 1$

(a) 1-bit addition with carry input

Inputs			Outputs	
A	B	C_{in}	S_O	C_O
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

(b) FA Function Table (TT)

Fig. 1.3: Design of Full Adder

Simplified (K-map) design equations for the outputs Sum(S_O) and Carry(C_O) are in Eq. 1.2

$$S_O = \Sigma_m(1,2,4,7) = A \oplus B \oplus C_{in}$$
$$C_O = \Sigma_m(3,5,6,7) = AB + BC_{in} + AC_{in} \qquad \textbf{Eq. 1.2}$$

(ii) Implementation using NAND gates :

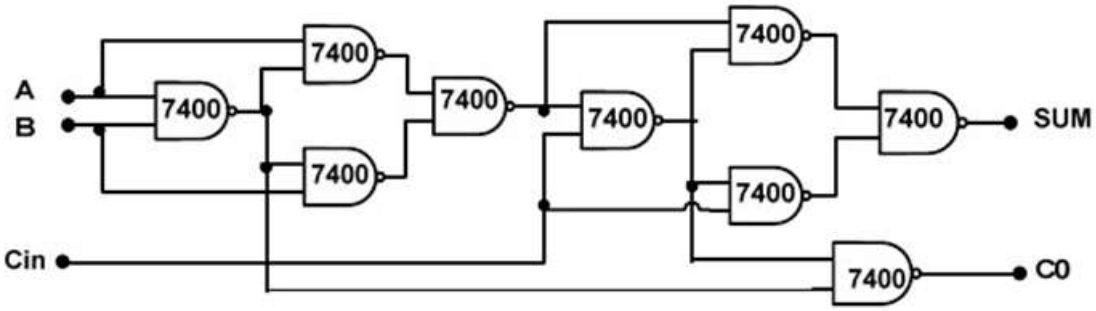


Fig. 1.4: Design of Full Adder using NAND gates

(iii) Observations:

Inputs			Outputs	
A	B	C_{in}	S_0	C_0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

(c) Half Subtractor (HS):

(i) Design:

0 - 0 = 0

0 - 1 = 1 1

1 - 0 = 1

1 - 1 = 0 0

(a) 1-bit subtraction

Inputs		Outputs	
A	**B**	**S_O**	**C_O**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

(b) HS Function Table (TT)

Fig. 1.5: Design of Half Subtractor

Simplified (K-map) design equations for the outputs Difference (D_O) and Borrow(B_O) are given in Eq. 1.3

$$D_O = \Sigma_m(1,2) = \overline{A}\,B + A\,\overline{B} = A \oplus B \text{ and } B_O = \Sigma_m(3) = \overline{A}\,B \qquad \textbf{Eq. 1.3}$$

(ii) Implementation using NAND gates:

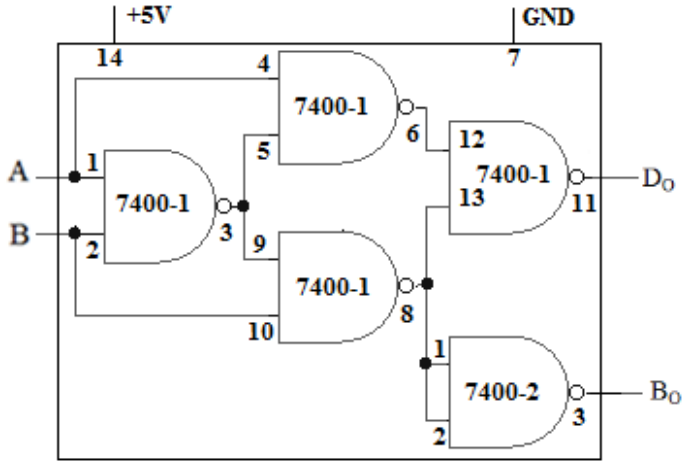


Fig. 1.6: Logic diagram of Half Subtractor using NAND gates

(iii) Observations:

Inputs		Outputs	
A	B	D_O	B_O
0	0		
0	1		
1	0		
1	1		

(d) Full subtractor (FS)

(i) Design:

0 - 0 - 0 = 0

0 - 0 - 1 = 1 1
0 - 1 - 0 = 1 1
0 - 1 - 1 = 0 1
1 - 0 - 0 = 1
1 - 0 - 1 = 0
1 - 1 - 0 = 0
1 - 1 - 1 = 1 1

(a) 1-bit subtraction with borrow input

Inputs			Outputs	
A	B	B_{in}	D_O	B_O
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

(b) FS Function Table (TT)

Fig. 1.7: Design of Full Subtractor

Simplified (K-map) design equations for the outputs Difference (D_O) and Borrow(B_O) are given in Eq. 1.4

$$D_O = \Sigma_m(1,2,4,7) = A \oplus B \oplus B_{in} \text{ and}$$
$$B_O = \Sigma_m(1,2,3,7) = \overline{A}\overline{B}B_{in} + \overline{A}\,B\,\overline{B_{in}} + \overline{A}\,B\,B_{in} + A\,B\,B_{in} \quad \textbf{Eq. 1.4}$$

(ii) Implementation using NAND gates:

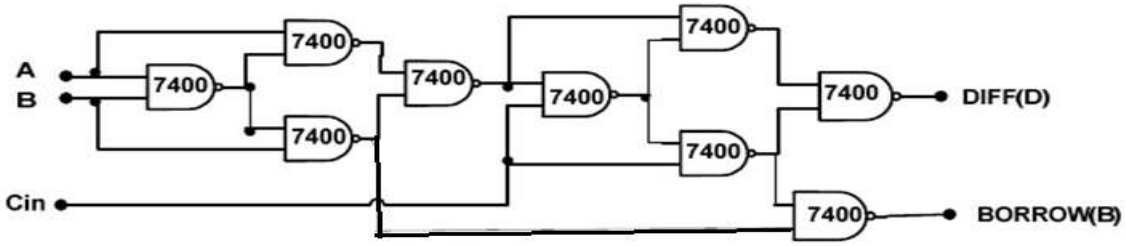


Fig. 1.8: Design of Full Subtractor using NAND gates

(iii) Observations:

Inputs			Outputs	
A	B	B_{in}	D_0	B_0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Procedure:

1. The circuit connections are made as shown in the diagrams (as per the designs).
2. V_{dd} and Gnd are connected to all the ICs used in the design.
3. The inputs are applied and the corresponding outputs are verified as in TTs.
4. The above procedure is applied to all the four circuits and the TTs are verified.
5. Conclusions/Inferences are derived from the conducted experiments.

(e) Parallel Adder/Subtractor:

Logic diagram of IC-74LS83:

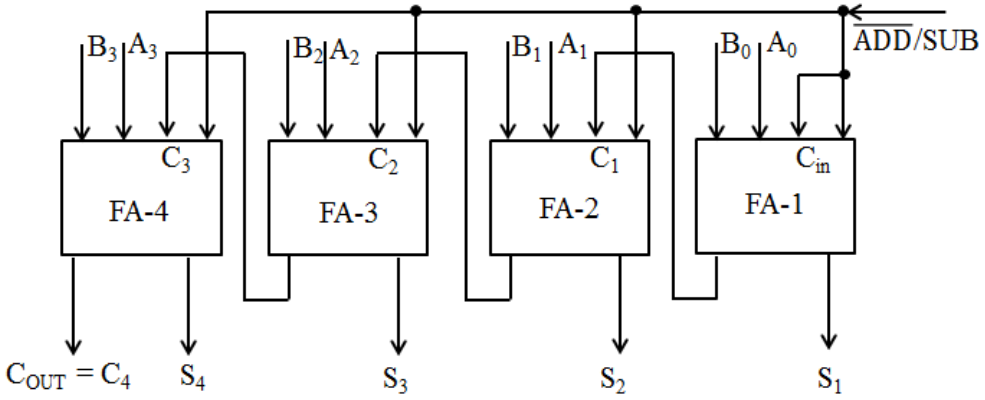
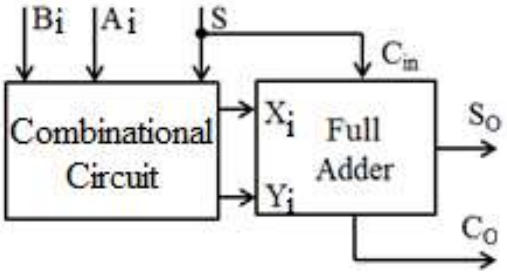


Fig. 1.9: Logic diagram of Parallel Adder

(i) Design:



S	X	Y	Cin
0	X	Y	0
1	X	Y′	1

S	A	B	X	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

Fig. 1.10: Algorithm for Adder/Subtractor design

Simplification using K-map results in the expressions of Eq. 1.5;

$X_i = A_i$, $Y_i = B_i$ **xor** S, and $C_{in} = S$. **Eq. 1.5**

The resulting combinational circuit is called the 'Controlled Inverter' as it produces 1s complement to realize 2s complement addition. Implementing the expressions in Eq. 1.5 along with a parallel adder results in a parallel adder/subtractor circuit shown in Fig. 1.11.

(ii) Implementation:

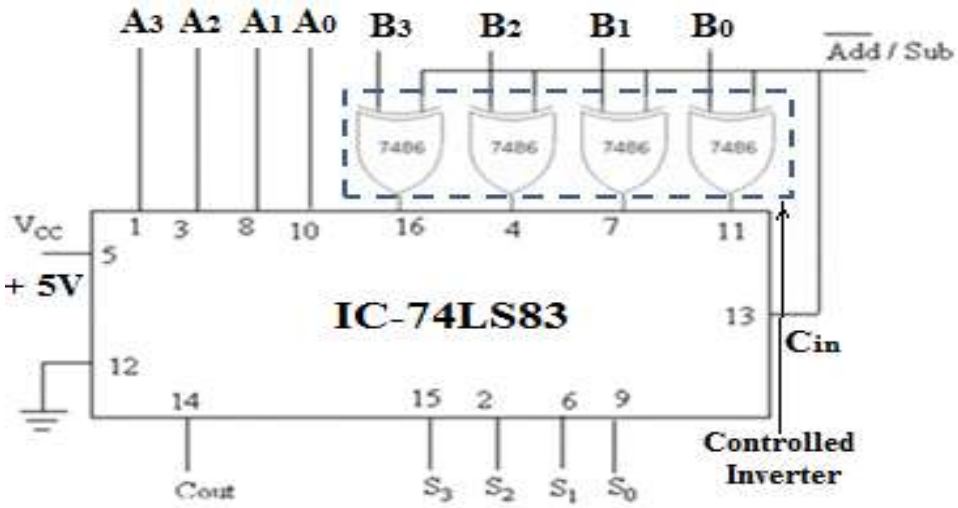


Fig. 1.11: Logic diagram of Parallel Adder

Procedure:

Four bit Parallel Adder:

1. Connections are made as shown in the circuit diagram of Fig. 1.11.
2. The control input $\overline{\text{Add}}/\text{Sub} = 0$, the inputs are applied and the adder outputs are noted for different inputs.
3. The procedure is repeated for different set of inputs and verified.

Four bit Parallel Subtractor:

1. Connections are made as shown in the circuit diagram of Fig. 1.11.
2. The control input $\overline{\text{Add}}/\text{Sub} = 1$,the inputs are applied, and the adder outputs are noted for different inputs.
3. The procedure is repeated for different set of inputs and verified.

(iii) Observations:

Note:

The adder IC-74LS83 performs addition of the minuend and 2's complement of subtrahend which is subtraction of the 2-binary numbers (C_{in} = '1')

PART 2

1.b)Practice Question: Design a parallel binary subtractor to get actual difference based on the value of C_{out}.

Inferences/Conclusions:

Staff Signature

Experiment-02

Arithmetic circuits- Realize the given Boolean expressions using MUX/DEMUX

Aim: To realize given Boolean expression using multiplexer and de-multiplexer.

Components Required: ICs - 7404, 74153, 74139, 7420,Digital trainer kit, patch cords.

Multiplexer and De-multiplexer /Decoders: In electronics, a multiplexer is a device that selects between several analog or digital input signals(2^n) and forwards it to a single output line based on n-selection lines. Whereas, demultiplexer routes single input to any of the output lines(2^n) based on n-selection lines. Typically, demux and decoders perform same operations. In digital circuits, these devices are used to realize logic functions.

PART 1

2.a) Realize the given Boolean expressions using MUX/DEMUX using IC-74153,IC-74139.

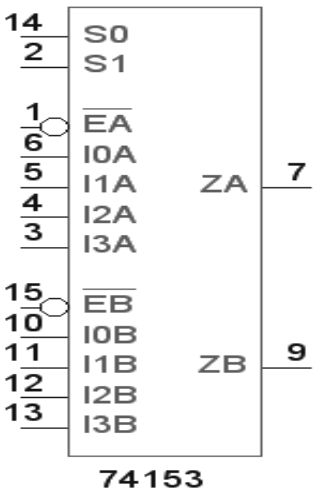
(a) Implementation of Boolean expression using MUX

(i) Design

The given Boolean expression for the implementation are,

F1(A,B,C)=∑m(1,2,4,7) and F2(A,B,C)=∑m(0,4,5,6)

Pin diagram of 74153



Implementation Table:

	I_0	I_1	I_2	I_3
F1 (MUXA)	A	A'	A'	A
***F2(*MUX B)**	1	A	A	0

(ii) Implementation

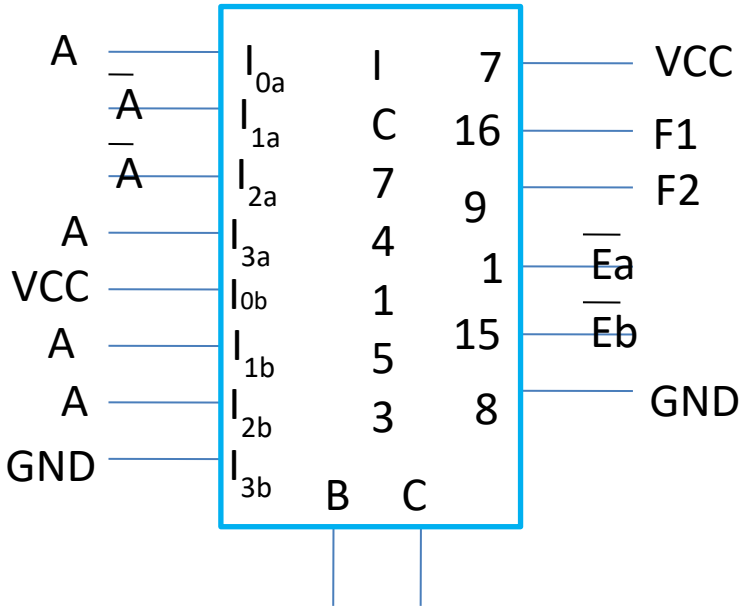


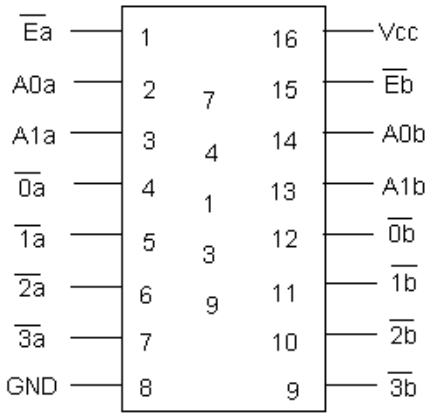
Fig.2.1 Implementation using IC-74153

(iii) Observations:

Inputs			Outputs	
A	B	C	F1	F2
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

(b) Implementation of Boolean expression using DEMUX

Pin Diagram of 74139



Function Table of IC-74139:

Decoder A

Ea	A1a	A0a	0a	1a	2a	3a
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Decoder B

Eb	A1b	A0b	0b	1b	2b	3b
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Note:Where E, A1 and A0 are Enable and Input lines respectively.

(i) Design:

The given Boolean expressions for the implementation are,

F1(A,B,C)=∑m(1,2,4,7) and F2(A,B,C)=∑m(0,4,5,6)

(ii) Implementation:

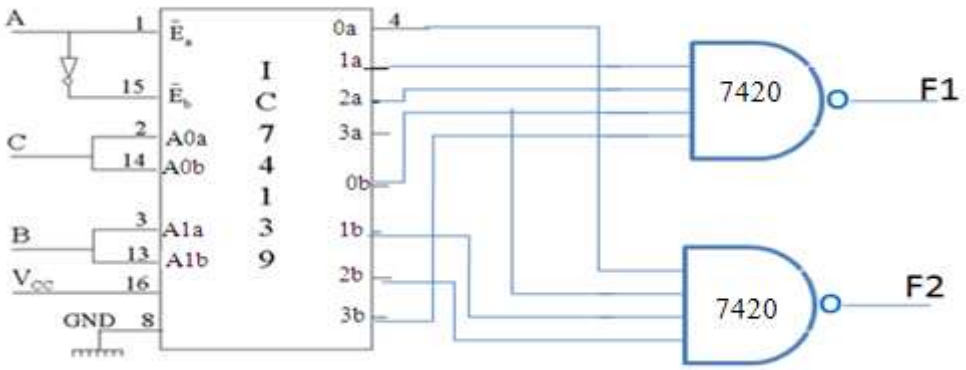


Fig.2.2 Implementation using IC-74139

Procedure:

1)Rig up the circuit using IC-74139 and NAND gates as shown in the design(Refer Fig.2.1, Fig.2.2)

2)Verify the output with the truth table Values.

(iii) Observation:

Inputs			Outputs	
A	B	C	F1	F2
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

PART 2

2.b)Practice Question: Realize FA/FS using MUX/DEMUX.

Inferences/Conclusions:

Staff Signature

Experiment-03
Code converters using DECODER/DEMUX

Aim: To design and implement code converters using DEMUX/DECODER

Components Required: IC74LS139, 74LS04,74LS10, 74LS20, 74LS30,74LS83 ,Digital trainer kit, patch chords.

Decoder/Demultiplexer: Decoder is logic circuits which has n-input lines and 2^n output lines. Decoder with Enable input can also be used as demultiplexer.

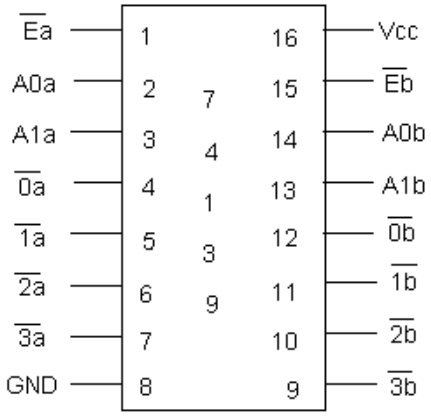
PART 1

3.a) Code convertors - Binary to Gray

(a) Binary to gray code conversion using IC-74139:

(i) Design:

Pin Diagram of IC- 74139:



Function Table of IC- 74139

Decoder A

$\overline{Ea}$	A1a	A0a	$\overline{0a}$	$\overline{1a}$	$\overline{2a}$	$\overline{3a}$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Decoder B

$\overline{Eb}$	A1b	A0b	$\overline{0b}$	$\overline{1b}$	$\overline{2b}$	$\overline{3b}$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Note: Where, E, A1 and A0 are Enable and Input lines respectively.

(ii) Implementation:

Referring the truth table of binary to gray code converter, the expressions represented in terms of minterms are as follows.

G0=∑m(1,2,5,6) , G1=∑m(2,3,4,5) and G3= B3.

The same can be implemented using 74139 as shown in circuit diagram below.

Realization of Binary to Gray code converter using IC- 74139.

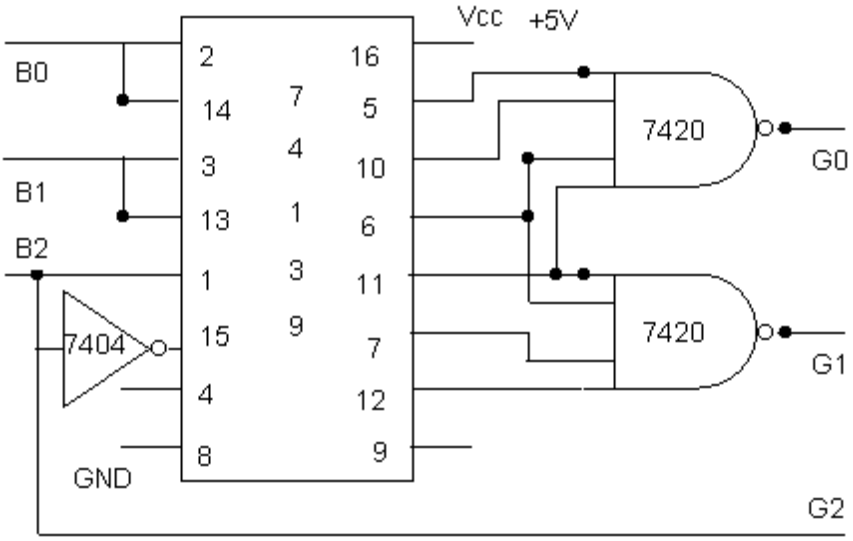


Fig.3.1 Implementation using IC-74139

(iii) Observations:

BINARY			GRAY		
B2	B1	B0	G2	G1	G0
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

PART-2

3.b) Practice Question: Gray to Binary using Decoder.

Inference/Conclusions:

Staff Signature

Experiment -04

Magnitude comparator & driving of LED display

Aim: Design of two bit magnitude comparator using logic gates.

Components required: IC 7400, IC7404, IC7410, IC7486,IC 7408, IC trainer kit, patch cords.

Magnitude comparator: A digital comparator or magnitude comparator is an electronic device that takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number.

PART-1

4.a) Design a two bit magnitude comparator using logic gates.

(i) Design:

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Fig4.1: Function Table of 2-bit Magnitude Comparator

Simplified (K-map) design equations for outputs A>B, A=B and A<B are in Eq.4.1.

$$(A > B) = \Sigma_m(4, 8, 9, 12, 13, 14) = A_1\overline{B_1} + A_0\overline{B_0B_1} + A_0A_1\overline{B_0}$$
$$(A = B) = \Sigma_m(0, 5, 10, 15) = (A_0 XNOR B_0).(A_1 XNOR B_1)$$
$$(A < B) = \Sigma_m(1, 2, 3, 6, 7, 11) = B_1\overline{A_1} + B_0\overline{A_0A_1} + B_0B_1\overline{A_0} \qquad \textbf{Eq.4.1}$$

(ii) Implementation:

Implementation of Eq.4.1 is as shown in Fig.4.2.

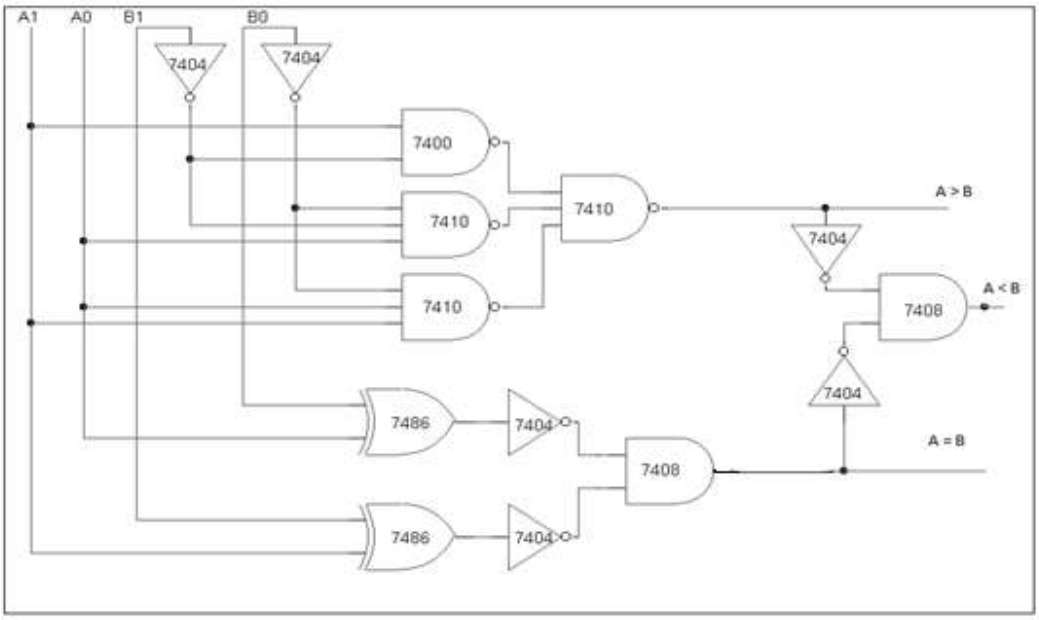


Fig.4.2: Logic diagram of 2-bit Magnitude Comparator

Note: From truth table, **(A=B)'. (A>B)'=(A<B)**

iii)Observation:

Inputs				**Outputs**		
A_1	**A_0**	**B_1**	**B_0**	**A>B**	**A=B**	**A<B**
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			

4.b) Drive the LED Display using IC-7447

Components required: 7447, 74147, LT5427 (Seven segment display), 1KΩ resistor,IC trainer kit, patch cords.

Decoder: One of the applications of a decoder is too drive a LED display. IC-7447 is an application specific decoder compatible to common cathode LED display.

(i)Design:

Function Table of Seven Segment Display

Inputs to IC 7447				**Outputs of IC 7447**							**Output of LT5427**
D	**C**	**B**	**A**	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$	**Decimal Number**
0	0	0	0	0	0	0	0	0	0	1	
0	0	0	1	1	0	0	1	1	1	1	
0	0	1	0	0	0	1	0	0	1	0	
0	0	1	1	0	0	0	0	1	1	0	
0	1	0	0	1	0	0	1	1	0	0	
0	1	0	1	0	1	0	0	1	0	0	
0	1	1	0	1	1	0	0	0	0	0	
0	1	1	1	0	0	0	1	1	1	1	
1	0	0	0	0	0	0	0	0	0	0	
1	0	0	1	0	0	0	0	1	0	0	

ii) Implementation:

Circuit diagram of BCD to Seven segment decoder driver using IC 7447&LT5427

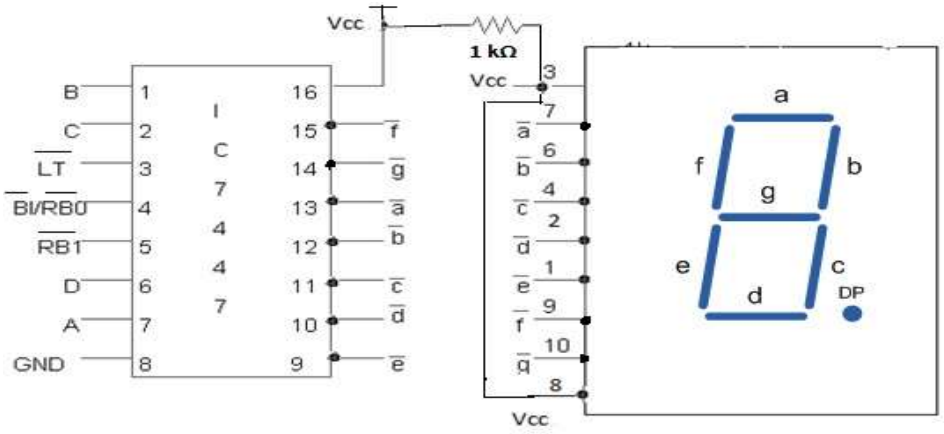


Fig. 4.3 Driving LED display using IC-7447

Procedure:

1. Connections are made as shown in the circuit diagram of Fig. 4.3.
2. The inputs are applied to the decoder and display is verified.

PART 2

4.c)Practice Question: Design an n-bit comparator using IC-7485 (make use of cascading facility)

Inference/Conclusion:

Staff Signature

Cycle II - Experiment -05
Design a Master-Slave JK-FF, D-FF and T-FF

Aim: To design a Pulse triggered Master JK-FF using NAND gates and D-FF and T-FF using same. Also to design Master-Slave JK- FF using circuit simulation software.

Components Required: IC Trainer kit, patch cords, IC-7400, IC-7410, IC-7420.

Flip-flops: Flip flops are the memory element used to store single bit data. The operation of these devices is controlled by clock. These are the sequential devices used to design registers and counters for various applications.

PART 1

5.a) Design a Master JK-FF using NAND gates. Also design D-FF and T-FF using same. Observe the waveform using CRO.

(i) Design:

Truth Table: Pulse triggered JK flip-flop

INPUT					OUTPUT		FUNCTION
Pr'	**Cr'**	**J**	**K**	**Clk**	**Q**	**Q'**	
0	0	X	X	X	1	1	INVALID condition; Pr' and Cr'
0	1	X	X	X	1	0	PRESET or SET condition
1	0	X	X	X	0	1	PRESET or CLEAR condition
1	1	X	X	0	0	1	NO CHANGE condition
1	1	0	0	1	0	1	NO CHANGE condition(HOLD).
1	1	1	0	1	1	0	SET condition.
1	1	0	1	1	0	1	RESET condition.
1	1	1	1	1	Qm'	Qm	TOGGLE condition.

(ii) Implementation:

Circuit Diagram: Pulse triggered JK-FF(Master)

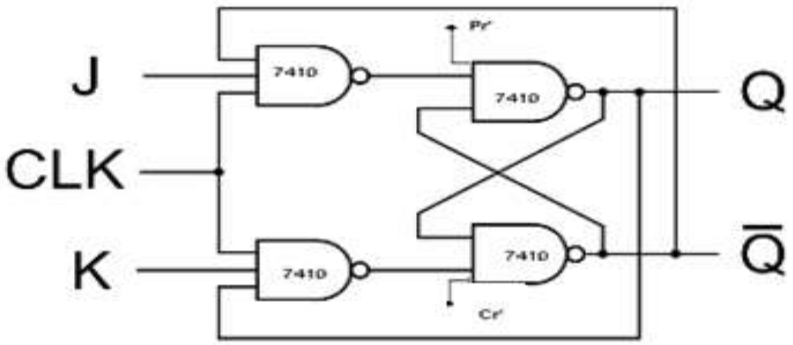
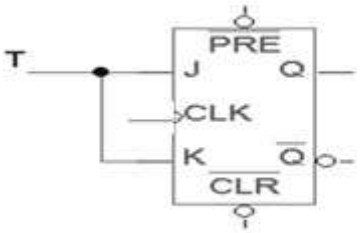


Fig.5.1: Implementation of pulse triggered JK FF using NAND gates

Procedure

1. Circuit connections are made as shown in Fig.5.1.
2. For different i/p's (J, K, Pr' and Cr'), note down the outputs Q and Q' and verify the truth table.
3. To demonstrate the working on CRO, connect CLK=1kHz(say) to clock input, display Q on CRO and observe that Q changes at the positive level of the CLK.

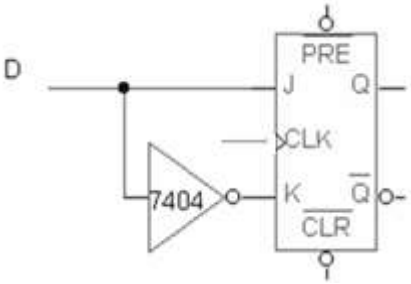
Circuit Diagram: T-FF from Pulse triggered JK-FF:



Truth Table: T-FF:

Pr'	Cr'	T	Clk	Q	Q'	Remarks
0	0	X	X	1	1	INVALID condition
0	1	X	X	1	0	PRESET condition
1	0	X	X	0	1	CLEAR condition
1	1	0	X	0	1	NO CHANGE condition
1	1	X	0	0	1	NO CHANGE condition
1	1	1	1	Q'	Q	TOGGLE condition

Circuit Diagram: D-FF from Pulse triggered JK-FF:



Truth Table : D-FF

Pr'	**Cr'**	**D**	**Clk**	**Q**	**Q'**	Remarks
0	0	X	X	1	1	INVALID condition
0	1	X	X	1	0	PRESET condition
1	0	X	X	0	1	CLEAR condition
1	1	X	X	0	1	NO CHANGE condition
1	1	0	1	0	1	Q = D

Procedure:

1. Considering the condition J=K=1 in Fig 5.1, it is clear that o/p toggles. Hence J and K are connected to High for T FF.
2. Considering the condition K=J', o/p will be Q=J=D. Hence to realize a D-FF, D i/p is connected to J and K=J' for D FF.
3. The o/p is noted down and the TTs are verified.

PART 2

5.b)Practice Question: Design Master Slave JK-FF using circuit simulation software and observe the waveforms.

Inference/Conclusion:

Staff Signature

Experiment -06
Realization of Asynchronous Counters using Various IC's and Shift Registers

Registers and Counters: Registers are used to store multiple bit data by using serial or parallel loading. Counters are the devices to increment or decrement values stored in the registers in a specific order. Basically, they are used to count the clock pulses.

PART 1

6.a) Realization of asynchronous mod-n counter using IC-7490, IC-7493

a)Aim: To Realize of asynchronous mod-n counter using IC- 74112, 7490, IC-7493.
Components Required: ICs -74112, 7490,7493, Digital trainer kit, patch cords.

1) Asynchronous mod-n counter using IC-74112

(i) Design:

Truth Table: 3 bit Asynchronous Counter

Up-Counter

Clk	Q2 MSB	Q1	Qo LSB
-	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Down-Counter

Clk	Q2 MSB	Q1	Qo LSB
--	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0
8	1	1	1

(ii) Implemention:

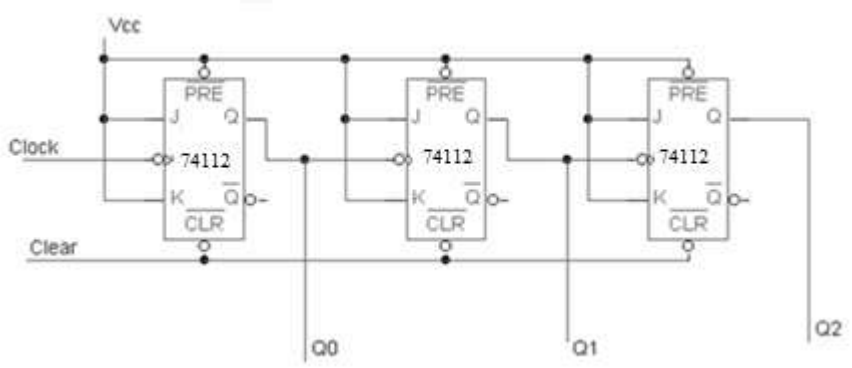


Fig: 6.1 Circuit Diagram: 3-bit up counter (Asynchronous) using IC-74112

Note: For down counter, connect Q' to CLK of succeeding flip flop.

Mod-6 Conter:

Truth Table:

Clk	Q2	Q1	Qo
--	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	0	0	0

Circuit Diagram: Mod-6 counter (Asynchronous):

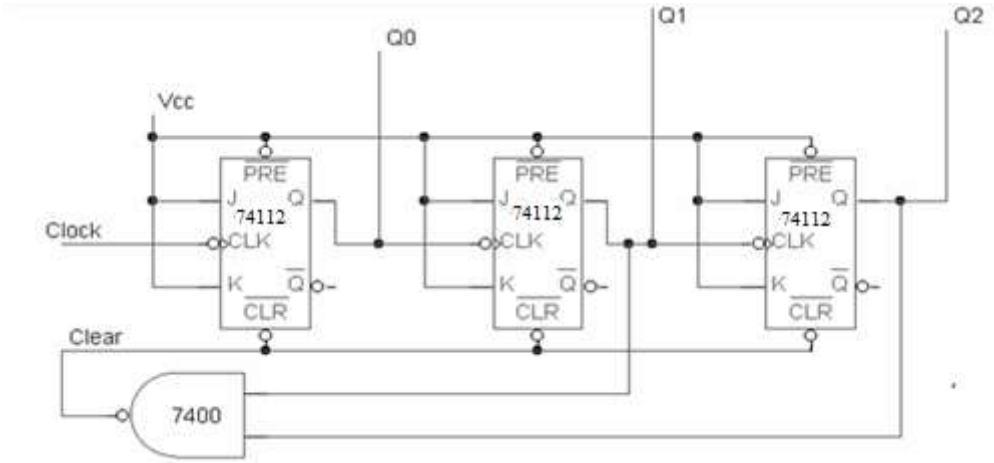


Fig.6.2 Mod -6 asynchronous Counter using IC-74112

Procedure:

1. Make Circuit connections as shown in Fig.6.2
2. Reset all flip-flops by making CLR=0 initially.
3. Apply manual clock pulses and observe outputs as in the truth table.
4. To display the waveforms, select suitable frequency of continuous pulses and connect outputs to oscilloscope.

(2) Mod-10 asynchronous Counter using IC-7490

Truth Table: Mod-10 counter using 7490

Clk	Q3	Q2	Q1	Qo
--	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

Circuit Diagram:mod-10

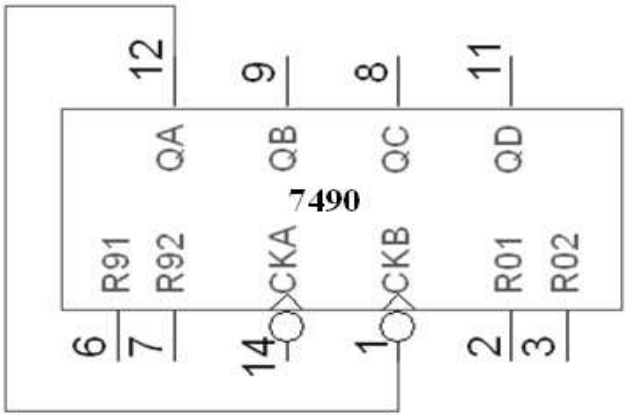


Fig.6.3 Mod -10 asynchronous Counter using IC-7490

Procedure

1. Make Circuit connections as shown in Fig.6.3
2. Apply clock input to pin 14. Connect 6, 7, 2 and 3 to ground.
3. Verify the count sequence by connecting output LED's and selecting clock of lower frequency.
4. To display the waveforms, select suitable frequency for clock pulses and connect o/p's (QA (LSB), QB, QC and QD (MSB) to oscilloscope

Circuit Diagram: Mod-6 counter using 7490

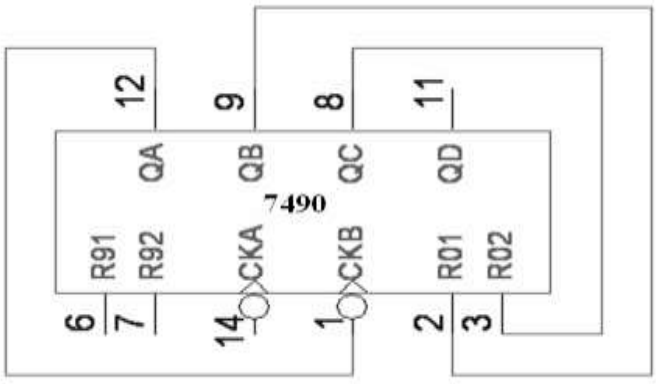


Fig.6.4 Mod -6 asynchronous Counter using IC-7490

Procedure:

1. Make Circuit connections as shown in Fig 6.4.
2. Apply clock input to pin 14.
3. Verify the count sequence by connecting output LED's and selecting clock of lower frequency.
4. To display the waveforms, select suitable frequency for clock pulses and connect o/p's to oscilloscope.

(3) Mod-10 asynchronous counter using IC-7493

Circuit Diagram: Mod-10 Counter using IC-7493

Pin Details of IC-7493

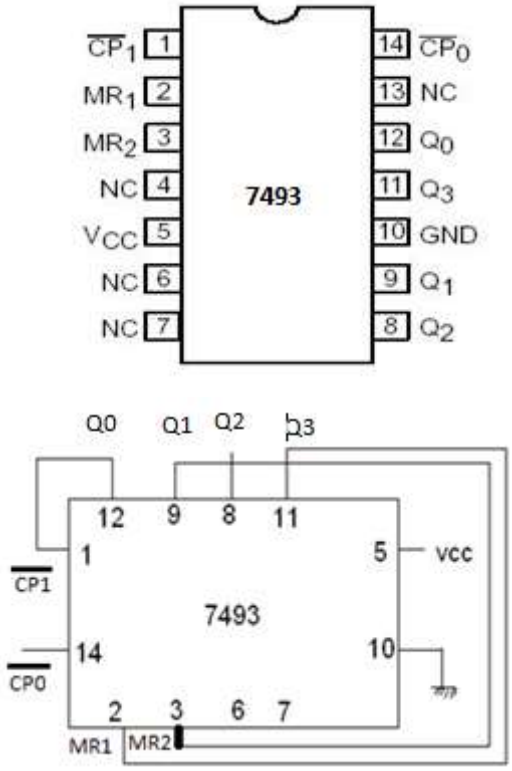


Fig.6.5 Mod -10 asynchronous Counter using IC-7493

Procedure

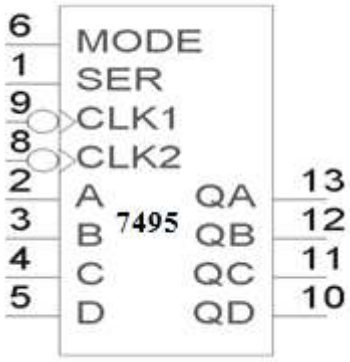
1. Make Circuit connections as shown in Fig.6.5.
2. Apply clock input to pin 14.

6b) Using IC-7495 perform SISO, SIPO, PISO, PIPO, Shift left operations.

Apparatus: IC trainer kit, patch cords, IC-7495.

(i) Design

Pin Diagram: Shift register IC 7495



Note: QD is LSB and QA is MSB.

Truth Table: Serial Input Serial Output (SISO)

SER=1101

Mode = 0 (Serial Mode)					
Clk	SER	QA	QB	QC	QD
--	1	X	X	X	X
1	0	1	X	X	X
2	1	0	1	X	X
3	1	1	0	1	X
4	--	1	1	0	1
5	--	--	1	1	0
6	--	--	--	1	1
7	--	--	--	--	1
8	--	--	--	--	--

SER=X

Mode = 0					
Clk	SER	QA	QB	QC	QD
--	X	1	1	0	1
1	X	X	1	1	1
2	X	X	X	1	0
3	X	X	X	X	1
4	X	X	X	X	X

Procedure:

1.Make Circuit connections as in the pin details.
2. Connect Mode=0 for serial operation.
3.Apply LSB of the data at SER and apply 1st clock pulse to CLK1 (pin9) and observe the data at QA [At the same time the previous data also shifts itself to next FF)
4. Apply next bit of the data to SER and apply 2nd clock pulse. QA shifts to QB and new data enter into QA.
5.Repeat the above procedure till MSB of data enters QA, which is serial input operation.
6.For serial output operation, apply series of clock pulses to Clock and observe output QD only.

Serial In Parallel Out (SIPO)

1.Follow steps 1 to 5 of SISO for SI operation.
2.Data is available in parallel on QD, QC, QB, QA after 4 clock cycles.

Parallel In Serial Out (PISO)

1. Connect Mode=1 for parallel input operation.
2. Enter the parallel data to parallel input lines A,B,C and D.
3. Apply clock pulse to clock pin 8. A falling edge on CLK2 (pin 8) enters the data.
4. To read data serially, change mode select S=0 and apply 4 clock pulses to CLK1 (Pin 9) and observe output of QD.

Parallel In Parallel Out (PIPO)

1.Follow procedure 1 through 3 of PISO for parallel i/p.
2.Data is available on QD, QC, QB, QA parallely.

Clk	Qo	Q1	Q2	Q3
↓	1	1	0	1

Circuit Diagram: Shift Left operation:

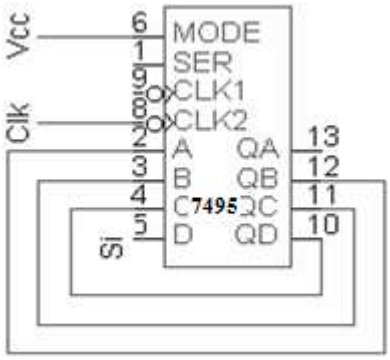


Fig.6.6 Shift left operation using 7495

Truth Table for Shift Left operation:

Clk	QA	QB	Q_C	Q_D	D(Si)
--	X	X	X	X	1
1	X	X	X	1	0
2	X	X	1	0	1
3	X	1	0	1	1
4	1	0	1	1	

Procedure

1. Make Circuit connections as shown in Fig 6.6.
2. With Mode select=1, SER =X, Apply input to 'D' (Si).
3. Apply clock pulse and verify the output at QA.

Shift-right operation

Note: Follow the procedure of SISO operation

Design ring & Johnson counter using IC-7495.

Circuit Diagram: 4-bit Ring Counter

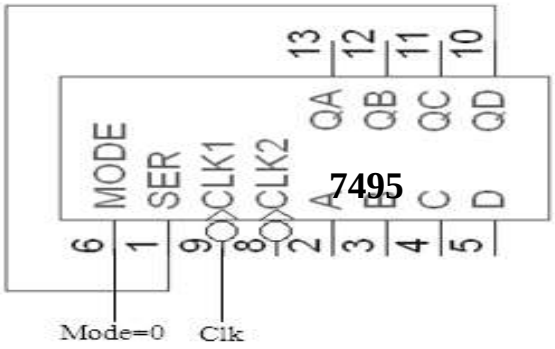


Fig.6.7 4-bit Ring counter using 7495

Truth Table: Count sequence of Ring counter

Clk	QA	QB	QC	QD
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

Procedure:

1. Enter data parallely 1000(A=1, B=0, C=0, D=0) into shift register by making Mode=1 and applying a clock to CLK2. Refer Fig.6.7 for the connections.
2. Connect QD to SER input and apply a series of clock pulse to Clk1 with Mode=0.
3. Data '1' moves in a ring fashion with clock pulses applied continuously.
4. For Johnson counter realization, connect inverter in between QD and SER of ring counter and follow the same procedure.

Truth Table: Count sequence of 4-bit Johnson counter

Clk	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

Inference/Conclusion

Staff Signature

Experiment -07
Design of Synchronous UP/DOWN counter

Aim: Design a synchronous up/down counter using IC-74112,74192/74193.
Apparatus: IC trainer kit, IC-74112 , 7408,74192/74193,7410, Patch cords.
Counters: Counter is a sequential circuit. A digital circuit which is used for counting pulses is known as counter. Counter is the widest application of flip-flops.

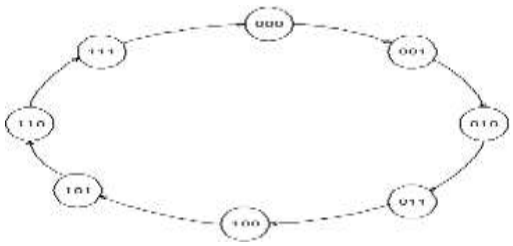
PART 1

7.a) Design of synchronous up counter using IC-74112.

(i)Design

MOD-8 counter:

Step1: Creating state transition diagram.



Step 2: Creating present state-next state table

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Step 3: Expand the present state-next state table to form the transition table.

No. of flip-flops (n)=>2^n >=N (Number states)

Present State			**Next State**			**Present inputs**		
Q_2	**Q_1**	**Q_0**	**Q_2**	**Q_1**	**Q_0**	**J_2K_2**	**J_1K_1**	**J_0K_0**
0	0	0	0	0	1	0X	0X	1X
0	0	1	0	1	0	0X	1X	X1
0	1	0	0	1	1	0X	X0	1X
0	1	1	1	0	0	1X	X1	X1
1	0	0	1	0	1	X0	0X	1X
1	0	1	1	1	0	X0	1X	X1
1	1	0	1	1	1	X0	X0	1X
1	1	1	0	0	0	X1	X1	X1

Step 4: Use Karnaugh maps to identify the present state logic functions for each of the inputs.

E.g. for J_2 we get:

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	1	0
1	X	X	X	X

$J_2 = Q_1Q_0$

Using similar techniques for the other inputs we get:

$K_2 = Q_1Q_0$

$J_1 = Q_0$, $K_1 = Q_0$

$J_0 = 1$, $K_0 = 1$

Note: Q2, Q1 and Q0 mentioned in the K-Map are present inputs.

(ii)Implementation

Step 5: Constructing Circuit as shown in Fig. 7.1.

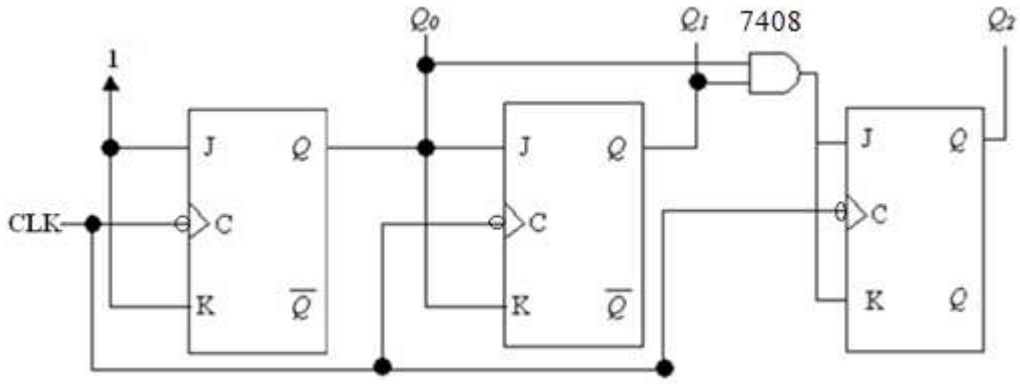


Fig 7.1 Design of 3-bit up counter

Note: Connect the Clear and Preset pins to active HIGH.

Procedure

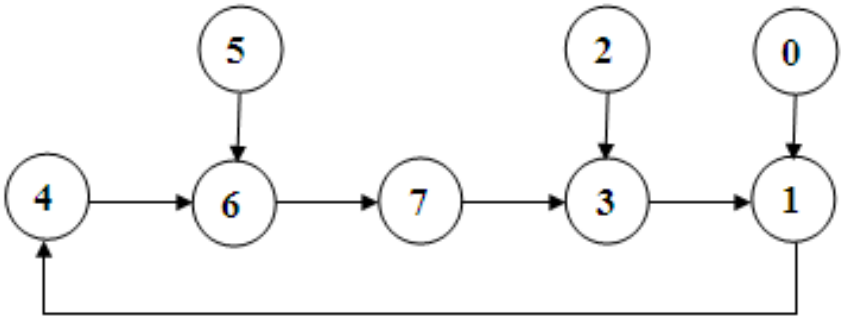
1. Insert the IC into trainer kit.
2. Connect V_{CC} and GND to suitable pins of the IC.
3. Verify the count sequence by connecting output LED's and selecting clock of lower frequency.
4. To display the waveforms, select suitable frequency for clock pulses and connect o/p's to oscilloscope.

7.b) Design of a synchronous counter for the following count sequence: 4-6-7-3-1----4-6……. (Avoid Lock out condition)

(i) Design

Step1: Creating state transition diagram.

To avoid Lock out condition, next states of unused states are sent to used state as shown in state diagram.



Here, unused states 5, 2 and 0 are forced to go into 6, 3 and 1 state respectively to avoid lockout condition.

Step 2: Creating present state, next state and transition table.

Present state			**Next state**			**Flip-Flop inputs**		
Q_2	**Q_1**	**Q_0**	**Q_2**	**Q_1**	**Q_0**	**J_2K_2**	**J_1K_1**	**J_0K_0**
0	0	0	0	0	1	0X	0X	1X
0	0	1	1	0	0	1X	0X	X1
0	1	0	0	1	1	0X	X0	1X
0	1	1	0	0	1	0X	X1	X0
1	0	0	1	1	0	X0	1X	0X
1	0	1	1	1	0	X0	1X	X1
1	1	0	1	1	1	X0	X0	1X
1	1	1	0	1	1	X1	X0	X0

Step 3:Use Karnaugh maps to identify the present state logic functions for each of the inputs.

For J_2,

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	0	0
1	X	X	X	X

$J_2=Q_1'Q_0$ K2=Q1Q0 J1=Q2 K1=Q2'Q0 J0=Q2'+Q1 K0=Q1'

(ii) Implementation

Step 4: Constructing Circuit as shown in Fig .7.2

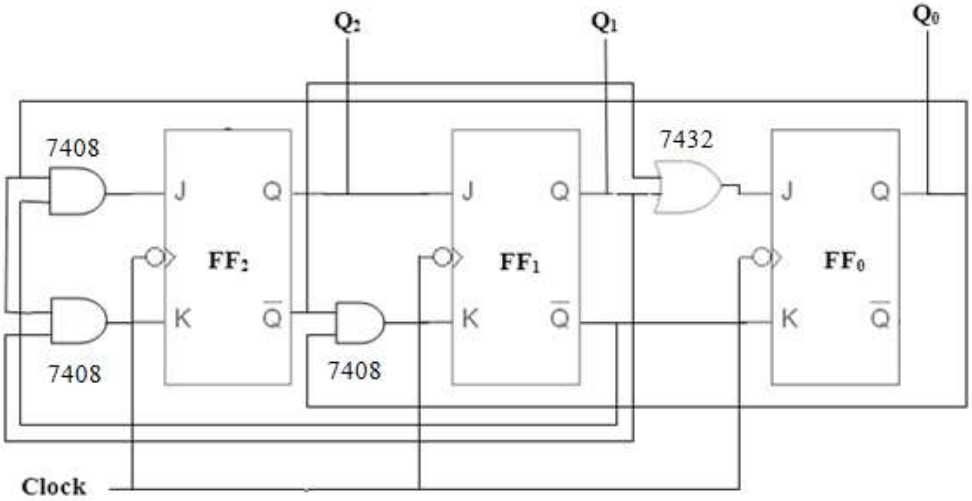


Fig. 7.2 Design of a counter to count specific sequence

Note: Connect the Clear and Preset pins to active HIGH.

Inference and Observations:

Staff Signature

Experiment -08

Design presentable counters IC-74192/193 perform mod-n counts.

Aim: Design presentable counters IC-74192/193 perform mod-n counts.

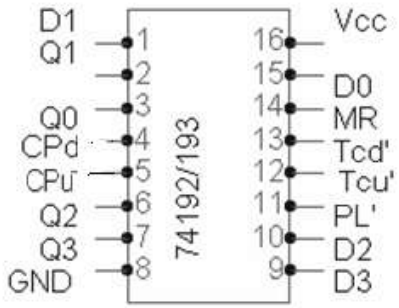
Apparatus: IC trainer kit, IC-74112 , 7408,74192/74193,7410, Patch cords.

Counters: Counter is a sequential circuit. A digital circuit which is used for counting pulses is known as counter. Counter is the widest application of flip-flops.

PART 1

8.a) Design presentable counters IC-74192/193 perform mod-n counts.

Pin diagram of Presettable up-down counter IC 74192/3



Note:
IC-74192 is a 4-bit synchronous programmable rising edge sensitive up/down decade counter.
IC-74193 is a 4-bit synchronous programmable rising edge sensitive up/down binary counter.
Function Table of 74193

Function Table:

Operating Mode	**MR**	**$\overline{PL}$**	**CP_u**	**CP_D**	**Do**	**D_1**	**D_2**	**D_3**	**Qo**	**Q_1**	**Q_2**	**Q_3**	**$\overline{TCu}$**	**$\overline{TCd}$**
RESET	H	X	X	L	X	X	X	X	L	L	L	L	H	L
	H	X	X	H	X	X	X	X	L	L	L	L	H	H
PARALLEL LOAD	L	L	X	L	L	L	L	L	L	L	L	L	H	L
	L	L	X	H	L	L	L	L	L	L	L	L	H	H
	L	L	L	X	H	X	X	H	$Q_N = D_N$				L	H
	L	L	H	X	H	X	X	H	$Q_N = D_N$				H	H
COUNT UP	L	H	↑	H	X	X	X	X	COUNT UP				H*	H
COUNT DOWN	L	H	H	↑	X	X	X	X	COUNT DOWN				H	H*

Note: 1). TCu becomes low when count reaches 1001.
2). TCd becomes low when count reaches 0000.

(1) Realization of Presettable up counters using 74192/74193

i) BCD Counter(0-9)

Make the connections as follows:

CPu<= Clock (Rising Edge triggered)

CPd<= 1

MR <= 0 (Master Reset disabled)

PL' <= 1 (Parallel Load disabled)

The outputs are Q3, Q2, Q1 and Q0. The TCu' is normally HIGH. When the count reaches maximum state (9 for 74192 and 15 for 74193), the next HIGH-to-LOW transition of CPu will cause TCu' to go LOW and stays LOW until CPu goes HIGH again.

ii) Up Counter (3-9)

CPu<= Clock (Rising Edge triggered)

CPd<= 1

MR <= 0 (Master Reset disabled)

PL' <= TCu'

D3D2D1D0 <=0011

The outputs are Q3, Q2, Q1 and Q0. The parallel data 0011 is loaded, when TCu' goes LOW, regardless of the state of Clock.

iii) Up Counter(3-6)

CPu<= Clock (Rising Edge triggered)

CPd<= 1

MR <= 0 (Master Reset disabled)

D3D2D1D0 <= 0011

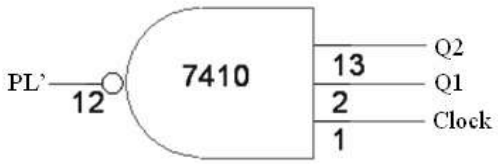


Fig. 7.3 Up counter(3-6)

The outputs are Q3, Q2, Q1 and Q0. When the count reaches 6 (Q2=1, Q1=1), PL' goes LOW on the next rising edge of the clock and parallel data(0011) will be loaded. Refer Fig.7.3

(2) Realization of Presettable Down Counters 74192/74193

(i)BCD Down Counter (9-0)

Make the connections as follows.

CPu<= 1

CPd<= Clock (Rising Edge triggered)

MR <= 0 (Master Reset disabled)

PL' <= 1 (Parallel Load disabled)

The outputs are Q3, Q2, Q1 and Q0. The TCd' is normally HIGH. When the count reaches 0 state, the next HIGH-to-LOW transition of CPd will cause TCd' to go LOW and stays LOW until CPd goes HIGH again.

(ii)Down Counter (6 to 0)

CPu<= 1
CPd<= Clock (Rising Edge triggered)
MR <= 0 (Master Reset disabled)
PL' <= TCd'
D3D2D1D0 <=0110

The outputs are Q3, Q2, Q1 and Q0. The parallel data 0110 is loaded, when TCd' goes LOW, regardless of the state of Clock.

(iii) Down Counter (6-3)

CPu<= 1
CPd<= Clock (Rising Edge triggered)
MR <= 0 (Master Reset disabled)
D3D2D1D0 <=0110

Fig. 7.4 Down counter(3-6)

The outputs are Q3, Q2, Q1 and Q0. When the count reaches 3 (Q1=1,Q0=1), PL' goes LOW on the next rising edge of the clock and parallel data(0110) will be loaded. Refer Fig. 7.4

PART 2

8.b) Practice Question: Design a synchronous 4-bit up/down counter using circuit simulation software and observe the waveforms.

Conclusion/Inference:

Conclusion/Inference:

Staff Signature

Innovative Lab Experiments
Experiment -09

Design a sequence generator using a shift register to obtain a sequence Y= 100010011010111

Conclusion/Inferences:

Staff Signature

Experiment -10

Using IC-74192/193, drive the LED display and generate given sequence of counts.

Conclusion/Inferences:

Staff Signature

Experiment-11

Design a 2-bit ALU operation using circuit simulation software and observe the waveforms

Conclusion/Inferences:

Staff Signature

Experiment-12

Virtual Lab Experiments

I. Digital Electronics IITG

https://de-iitg.vlabs.ac.in/List%20of%20experiments.html

1. To implement Half adder & Full adder by using basic and universal gates
2. To study Parallel Binary Adder
3. Implementation of JK Flip-Flop
4. Implementation of simple two-bit ripple counter
5. synchronous up down counter

II. Digital Electronics IITR

https://de-iitr.vlabs.ac.in/List%20of%20experiments.html

1. Implementation and verification of decoder/de-multiplexer and encoder using logic gates.
2. Implementation of 4x1 multiplexer and 1x4 demultiplexer using logic gates.

III. Digital Logic Design IITB

https://dld-iitb.vlabs.ac.in/List%20of%20experiments.html

1. Design of Gray to Binary code converter using MSI ICs

2. Design of 8-bit digital Comparator using MSI ICs

IV. Digital Electronic Circuits Laboratory IITKGP

http://vlabs.iitkgp.ac.in/dec/#

1. Analysis of Functions of BCD-TO-7-segment Decoder / Driver and Operation of 7-segment LED Display
2. Analysis and Synthesis of Sequential Circuits using Basic Flip-Flops
3. Analysis and Synthesis of Multi-bit Sequential Circuits using Shift Registers
4. Design of Arithmetic Logic Unit

Instructions to Virtual Lab Experiments:

1. The list of experiments is mentioned clearly under each center and their URL is also provided.
2. Go to the respective URL, follow the instructions given, conduct the experiment, attend the quiz, Write the inference of that experiment.
3. Once conduction is over, take the screen shot, if any waveforms are generated, take the screenshot/print of the same.
4. Along with regular lab experiments, you need to follow the same procedure for this Virtual labs, while writing the datasheets. Virtual Lab experiments also should contain, Aim, components required, Brief procedure, Design aspects, Circuit Diagram, Tabular column and Conclusions or inference.

Conclusion/Inferences:

Staff Signature

Appendix:

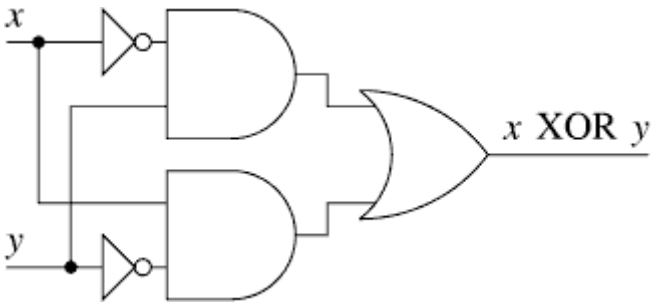
Circuit Simulation Software: Logisim

Logisim allows you to design and simulate digital circuits. It is intended as an educational tool, to help you learn how circuits work.

To practice using Logisim, let's build a XOR circuit - that is, a circuit that takes two inputs (which we'll call *x* and *y*) and outputs 0 if the inputs are the same and 1 if they are different. The following truth table illustrates.

x	*y*	*x* XOR *y*
0	0	0
0	1	1
1	0	1
1	1	0

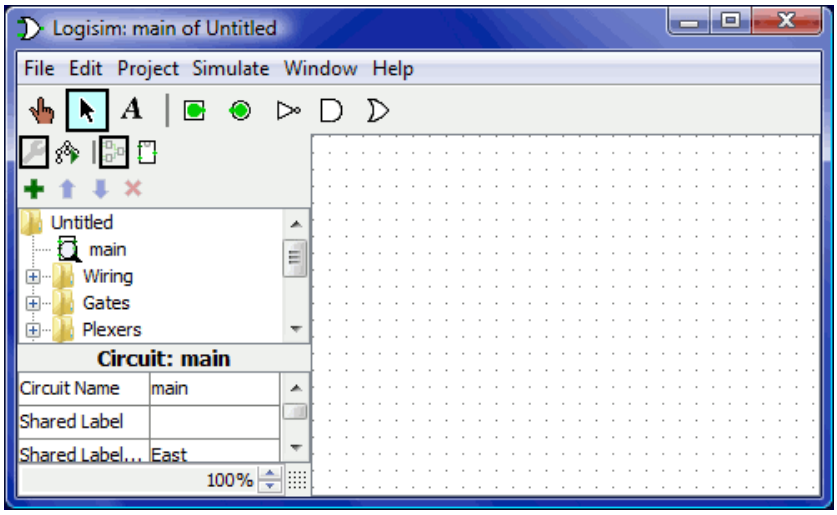
We might design such a circuit on paper.



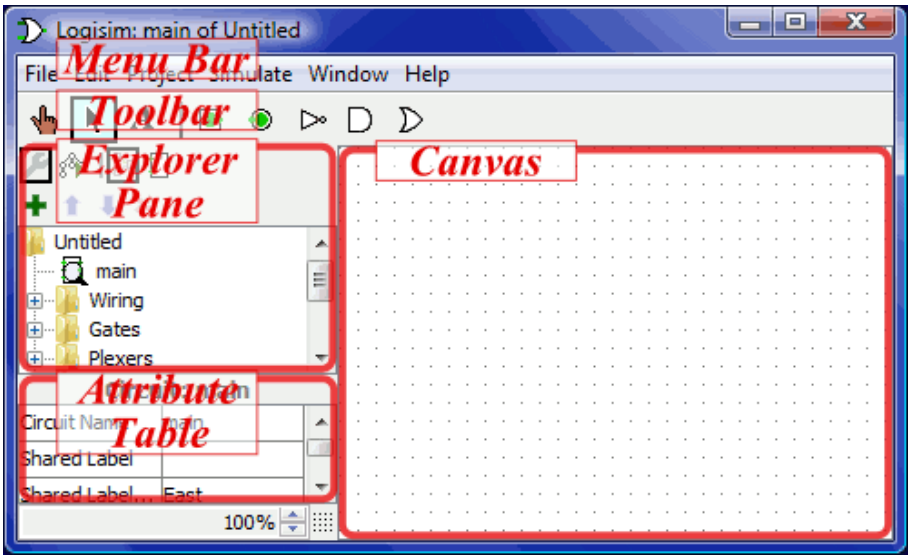
But just because it's on paper doesn't mean it's right. To verify our work, we'll draw it in Logisim and test it.

Orienting yourself

When you start Logisim, you'll see a window similar to the following.



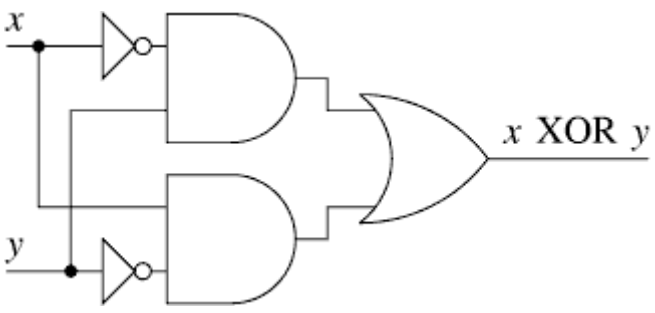
All Logisim is divided into three parts, called the *explorer pane*, the *attribute table*, and the *canvas*. Above these parts are the *menu bar* and the *toolbar*.



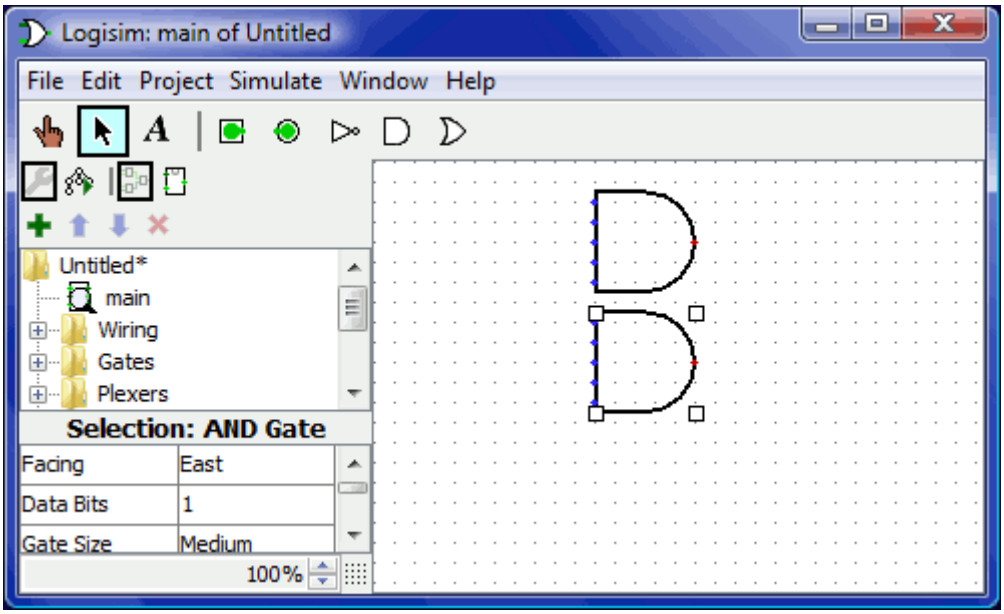
The canvas is where you'll draw your circuit; and the toolbar contains the tools that you'll use to accomplish this. Also, the menu bar is self-explanatory.

Step 1: Adding gates

Recall that we're trying to build the following circuit in Logisim.

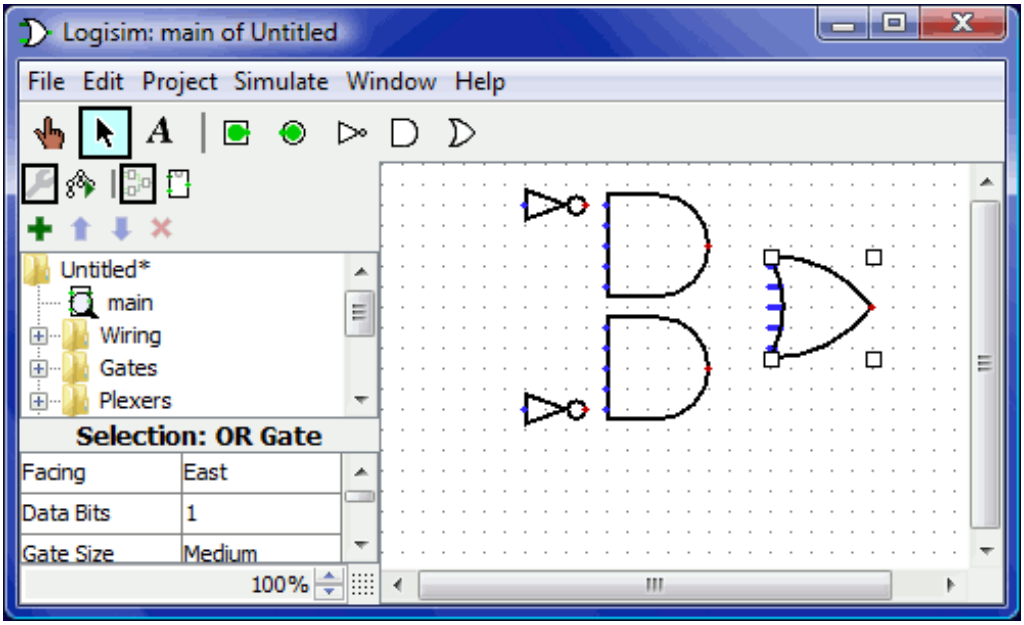


Click on the AND tool in the toolbar (, the next-to-last tool listed). Then click in the editing area where you want the first AND gate to go. Be sure to leave plenty of room for stuff on the left. Then click the AND tool again and place the second AND gate below it.



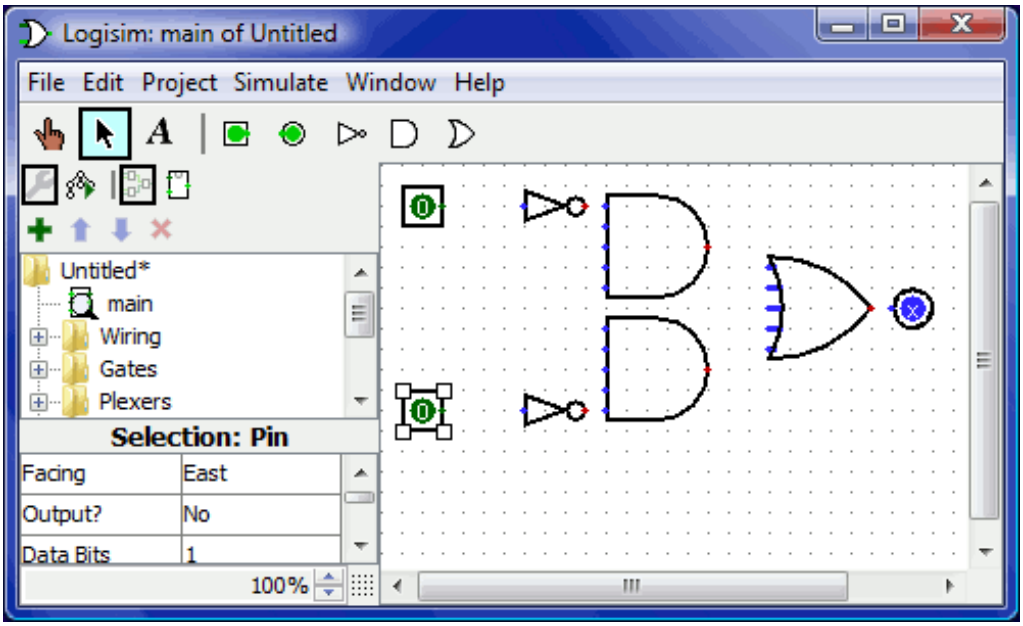
Notice the five dots on the left side of the AND gate. These are spots where wires can be attached. It happens that we'll just use two of them for our XOR circuit; but for other circuits, you may find that having more than two wires going to an AND gate is useful.

Now add the other gates. First click on the OR tool ( ); then click where you want it. And place the two NOT gates into the canvas using the NOT tool ( ).



Now we want to add the two inputs *x* and *y* into the diagram. Select the Input tool ( ), and place the pins down. You should also place an output pin next to the OR gate's output using

the Output tool ( ). (Again, I'm leaving a bit of space between the OR gate and the output pin, but you might choose to place them right next to each other.)



If you decide you don't like where you placed something, then you can select it using the Edit tool ( ) and drag it to the desired spot. Or you can delete it altogether by selecting Delete from the Edit menu or pressing the Delete key.

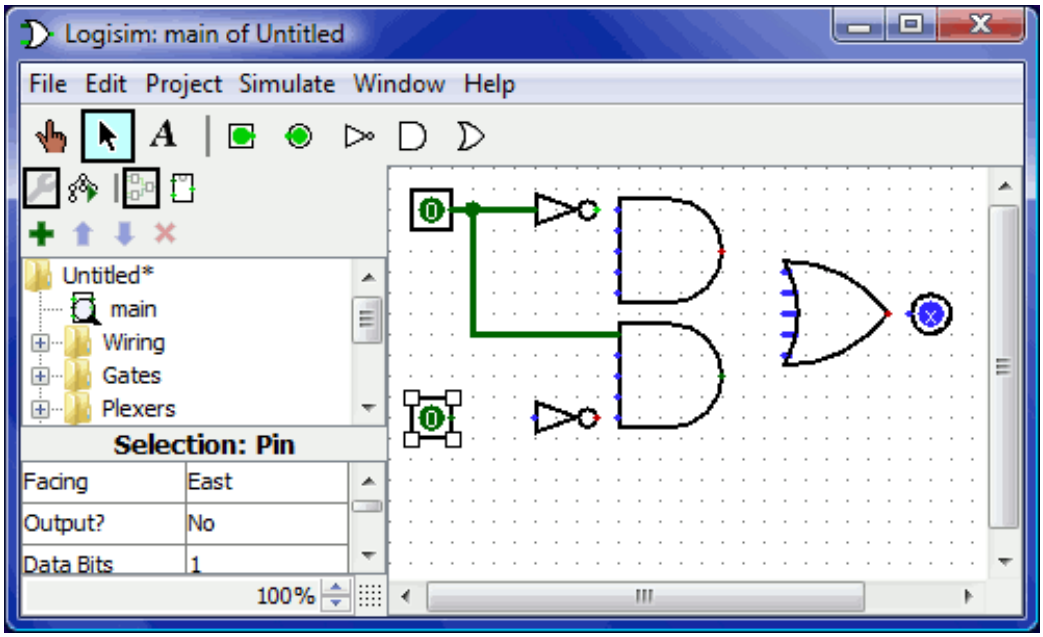
As you place each component of the circuit, you'll notice that as soon as the component is placed, Logisim reverts to the Edit tool so that you can move the recently-placed component or (as we'll see soon) connect the component to others by creating wires. If you want to add a copy of the recently placed component, a shortcut is to press Control-D to duplicate the selection.

Step 2: Adding wires

After you have all the components blocked out on the canvas, you're ready to start adding wires. Select the Edit Tool ( ). When the cursor is over a point that receives a wire, a small green circle will be drawn around it. Press the mouse button there and drag as far as you want the wire to go.

Logisim is rather intelligent when adding wires: Whenever a wire ends at another wire, Logisim automatically connects them. You can also "extend" or "shorten" a wire by dragging one of its endpoints using the edit tool.

Wires in Logisim must be horizontal or vertical. To connect the upper input to the NOT gate and the AND gate, then, I added three different wires.

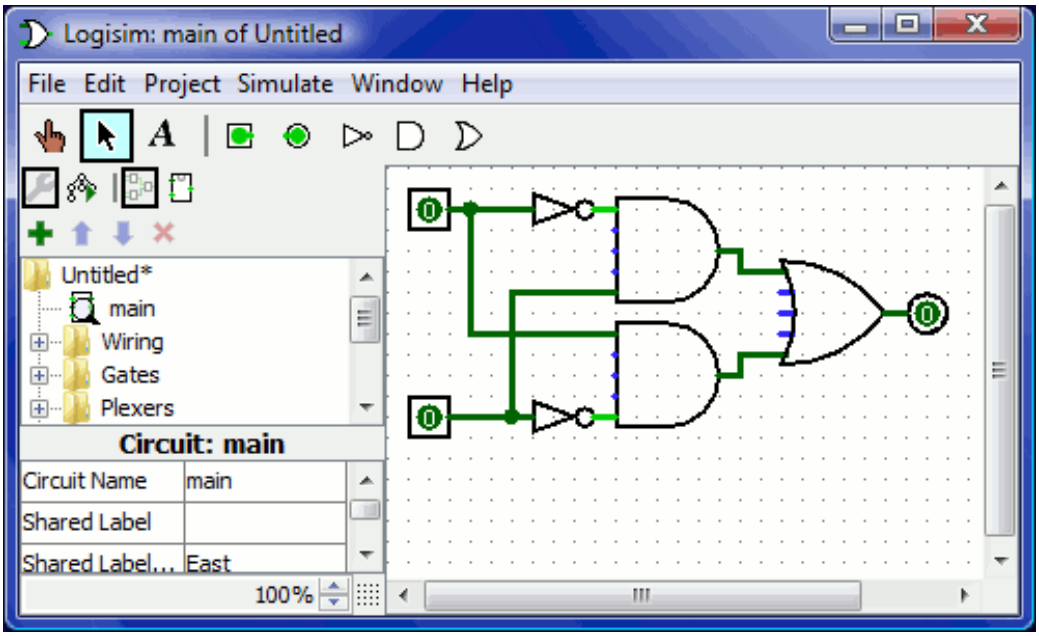


Logisim automatically connects wires to the gates and to each other. This includes automatically drawing the circle at a *T* intersection as above, indicating that the wires are connected.

As you draw wires, you may see some blue or gray wires. Blue in Logisim indicates that the value at that point is "unknown," and gray indicates that the wire is not connected to anything. This is not a big deal as you're in the process of building a circuit. But by the time you finish it, none of your wires should be blue or gray. (The unconnected legs of the OR gate will still be blue: That's fine.)

If you do have a blue or a gray wire after you think everything ought to be connected, then something is going wrong. It's important that you connect wires to the right places. Logisim draws little dots on the components to indicate where wires ought to connect. As you proceed, you'll see the dots turn from blue to light or dark green.

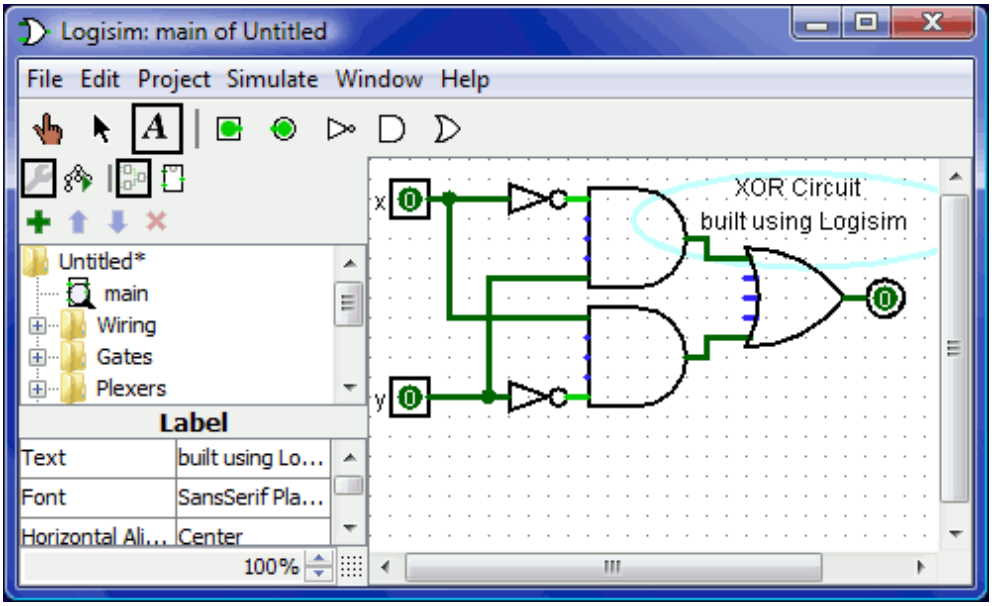
Once you have all the wires connected, all of the wires you inserted will themselves be light or dark green.



Step 3: Adding text

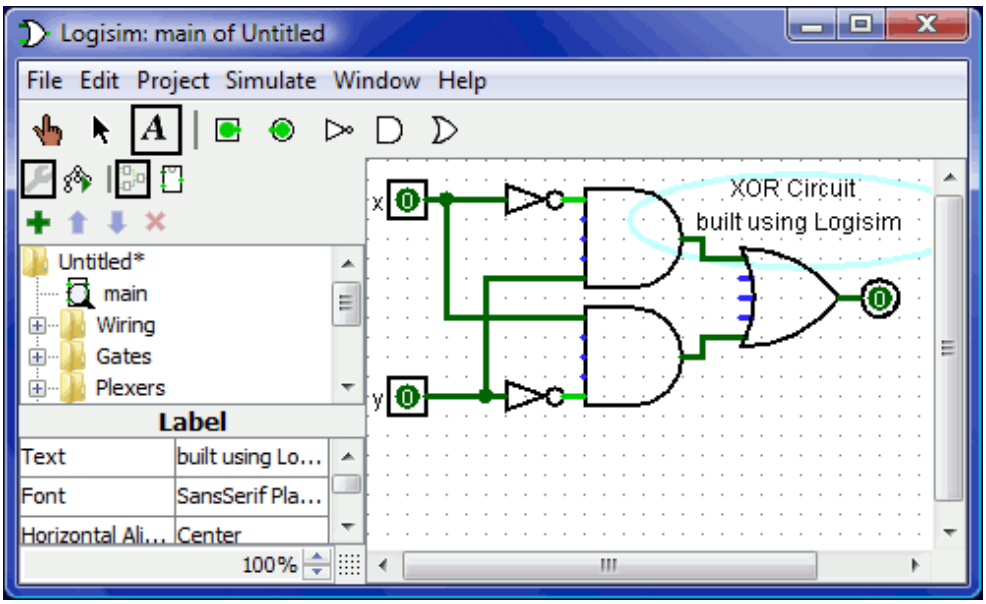
Adding text to the circuit isn't necessary to make it work; but if you want to show your circuit to somebody (like a teacher), then some labels help to communicate the purpose of the different pieces of your circuit.

Select the text tool (**A**). You can click on an input pin and start typing to give it a label. (It's better to click directly on the input pin than to click where you want the text to go, because then the label will move with the pin.) You can do the same for the output pin. Or you could just click any old place and start typing to put a label anywhere else.



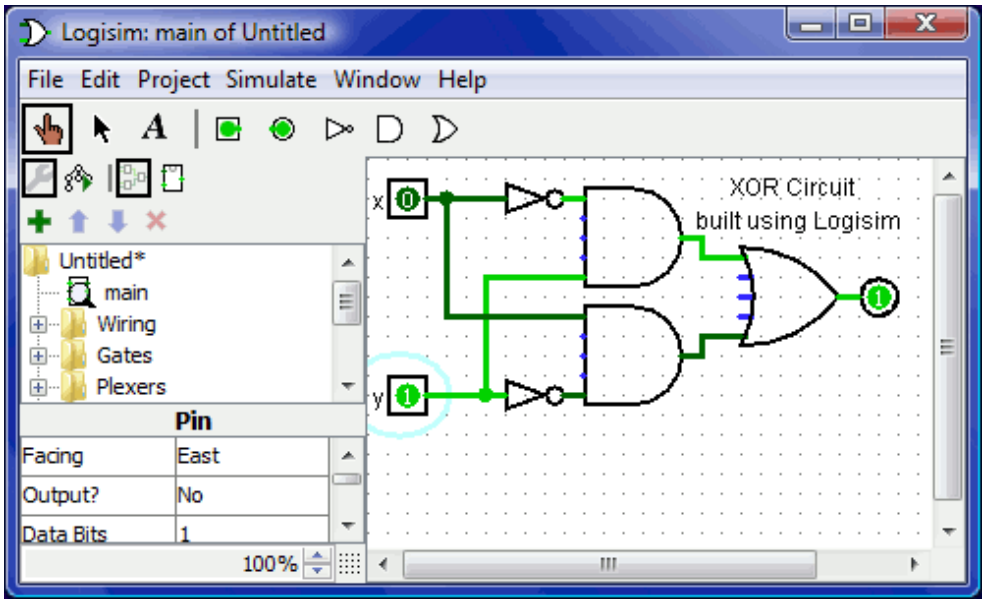
Step 4: Testing your circuit

The final step is to test our circuit to ensure that it really does what we intended. Logisim is already simulating the circuit. Let's look again at where we were.



Note that the input pins both contain 0s; and so does the output pin. This already tells us that the circuit already computes a 0 when both inputs are 0.

Now to try another combination of inputs. Select the poke tool () and start poking the inputs by clicking on them. Each time you poke an input, its value will toggle. For example, we might first poke the bottom input.



When you change the input value, Logisim will show you what values travel down the wires by drawing them light green to indicate a 1 value or dark green (almost black) to indicate a 0 value. You can also see that the output value has changed to 1.

So far, we have tested the first two rows of our truth table, and the outputs (0 and 1) match the desired outputs.

x	*y*	*x* XOR *y*
0	0	0
0	1	1
1	0	1
1	1	0

By poking the switches through different combinations, we can verify the other two rows. If they all match, then we're done: The circuit works!

To archive your completed work, you might want to save or print your circuit. The File menu allows this, and of course it also allows you to exit Logisim.

Now that you are finished with tutorial, you can experiment with Logisim by building your own circuits. Logisim is a powerful program, allowing you to build up and test huge circuits; this step-by-step process just scratches the surface.