LOGIC DESIGN LABORATORY

ITU
Electronics and
Communication
Engineering
Department

2021

Experiment 1 Combinational Circuit Analysis

1. Introduction

A Boolean function with input variables x = x1, x2, …, xn and output variable z, is defined as combinational function when z only depends on present values of x, and is independent from previous values of z. Implementation of a combinational function with circuit elements is called combinational circuit. Graphs of combinational circuits are directed graphs, and they do not include meshes and feedback connections. Therefore, combinational circuits are also called as acyclic circuits.

Analysis of combinational circuits consists of determination of function of implemented circuit. Since digital circuits are a combination of digital components, output function of circuit can be expressed as a Boolean expression using Boolean algebra. In the Boolean algebra, every connection in circuits is expressed as a variable which can take values either 0 or 1. Circuit function can be determined using a truth table which is combination of every possible inputs and resulting outputs. Determination of circuit function can be in the form of either Boolean expressions or truth table. A digital circuit is analyzed during test phase to see if it implements the circuit function correctly or not. After detection of an error, the place causing error is determined and error-correction phase starts.

2. Experiment Assignments

2.1. Step 1

Find the Boolean expressions of each logic gate(G1-G8) given in Figure 1 in terms of input variables(x0-x3). Implement this circuit onto experiment board. Make sure that all ICs have supply and ground connections. Connect inputs to switches (S0-S3), and connect outputs to LEDs (L0-L7).

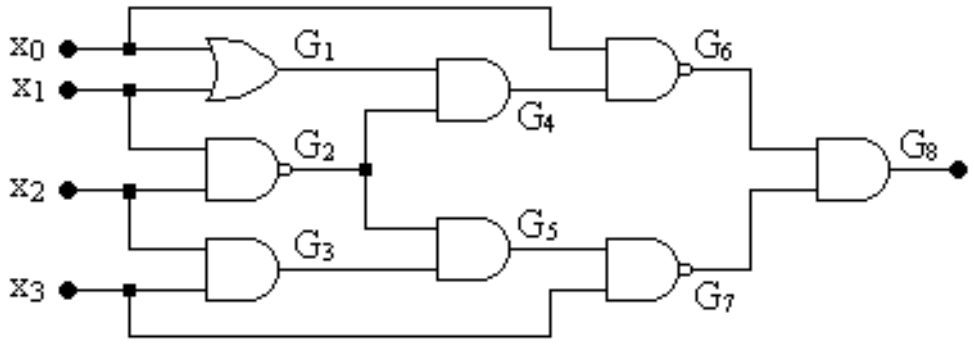


Figure 1: Combinational circuit to be analyzed.

Fill the truth table given in **Table 1**.

Table 1: Truth table of circuit given in Figure 1.

x3	x2	x1	x0	G1	G2	G3	G4	G5	G6	G7	G8
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								
1	1	1	0								
1	1	1	1								

2.2. Step 2

Implement the circuit given in Figure 2. Make sure that all ICs have supply and ground connections. Connect inputs to switches (S0-S3), and connect outputs to LEDs (L0-L2). Fill the truth table given in **Table 1**. Determine the circuit function with the help of truth table.

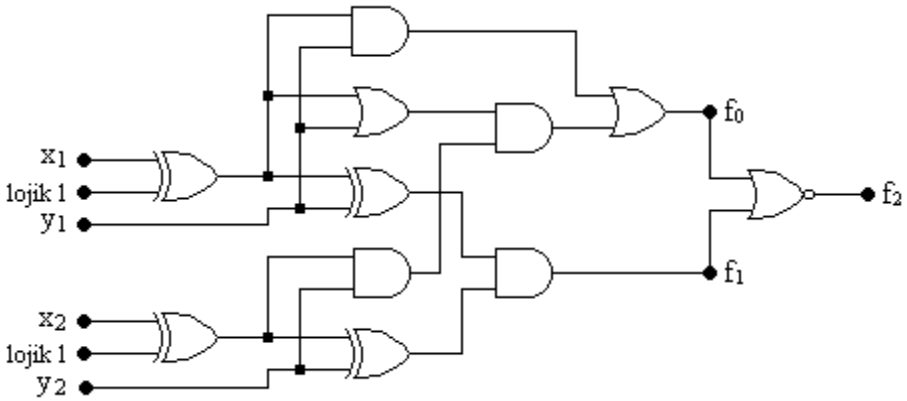


Figure 2: Combinational circuit.

Table 2: Truth table of circuit given in Figure 2.

x1	x2	y1	y2	f0	f1	f2
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

3. Equipment List

Equipment	Quantity	Library
7400 – NAND IC	1	logi7400dip
7402 – NOR IC	1	logi7400dip
7408 – AND IC	1	logi7400dip
7432 – OR IC	1	logi7400dip
7486 – EXOR IC	1	logi7400dip

Experiment 2 Combinational Circuit Synthesis

1. Introduction

In the design of combinational circuit, for the circuit function which is orally defined, truth table corresponding to output z is constructed with inputs x1, x2, …, xn. In the truth table, 2n combination of input variables x1, x2, …, xn contructs a set which is called n-dimensional space. For every point in the n-dimensional space, value of output z, becomes 1, 0 or dont care(x). Not defining and output for an input combination means that either this input combination can not be applied or output can become either 0 or 1. If a circuit produces 1 for input combinations corresponding to 1, and produces 0 for input combinations corresponding to 0, function of output can be implemented. Aim of combinational circuit synthesis is implementing circuit function. Combinational circuit synthesis can be categorized into two methods. First method is finding minimal function with the help of QuineMcCluskey or Karnaugh methods. Number of independent variables determines the method to be used for finding minimal function. Karnaugh method is prefered with functions which have up to 4 or 5 variables. Circuit corresponding to minimal function can be implemented as two-level (sum of products or product of sums) or in a way that which is going to have less circuit complexity than two-level implementation. Complexity of combinational circuits is defined as sum of number of logic gates and number of inputs. However, if the number of inputs of each logic gate is same, complexity can be defined as number of logic gates. Level of combinational circuits is defined as maximum number of logic gates from inputs to outputs. Second method for combinational circuit synthesis producing an algorithm from oral definiton, and implementing corresponding circuit to this algorithm. These method is useful for function with a lot of variables. For example, comparator and encoder circuits can be implemented with this method.

When these two methods compared, for the first one, truth table exponentialy grows when number of variables increase. Fort he latter one, exponential growth problem does not exist, but it is not always possible to produce an algorithm from oral definition.

In theory, circuit design with logic gates is based on minimilization of numbers of logic gates and inputs. Since integrated circuits is used in practice, minimilization is related to number of integrated circuits.

Verbal Description 1 : Design a combinational circuit which converts Binary-coded Decimals(BCDs) to 84-2-1 coded binary. This combinational circuit has four independent inputs, x3x2x1x0, (x3: MSB) and four outputs, 8 4 -2 -1. For example, 5 is represented in BCD as "0101", and its 8 4 -2 -1 coded version is "1011"(1*8 + 0*4 + 1*(-2) + 1*(-1) = 5). Truth table of converter is given in Table 1

Tablo 1 : BCD – (84-2-1) kod dönüştürücü doğruluk tablosu.

x_3	x_2	x_1	x_0	f_8	f_4	f_{-2}	f_{-1}
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	1	1
10-		15		Keyfi			

Boolean functions for each output from truth table:

$$\begin{aligned} \mathbf{f_8} &: x_0x_2 + x_1x_2 + x_1'x_3 \\ \mathbf{f_4} &: x_0x_2` + x_1x_2` + x_0'x_1'x_2 \\ \mathbf{f_{-2}} &: x_0x_1' + x_0'x_1 \\ \mathbf{f_{-1}} &: x_0 \end{aligned} \qquad (1)$$

If these output functions are implemented using AND, OR and NOT gates as two-level in the form of sum of products, 7 AND gates with two inputs, 1 AND gate with three inputs, 2 OR gates with three inputs, 1 OR gate with two inputs and 3 NOT Gates are required. These requirements led to usage of 6 ICs. In Figure 1, considering common components of output functions, same circuit is designed as four-level with 3 ICs.

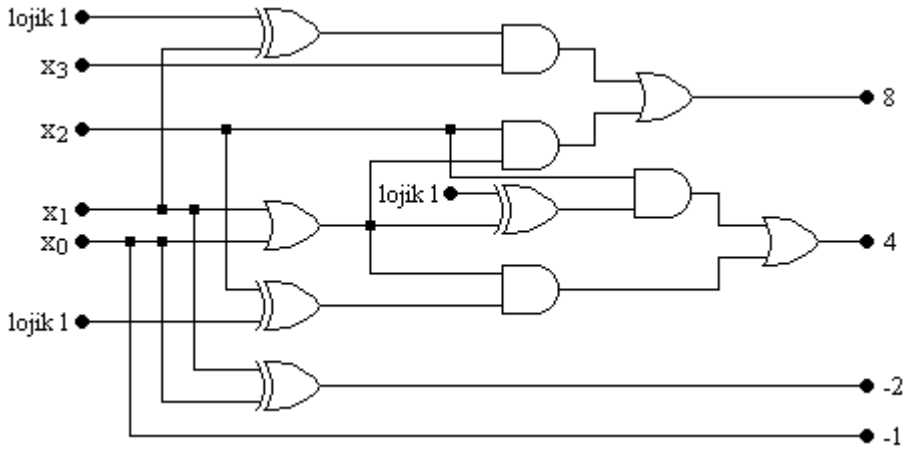


Figure 1: BCD to 84-2-1 converter circuit.

Verbal Description 2: Design an index coder with eight inputs, $x_7x_6x_5x_4x_3x_2x_1x_0$, and three outputs, $Z_2Z_1Z_0$. Index is determined when an input is 1 and others are 0. Output is a binary coded value of index of input which is one. Truth table of index coder is given in **Table 2**.

Table 2: Index coder truth table.

x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	Z_2	Z_1	Z_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1
Other Input Combinations								Don't care		

Instead of finding output functions by using Quine-McCluskey or Karnaugh methods, for each output, determining which inputs must be 1, is an easier method for designing index coder. Index coder circuit is given in Figure 2.

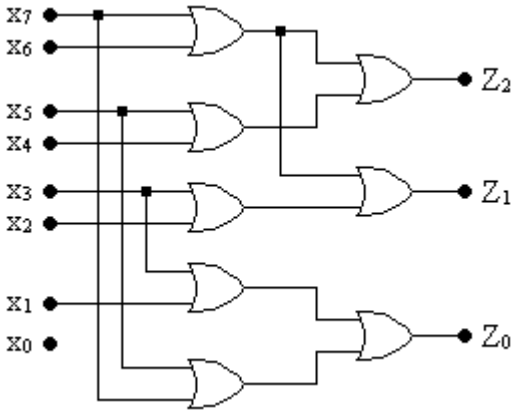


Figure 2: Index coder circuit.

2. Experiment Assignments

2.1. Step 1

Implement this circuit given in Figure 1 onto experiment board. Make sure that all ICs have supply and ground connections. Connect inputs to switches, and connect outputs to LEDs. Apply input combinations to your circuit and check that whether your circuit gives same outputs in Table 1.

2.2. Step 2

Implement this circuit given in Figure 2 onto experiment board. Make sure that all ICs have supply and ground connections. Connect inputs to switches, and connect outputs to LEDs. Apply input combinations to your circuit and check that whether your circuit gives same outputs in Table 2.

3. Equipment List

Equipment	Quantity	Library
7408 – AND tümdevresi	1	logi7400dip
7432 – OR tümdevresi	2	logi7400dip
7486 – EXOR tümdevresi	1	logi7400dip

Experiment 3 Combinational Circuit Synthesis and Analysis Using MSI Decoders, Multiplexers and Encoders

1. Introduction

Decoders are the MSI circuit elements that have n inputs and 2^n outputs. Also, decoders with 4 inputs and 10 outputs (4x10) are available for the decimal (Binary Coded Decimal (BCD): 0-9) numbers. For each input of the decoder, only the equivalent output in the decimal number of the given input is active, while the other outputs are inactive. There are two types of decoders according to active-0 and active-1. Active output is logic-0 in active-0 decoders, while the other outputs are logic-1 (Similarly, active output is logic-1 in active-1 decoders, while the other outputs are logic-0). In this way, each output generates maxterm (minterm) of given input. Therefore, any Boolean function can be implemented using decoders. Number of inputs and outputs of decoder can be increased using multiple decoder with different number of inputs and outputs.

Multiplexers are used to select one of its inputs and send it to the output. Multiplexing is a process for sending large number of information over less number of channel. In this way, lots of information can be transferred on one datapath in the desired order. Therefore, multiplexers are also called as data collectors. On the other hand, multiple data can be transferred on single datapath by multiplexing using demultiplexer. Multiplexers have n control inputs and 2^n data inputs. One of the 2^n data inputs transferred to output by changing the value of n control inputs. The index of input, which the data is to be transferred to the output, equals to the decimal value of control input. In this way, Boolean functions with n variable can be implemented using 2nx1 multiplexer. Number of inputs and outputs of multiplexer can be increased using multiple multiplexer with different number of inputs and outputs.

Encoders have 2^n inputs and n outputs. Encoders have an opposite functionality than decoders. When one of the inputs of encoder is active (If there is an active-0 encoder, only one of the inputs has logic-0 value, else if there is an active-1 encoder, only one of the inputs has logic-1 value.), the output of the encoder takes a value of index of input on binary form. For example, if the input value (x7x6x5x4x3x2x1x0 : 00010000) is given to active-1 encoder, the output is $(3)_{10} = (011)_2$. If there are more than one active inputs, the output is undefined. Priority encoders allow to use more than one active inputs. Then, priority encoders select one of the active inputs with priority information, and they ignore other active inputs. Priority ordering is usually from the largest entry index to the smallest entry index. Priority encoders are generally used in the microprocessor to control interrupts.

2. Experiment Assignments

2.1. Step 1

Design the functions f1 and f2 with decoders and AND gates using the truth table given in Table 1. The circuit is shown in Fig. 1.

Table 1 : Truth Table of f1 and f2

x_2	x_1	x_0	f_1	f_2
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

Implement the circuit given in Fig. 1. Make sure that all ICs have supply and ground connections. Connect appropriate logic values to control inputs of decoder. For this purpose, you need to look data sheet of decoder. Connect inputs to switches (S0-S2), and connect outputs to LEDs (L0-L1). Determine the circuit function with the help of truth table.

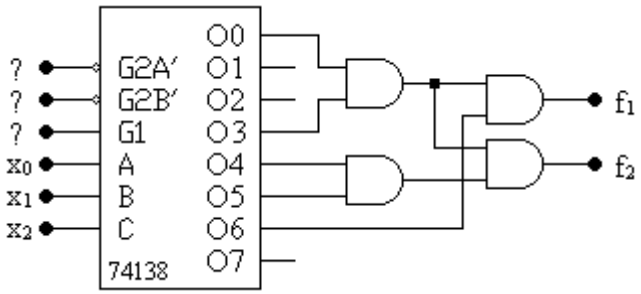


Figure 1: Circuit of f1 and f2 functions given in Table 1 with 74138 decoder.

Step 2

Table 1 Design the functions f1 and f2 with multiplexers using the truth table given in Table 1. The circuit is shown in Fig. 2.

Implement the circuit given in Fig. 2. Make sure that all ICs have supply and ground connections.

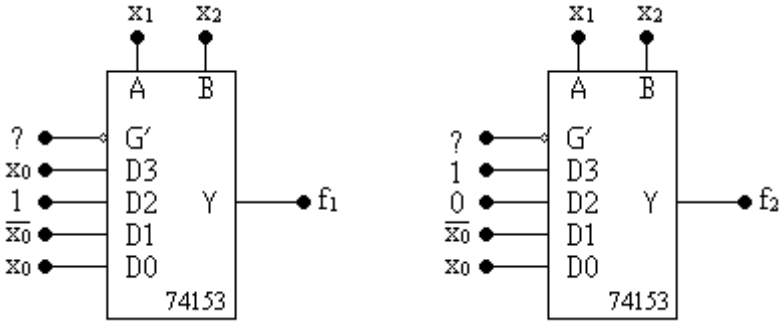


Figure 2: Circuits of f1 and f2 functions given in Table 1 with 74153 multiplexer

Connect appropriate logic values to control inputs of encoder. For this purpose, you need to look data sheet of decoder. Connect inputs to switches, and connect outputs to LEDs. Determine the circuit function with the help of truth table.

2.2. Step 3

Analyze the circuit given in Fig. 3, which have 74138 decoder with active-0 output and 74148 priority encoder with active-0 input and output.

Implement the circuit given in Fig. 3. Make sure that all ICs have supply and ground connections. Connect inputs to switches, and connect outputs to LEDs. Fill the truth table given in Table 2, and determine the circuit function with the help of truth table.

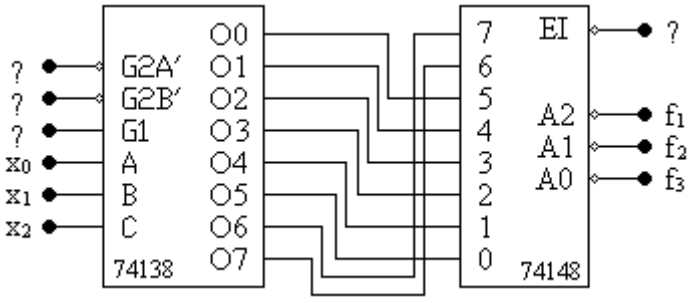


Figure 3: The circuit whose function is to be determined.

Table 2: Truth table of the circuit given in Fig. 3

x_2	x_1	x_0	f_1	f_2	f_3
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			

1	1	1			

3. Equipment List

Equipment	Quantity	Library
7404 – NOT IC	1	logi7400dip
7408 – AND IC	1	logi7400dip
74138 – Decoder IC	1	logi7400dip
74148 – Priority Encoder IC	1	**7400-lib****
74153 – Multiplexer IC	1	logi7400dip

1. Introduction

Using block structures (iterative networks) is more suitable to implement some multivariable functions, because of these functions nature. Accordingly, the circuits with desired input and output numbers can be produced by doing right connections between equivalent circuits. This method is much easier than the other methods for this implementations. As known, when the input number increases, the output number of a function also increases exponentially. However, in iterative networks, only one basic module is designed and other high level circuits can be designed connecting these modules. For instance, adder, subtractor, multiplier, comparator and similar circuits can be designed using iterative networks.

A half adder is the most basic block structure of an adder circuit. The half adder circuit has two inputs, A and B, and two outputs E and T. A and B are addend bits, T is Sum and E is Carry bits. Figure 1, shows truth table and design of the half adder circuit.

A	B	E	T
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

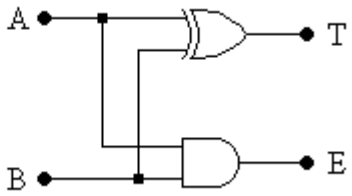


Figure 1: Truth table and circuit of a half adder.

In an adding operation if addends have more than one bit, carry bits should be taken into consideration. The truth table and the full adder circuit which was made with half adders are shown in the figure 2. Full adders are blocks which has 3 inputs and 2 outputs. Unlike the half adders, in full adders the modules which is doing addition, can be made by using a "carry-in" input.

E_0	A	B	T	E_1
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

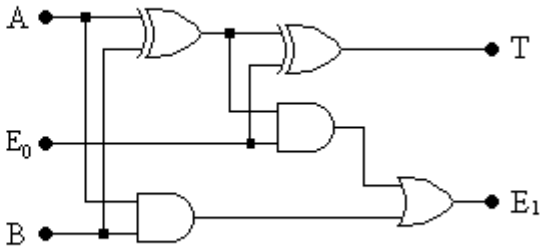


Figure 2: truth table and circuit of a full adder.

A *n*-bit parallel adder circuit can be made by transferring the "carry-out" bits from LSB to MSB as shown in Figure 3. The *n*-bit parallel adder works slow because the *n'th* full adder block waits for *(n-1)* full adders to generate sum bit. Parallel adder circuits are designed using look ahead carry adder to avoid this disadvantage.

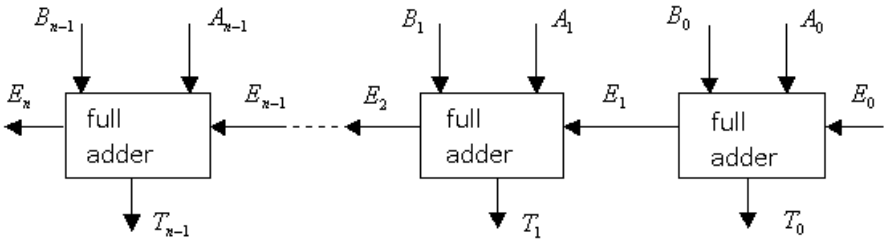


Figure 3: *n*-bit parallel adder

Subtractor circuits can be made using n's complement or (n-1)'s complement addition methods. Digital systems generally use binary system so the subtraction can be done using 1's complement or 2's complement methods. 2's complement of B can be calculated as $B_2 = 2^n - B$ *(B:number, n: digit number)*. The subtraction can be done as adding 2's complement of subtrahend to minuend. Thus, $T = A + B_2$ addition is equal to $A + (2^n - B) = 2^n + (A-B)$ *(A:minuend, B:subtrahend).* Accordingly,

i) if A≥B, the sum T, will produce an end carry 2^n, which can be discarded; what is left is the result A-B.

ii) if A<B sum T, does not produce an end carry and is equal to $2^n + (A-B) = 2^n - (B-A)$, which is the 2's complement of (A-B). To obtain the answer in a familiar form, take the 2's complement of the sum and place a negative sign in front. Similarly, subtraction can be done using 1's complement.

Figure 4 shows the design of an addition and 2's complement subtraction circuit using 74283 IC. Design has a control input to select addition or subtraction operations.

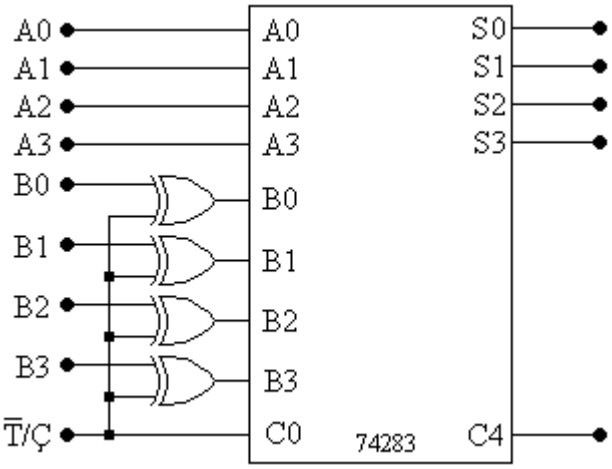


Figure 4: Adder/Subtractor circuit design (using 74283 IC).

Comparators are combinational logic circuits that are used for testing whether one number is greater than, less than, or equal to the other number. Compare operation can be done iteratively from MSB to LSB or viceversa. 1-bit comparator, compares individual bits and *n-bit* comparator can be produce by cascading together n of these modules as shown in Figure 5.

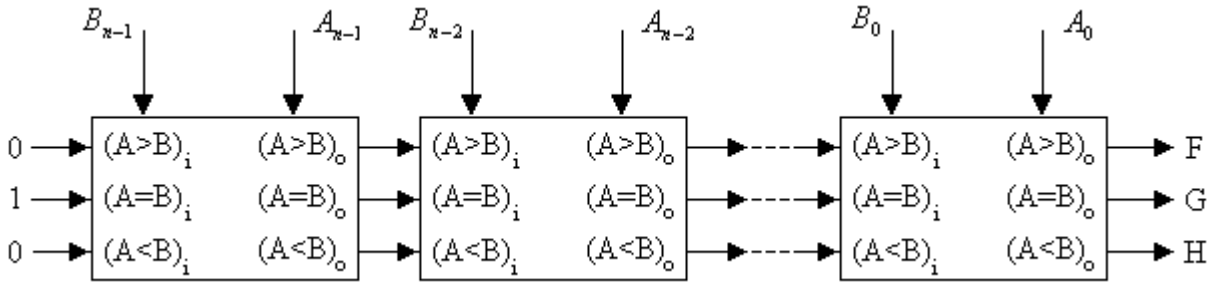


Figure 5: Structure of an iterative circuit for n-bit comparator (from MSB to LSB)

Comparison starts with giving $(A>B)_i = 0$, $(A=B)_i = 1$ and $(A<B)_i = 0$ values to input of *n'th* basic module. This operation requires two basic blocks. First block compares the first "i" input bits (A_i and B_i) then produces output bits ($a_i : A_i > B_i$ $b_i : A_i < B_i$ and $c_i : A_i = B_i$) according to relation.

Second basic block compares a_i, b_i, c_i bits and f_{i+1}, g_{i+1}, h_{i+1} bits then produce f_i, g_i, h_i bits as outputs. This a_i, b_i, c_i bits are first basic block's outputs. f_{i+1}, g_{i+1}, h_{i+1} bits represent previous module's outputs, this outputs produce by comparing number A and B from *(n-1)'th* bit to *i'th* bit. Figure 6 shows block diagram of the comparator circuit's basic module.

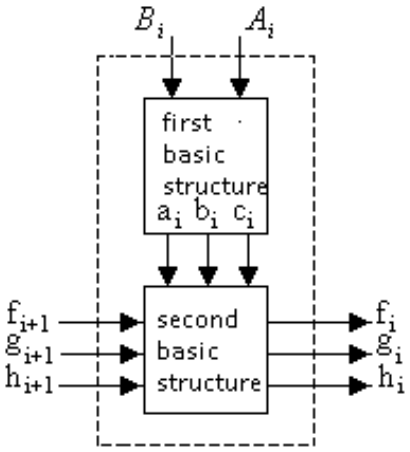


Figure 6: Block diagram of the comparator circuit's basic module.

In first basic block, a_i, b_i and c_i outputs can be found as:

$$a_i = \begin{cases} 1 & if \quad A_i > B_i \\ 0 & otherwise \end{cases} \qquad b_i = \begin{cases} 1 & if \quad A_i < B_i \\ 0 & otherwise \end{cases} \qquad c_i = \begin{cases} 1 & if \quad A_i = B_i \\ 0 & otherwise \end{cases}$$

If $A_i > B_i$ then A_i =1, B_i =0. Therefore $a_i = A_i . \overline{B_i}$. i = 0, 1, 2, ..., (n-1)

If $A_i < B_i$ then A_i =0, B_i =1. Therefore $b_i = \overline{A_i} . B_i$. i = 0, 1, 2, ..., (n-1)

If $A_i = B_i$ then A_i =0, B_i =0 or A_i =1, B_i =1.

Therefore $c_i = \overline{A_i} . \overline{B_i} + A_i . B_i = \overline{A_i \oplus B_i} = \overline{a_i + b_i}$. i = 0, 1, 2, ..., (n-1)

In second block, f_i, g_i and h_i outputs can be found as:

If $A_{n-1} A_{n-2} ... A_i > B_{n-1} B_{n-2} ... B_i$ then f_i=1,

If $A_{n-1} A_{n-2} ... A_i = B_{n-1} B_{n-2} ... B_i$ then g_i=1,

If $A_{n-1} A_{n-2} ... A_i < B_{n-1} B_{n-2} ... B_i$ then h_i=1.

f_i = 1 when g_{i+1} = 1 and a_i = 1 or f_{i+1} = 1 $\Rightarrow$ $f_i = f_{i+1} + g_{i+1}.a_i$

g_i = 1 when g_{i+1} = 1 and c_i = 1 $\Rightarrow$ $g_i = g_{i+1}.c_i$

h_i = 1 when g_{i+1} = 1 and b_i = 1 or h_{i+1} = 1 $\Rightarrow$ $h_i = h_{i+1} + g_{i+1}.b_i$

Figure 7 shows, design of the gate elements of the comparator circuit's basic module using expression given above. In this experiment 7485 IC is used as 4-bit comparator circuit.

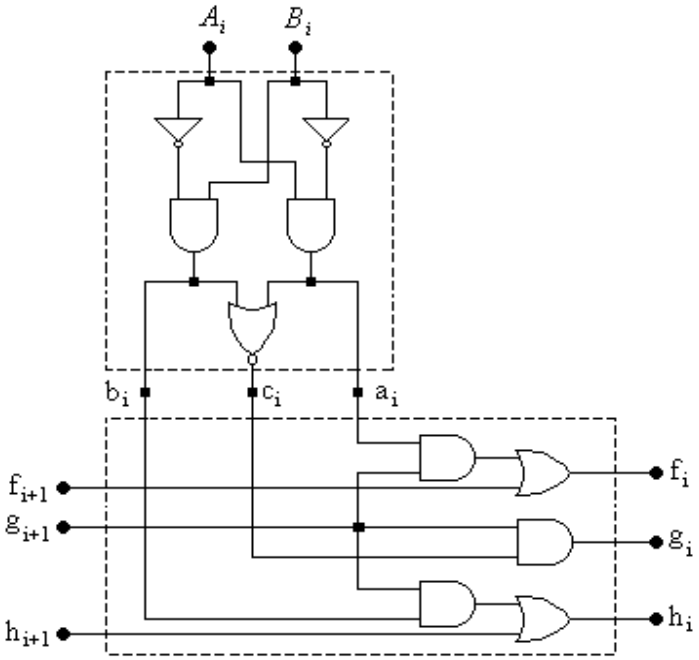


Figure 7: Implementation of basic comparator module using logic gates.

2. Experiment Assignments

2.1. Step 1

Implement the parallel 2bit adder circuit which you designed before this experiment. Make sure that all ICs have supply and ground connections. Connect inputs to switches, and outputs to LEDs. Check that whether your circuit works correctly.

2.2. Step 2

Implement this circuit given in Figure 4 onto experiment board. Make sure that all ICs have supply and ground connections. Connect inputs to switches, and outputs to LEDs. Fill the Table 2 given below.

Table 1: Result table of Adder/Subtractor circuit

T/Ç	A	B	$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	C_4	$S_3S_2S_1S_0$
0	8	7				
0	11	12				
0	3	4				
1	1	5				
1	6	6				
1	14	9				

2.3. Step 3

Place the 7485 IC onto experiment board. Make all necessary connections. Connect inputs to switches, and outputs to LED. Fill the table 2.

Table 2: Result table of comparator circuit.

A	B	$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	A=B	A>B	A<B
8	3					
4	4					
12	15					
2	13					
11	11					
10	0					

3. Equipment List

Equipment	Quantity	Library
7408 – AND IC	1	logi7400dip
7432 – OR IC	1	logi7400dip
7485 – 4-bit comparator IC	1	logi7400dip
7486 – EXOR IC	1	logi7400dip
74283 – 4-bit binary adder IC	1	logi7400dip

1. General Explanations

The outputs of the combinational circuits depend only on the input values at that moment, while the outputs of the sequential circuits are dependent on the input values and states at that moment. Thus, sequential circuits (machines) contain memory elements that store past states, unlike combinational circuits. There are two types of sequential circuits, Mealy and Moore, depending on the output types. The outputs of the Mealy type sequential circuit are dependent on the inputs and conditions at that moment. In the Moore type sequential circuit, the outputs are only dependent on the states at that moment. This situation is shown in **Figure 1.**

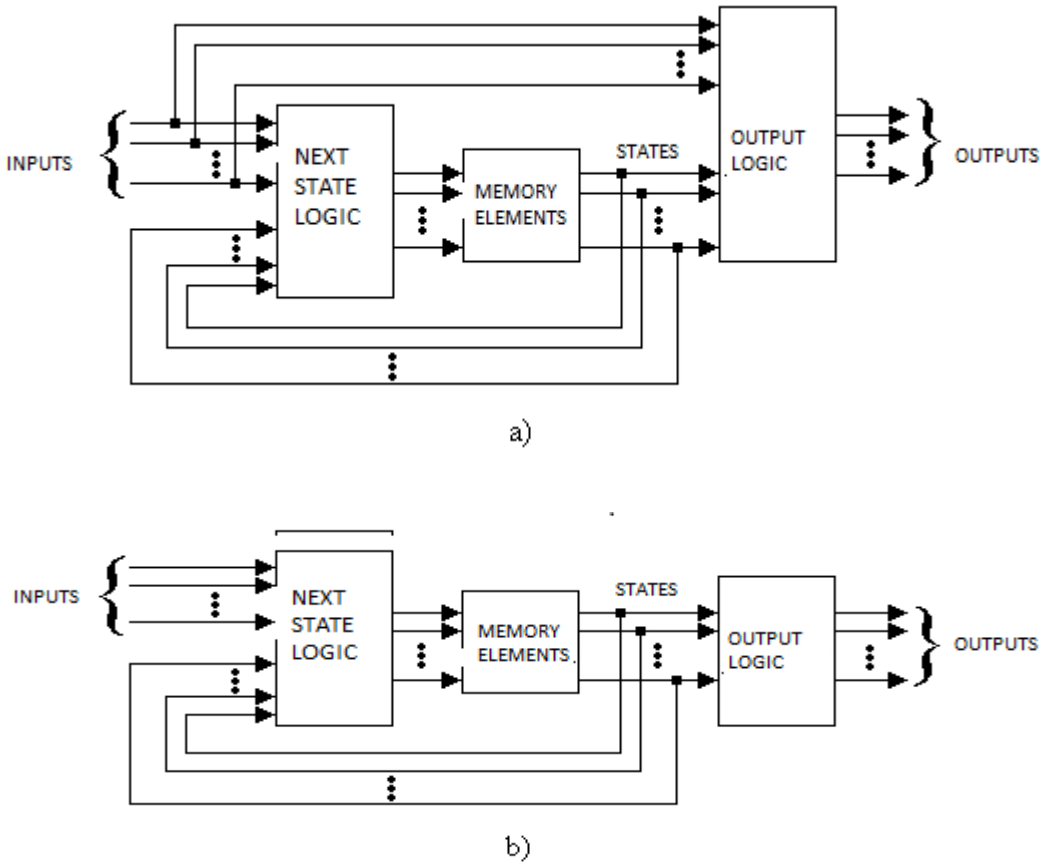


Figure 1: a) Mealy type circuit model , b) Moore type circuit model.

Sequential circuits are divided into two types; asynchronous and synchronous depending on whether they are oscillated or not. In asynchronous sequential circuits, there is no central clock and state transitions are provided by changing the input values. In synchronous sequential circuits, there is a central clock that generates periodic clock pulses, which are connected to the clock inputs of all memory elements. The circuit changes state only when it is triggered by the clock, and the new state depends on the inputs and conditions at which the trip is triggered. The other clock pulse protects the circuit state until the next millennium. If synchronous sequential circuit has unused conditions and the

circuit can not go back to the used states that when it goes to one of these states, then it is stated that the type of circuit is locked.

The analysis stages of a synchronous sequential circuit are generally as follows:

- With the aid of the given circuit, the output functions of the input and sequential circuits of the memory elements are determined in terms of the current state and the input variables of the sequential circuit.
- State equations are obtained by using the input functions of the memory elements and the definition relations. State equations are statements that specify the next state. The definition associations of the memory elements are Q + = JQ '+ K'Q, Q + = D, Q + = S + R'Q and Q + = TQ' + T'Q.
- The status table or state diagram is generated with the state equations obtained and the output functions of the sequential circuit.

In synchronous sequential circuits, when the initial state and the input sequence are given, the diagrams that show the change of the next states and outputs with respect to the time are called the time diagram. Physically, the inputs of a synchronous sequential circuit can not be changed at the same time as the clock signal is triggered. It can only be changed after the time signal is triggered, that is, after the desired states are met. Because the outputs of the Mealy type circuit are also connected to the inputs at that point, unwanted values can be seen in the circuit outputs during this process. This process is called the critical time interval. In this time interval, the output of the circuit is wrong. Faulty outputs are divided into hazardous or non-hazardous. Faulty outputs are also divided into two according to whether the incorrect output has a value of 0 or 1. If there is a continuous change in the output before the critical time interval, after the critical time interval and the critical time interval, or when the change is 010 or 101, it means that this output has the wrong harmful 1 or 0 value respectively. In other cases, the output is erroneously harmless. Moore-type machines do not have such faulty outputs. Mealy-type machines have various methods to get rid of faulty outputs. One of these methods is the Moore machine, which takes into consideration the increase in the number of states, the function realized by the Mealy machine. In **Figure 2**, hazardous and non-hazardous outputs are shown and it is assumed that the state transitions are on the rising edge of the clock signal.

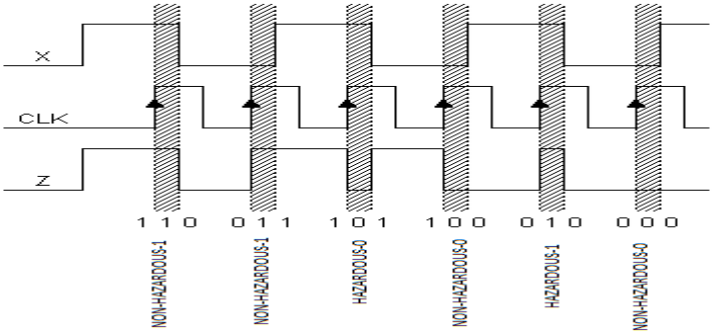
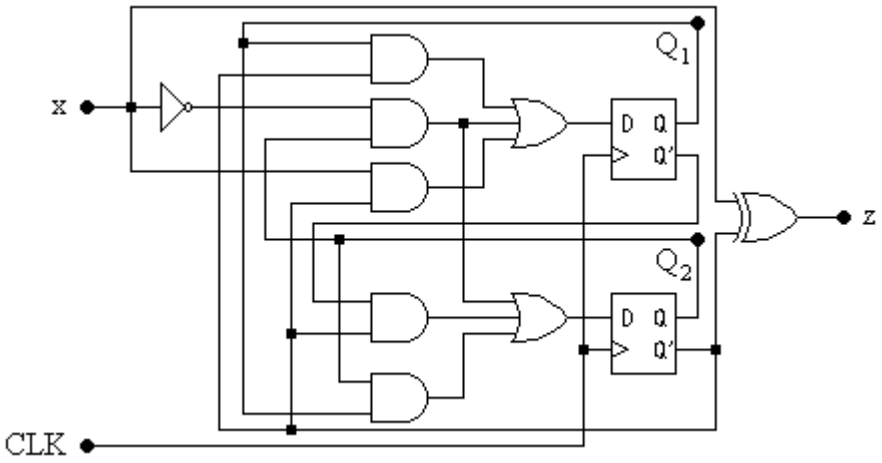


Figure 1: Hazardous and non-hazardous outputs on time diagram.

The circuit diagram for the Mealy machine to be analyzed in this experiment is given in Figure 3:



Şekil 2: The circuit that will be analyzed.

Theoretically, the number of gates to be used when synthesizing with SSI elements, and the number of gate input fans and memory elements are to be minimized. In practice, however, the concept of minimality is related to the number of integrated circuits since integrated circuits are used. A total of 6 integrated circuits are required for the circuit shown in Figure 3, while a total of 3 integrated circuits are required for the circuit in Figure 4, which is designed with NAND and XOR gates using common structures for common components and performing the same kind of gate transformation.

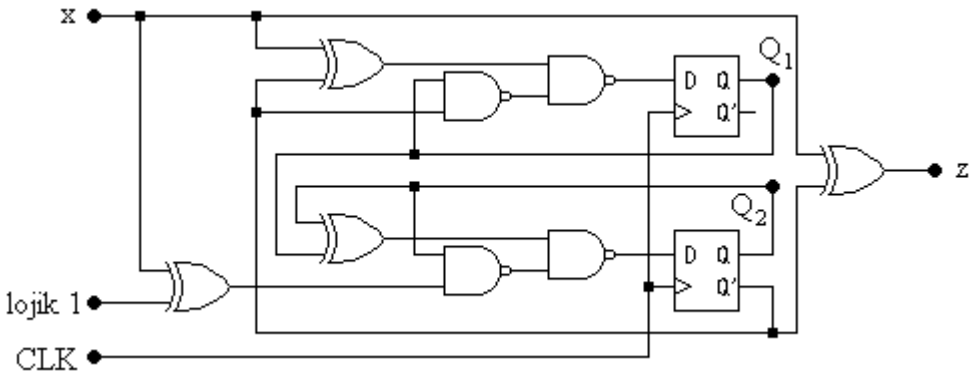


Figure 3: Realization of the circuit in Figure 3 by using NAND and XOR gates.

Input and output functiions of memory elements for the circuit in Figure 3:

$D_1 = Q_1.Q_2' + x'.Q_2 + x.Q_2'$ $D_2 = x'.Q_2 + Q_1'.Q_2' + Q_1.Q_2$ $z = x \oplus Q_2'$ (1)

State equations by using D-type memory elements ;

$Q_{+1} = Q_1.Q_2' + x'.Q_2 + x.Q_2'$ $Q_{+2} = x'.Q_2 + Q_1'.Q_2' + Q_1.Q_2$ (2)

State transitions and output of the circuit according to the all input combinations are given in the following **Table 1:**

Tablo 1: State transitions and outputs for the circuit in Figure 4.

x	Q_1	Q_2	Q_{+1}	Q_{+2}	z
0	0	0	0	1	1
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	1	1

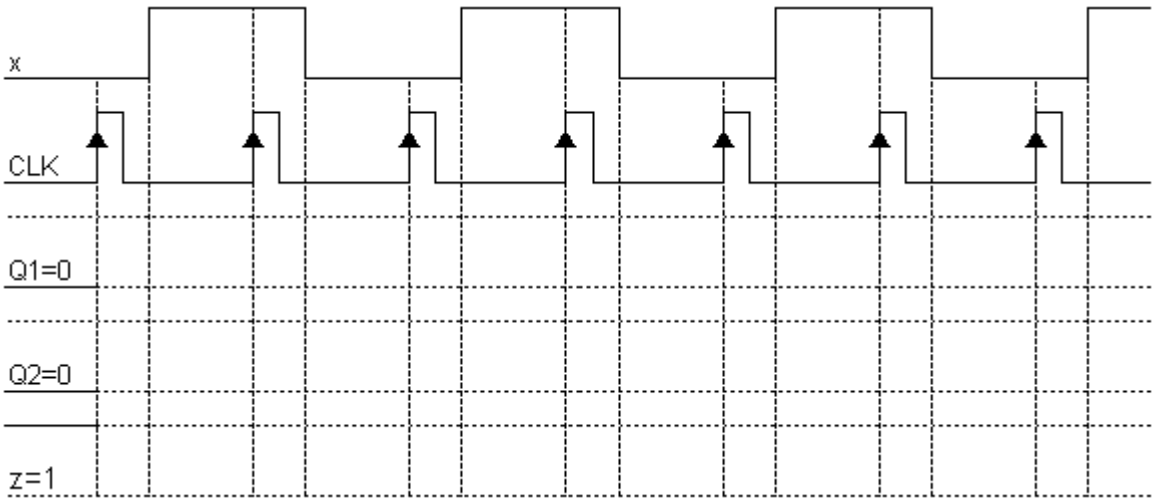
1. During the Experiment

1.1. Step 1

Setup the circuit in Figure 4 to the experiment set. Make source and ground connections for all integrated circuits. The input of the circuit will be taken from the logic switches. Connect the preset and clear inputs of the memory elements to the logic switches to get the desired initial state. Take clock inputs of memory elements from common **debounce push button**. Connect the circuit output and outputs of the memory elements to the LEDs. Determine whether the circuit is operating according to the state diagram by using the clear and preset inputs of the memory elements and observing the outputs of the memory elements and the circuit from the LEDs.

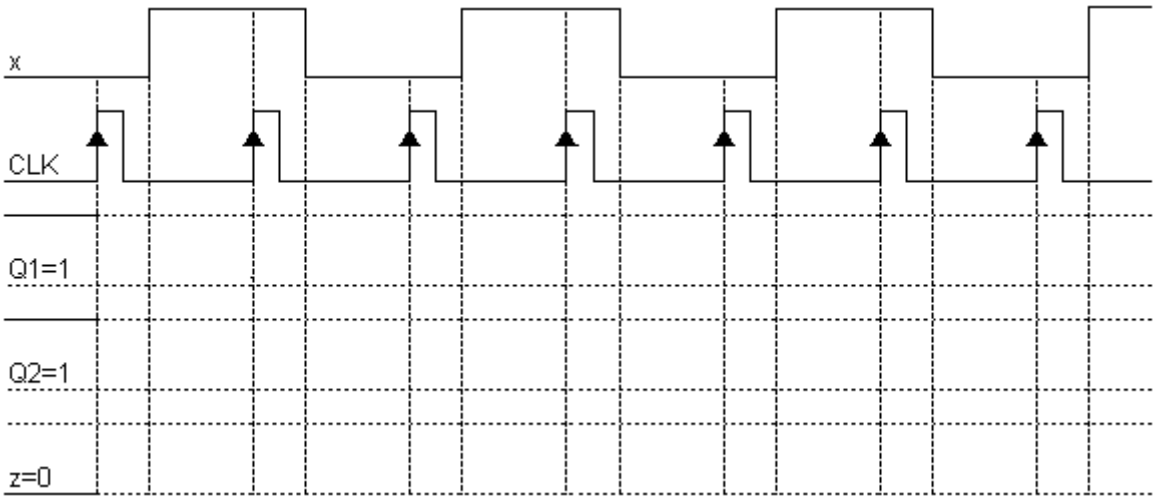
1.2. Step 2

Complete the time diagrams for the initial state Q_1Q_2 = 00. Specify the types of faulty outputs that will occur.



1.3. Step 3

Complete the time diagrams for the initial state Q_1Q_2 = 11. Specify the types of faulty outputs that will occur.



2. Meterial List

Meterial	Quantity	Library
7400 – NAND IC	1	logi7400dip
7474 – D –Type Memory IC	1	logi7400dip
7486 – XOR IC	1	logi7400dip

1. Introduction

There are different methods for designing sequential circuits. The steps of one of these methods are given below.

1. Obtaining the state diagram of the sequential function from the verbal definition.
2. State reduction on the state diagram obtained in step 1.
3. State assignment
4. Generation of the state truth table.
5. Selection of the memory elements to be used in the design of the sequential circuit.
6. Formation of state transition table of the sequential circuit and determining memory input and sequential circuit output function using memory functions.
7. Implementation of sequential circuit.

Verbal Definition: Design a 3 bit even parity generator using sequential circuit elements. This sequential circuit has one serial 'x' input and 'Z' output. When sequential circuit takes 3 bits input from its serial input, if the number of ones in the 3-bit input is odd, Z takes the value of 1, else it takes the value of 0. Also, Z takes value of 0 for the intermediate states. After taking 3-bits input and generating the output, the sequential circuit will be returned to initial state to wait new input sequences. Fig. 1 shows the state diagram of given verbal definition.

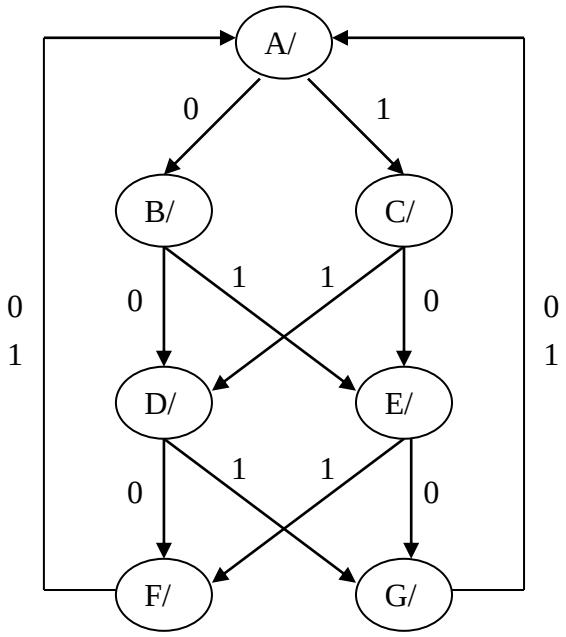


Figure 1: State Diagram of 3-Bit Even Parity Generator

State reduction eliminates the equal states. After state reduction, states given in Fig. 1 are coded as A : 000, B : 010, C : 011, D : 110, E : 111, F : 100 and G : 101. Then, state and state transition tables are shown in Fig. 2.

$Q_1Q_2Q_3$ \ x	0	1
A : 000	010,0	011,0
B : 010	110,0	111,0
C : 011	111,0	110,0
D : 110	100,0	101,0
E : 111	101,0	100,0
F : 100	000,0	000,0
G : 101	000,1	000,1

$Q_{+1}Q_{+2}Q_{+3}$,Z

K : Keyfi

x	Q_1	Q_2	Q_3	Q_{+1}	Q_{+2}	Q_{+3}	Z
0	0	0	0	0	1	0	0
0	0	0	1	K	K	K	K
0	0	1	0	1	1	0	0
0	0	1	1	1	1	1	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	1	0	0	0
0	1	1	1	1	0	1	0
1	0	0	0	0	1	1	0
1	0	0	1	K	K	K	K
1	0	1	0	1	1	1	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	1
1	1	1	0	1	0	1	0
1	1	1	1	1	0	0	0

Figure 2: State and Transition Tables of Synchronous Sequential Circuit

JK flip-flop is selected to design given sequential circuit. The inputs of JK flip-flops ($J_1 : Q_2$, $K_1 : Q_2'$, $J_2 : Q_1'$, $K_2 : Q_1$, $J_3 : x.Q_1' + x.Q_2$, $K_3 : x + Q_2'$) and output of sequential circuit ($Z : Q_2'.Q_3$) are determined using JK flip-flop function and state transition table. Fig. 3 shows the design of the sequential circuit using NAND gates and JK flip-flops.

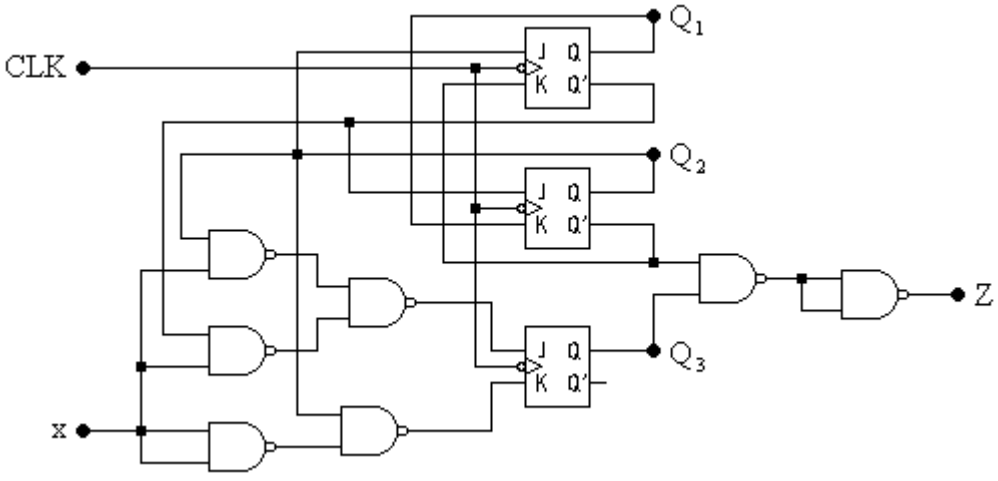


Figure 3: Design of 3-Bit Even Parity Generator Using NAND Gates and JK Flip Flops

2. Experiment Assignments

2.1. Step 1

Implement the circuit given in Fig. 3. Make sure that all ICs have supply and ground connections. Connect x input to logic switch and connect CLK input to debounce pushbutton. Connect preset and clear inputs of FFs to logic switches. Connect the output of sequential circuit and outputs of FFs to LEDs and check your state table.

2.2. Step 2

Apply 0101000111010100101 (first bit is 1) sequence to x input starting from the state 000. Send a bit, before the falling edge of each clock pulse. Then, determine the output sequence.

3. Equipment List

Equipment	Quantity	Library
7400 – NAND IC	2	logi7400dip
7476 – JK FF IC	2	logi7400dip

1. Introduction

Sequential circuits that repeat at least one state sequence are called counters. Counters can repeat various status sequences. Accordingly, the counters can be in various types such as forward, backward, forward / backward, programmable, binary code, BCD, Gray. Counters are categorized as asynchronous and synchronous according to the triggering of the memory elements. In asynchronous counters, a memory element is triggered by the output of the first less significant memory element. The least significant memory element is triggered by the applied pulse (or clock). In Figure 1, modulo 16 (2^4) is given an asynchronous forward counter.

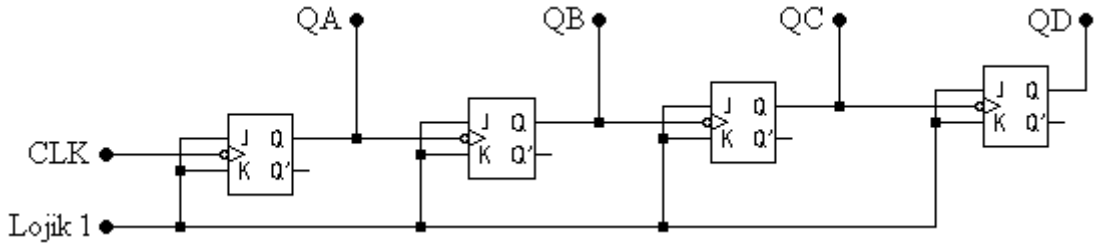


Figure 1: 4-bit asynchronous forward counter circuit.

The J-K memory elements in Figure 1 are triggered by falling edges and inputs J and K are assigned as logic 1 value. Thus, the J-K memory element outputs complement the previous outputs (Q). The time diagram of the asynchronous counter is given in Figure 2.

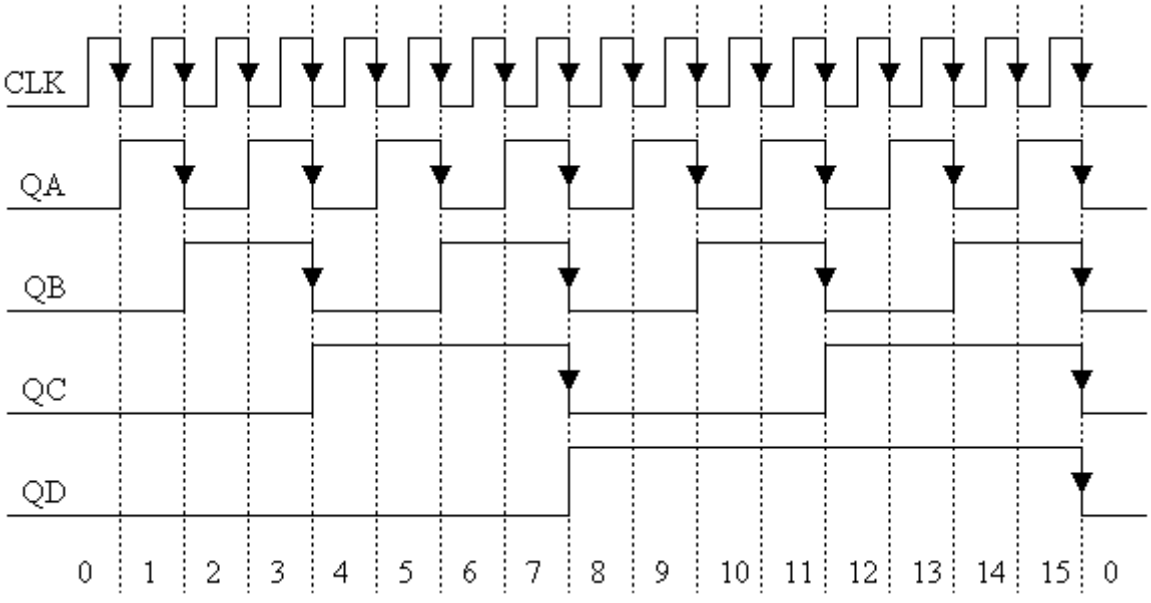


Figure 2: 4-bit asynchronous forward counter time diagram.

The integrated circuit 7493 can be shown as an example asynchronous circuit. 7493 is a 4-bit asynchronous counter. There are 2 reset inputs as the control input of this unit, RO1, RO2, two clock inputs, CKA, CKB and four outputs, QD, QC, QB and QA (MSB: QD). Both RO1 and RO2 reset inputs are active at a logic 1 value, and when these two control inputs have logic 1, the outputs receive logic

0, meaning the counter is reset. When the reset inputs are inactive and the QA output is connected to the CKB, the 7493 integrated circuit behave as a modulo 16 counter. This is illustrated in Figure 3a. In Figure 3b, a 7493 integrated circuit and an AND gate forms a BCD counter. Accordingly, when the BCD counter is being asserted, when the outputs are 1010 (QD and QB logic 1), the output of the AND gate will be logic 1 and the reset inputs will reset the outputs of all the arrays and counting will continue.

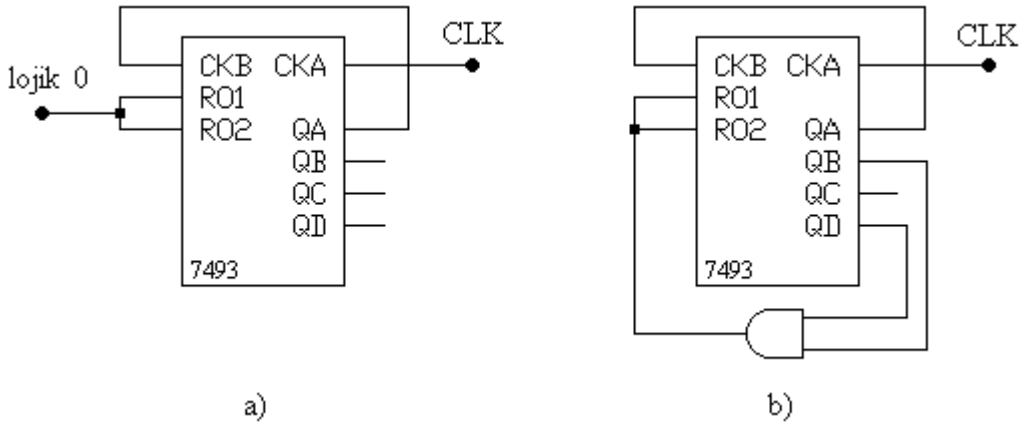


Figure 3: a) Modulo 16 counter with 7493 IC, b) BCD counter with 7493 IC.

The 0-99 counter obtained by the BCD counter unit block shown in Figure 3b is shown in Figure 4.

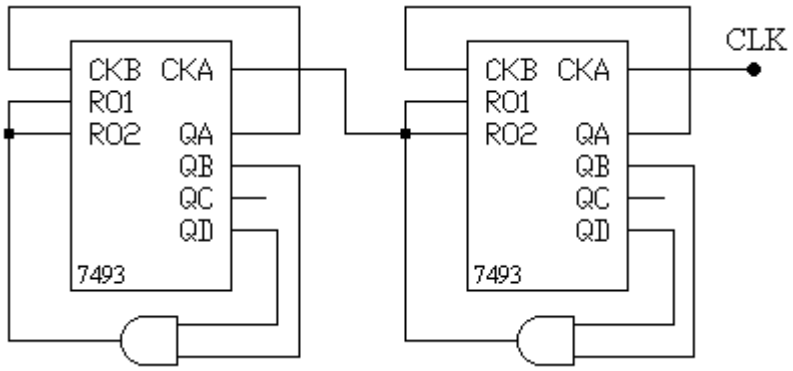


Figure 4: 0-99 counter implemented by BCD counter shown in Figure 3b.

In synchronous counters, a central clock triggers all memory elements at the same time. As a result, synchronous counters are faster than asynchronous counters. Figure 5 shows a 4-bit synchronous forward counter circuit.

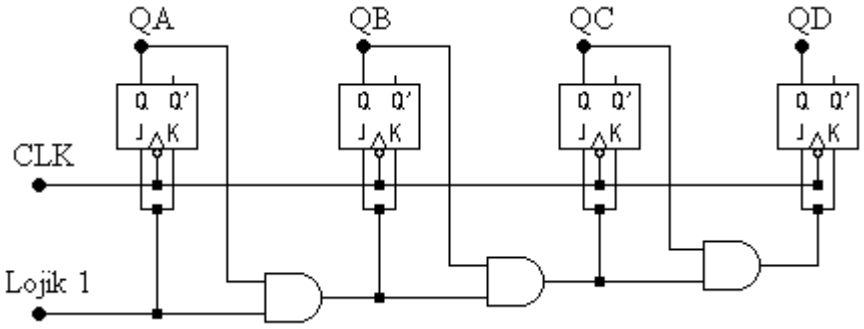


Figure 5: 4-bit synchronous forward counter.

The 4-bit forward / reverse counter circuit designed by the circuit shown in Figure 5 is shown in Figure 6.

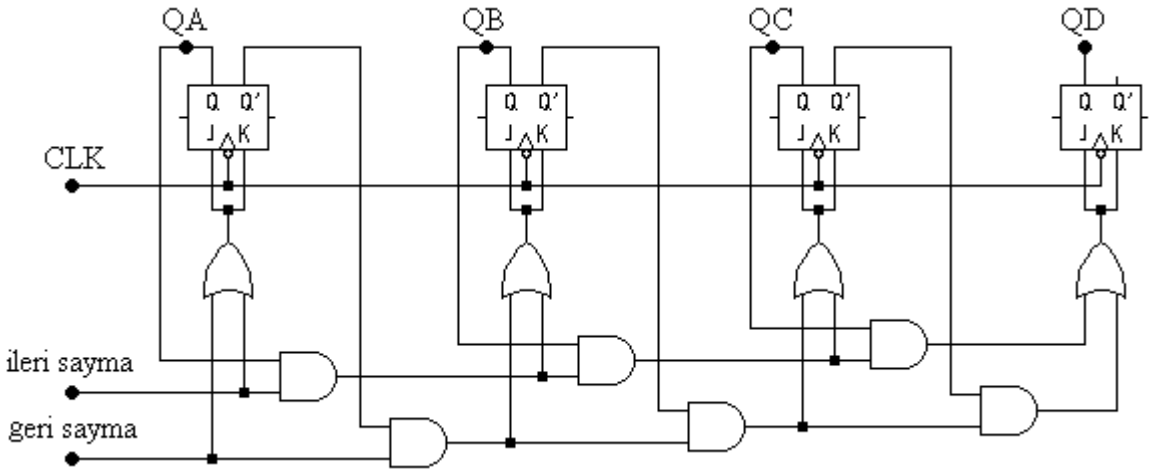


Figure 6: 4-bit synchronous forward / reverse counter.

In addition to the forward / backward counting ability, the synchronous counter can be provided with parallel loading capability, so that the counter can start from a desired number. 74161 is an IC with a parallel load. There are four control inputs: ENableP (ENP), ENableT (ENT), Load and Clear along with a clock input, four-bit parallel output, four-bit parallel input and one-bit output. Clear and Load are active at logic 0 and ENP and ENT are active at logic 1. The Clear input is used to load a logic 0 value to the parallel outputs of the IC, while the Load input is used to load the value of the parallel input. The ENP and ENT control inputs are used to stop or resume counting. Figure 7 shows examples of counters designed with 74161 IC.

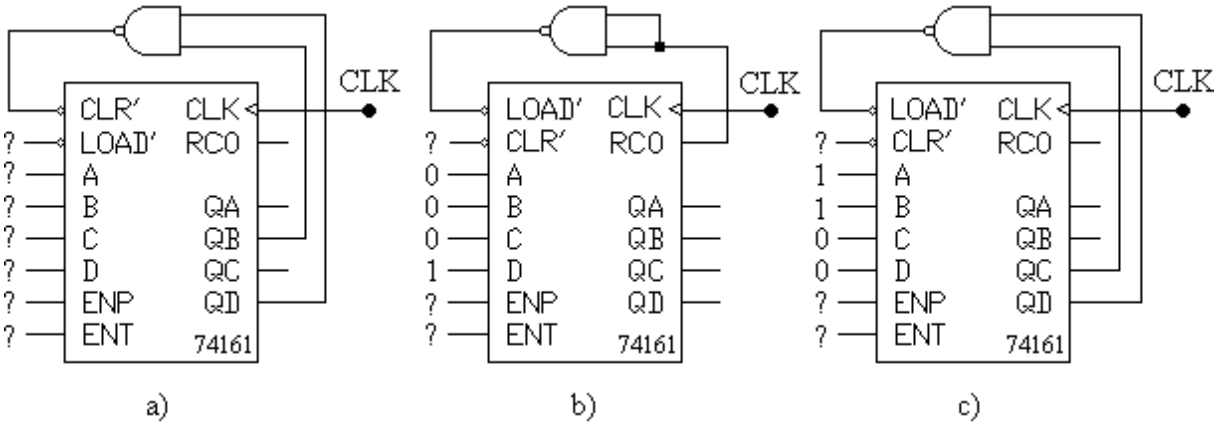


Figure 7: a) BCD counter, b) 8 to 15 counter, c) 3 to 12 counter.

Counters are generally used to determine the number of occurrences of an event or to obtain timing marks used to monitor operations in a digital system. These applications can be applications such as frequency division, information storage, pulse counting.

2. Experiment Assignments

2.1. Step 1

Install the circuit shown in Figure 1. Make all necessary connections of all the circuits. The clock input of the memory elements is taken from the 1 Hz TTL waveform signal. After giving the appropriate values to the inputs of the memory elements, observe the outputs from the LEDs to determine if the circuit performs the desired function.

2.2. Step 2

Install the circuit shown in Figure 3b. Make all necessary connections at all the circuits. Time clock input is taken from the 1Hz TTL waveform signal. By observing the outputs of the entire device from the LEDs, you determine whether your circuit is performing the desired function.

2.3. Step 3

Set the circuit shown in Figure 5. Make all necessary connections of all the circuits. The clock input of the memory elements is taken from the 1 Hz TTL waveform signal. By observing the outputs of the memory elements from the LEDs, you determine whether the circuit is performing the desired function.

2.4. Step 4

Install the circuits shown in Figures 7a, b, c respectively. Make all necessary connections of all the circuits. Take the clock input from the 1Hz TTL waveform signal. By observing the outputs of the entire device from the LEDs, you determine whether your circuit is performing the desired function.

3. Equipment List

Equipment	Quantity	Library
7400 – NAND IC	1	logi7400dip
7408 – AND IC	1	logi7400dip
7476 – JK memory IC	2	logi7400dip
7493 – Asynchronous counter IC	2	**logi7400ic****
74161 – Synchronous counter IC	1	logi7400dip