Experiment 5 Sequential Lock

1. **Purpose**
   1) To familiarize and use shift register 74LS194.
   2) To know serial/parallel in, serial/parallel out in digital circuit.
   3) Learn to design, assemble and test synchronous timing sequence network by a LOCK system.
2. **Experiment Explanation**
   1) Outline of synchronous timing sequence network design

   (1) Logic description

   Based on practical situation, confirm state number and transforming conditions. Draw state flow chart and simplify it necessarily.

   (2) States naming

   Replace the states expressed by none-binary codes with binary codes, and consequently confirm the number of flip-flops needed. If the simplest states number is *N*, the number of flip-flops needed *M* will be $2^M \geq N$ or $M \geq \log_2 N$.

   (3) Flip-flop choosing

   Usually D or J-K flip-flops are used.

   (4) Getting driving equations

   According to state tables of chosen flip-flops, list logic equations of each flip-flops control terminals, and simplify them and get the simplest logic expressions.

   (5) Circuits construction and static and dynamic testing

   Now we will illustrate above design procedures by an example.

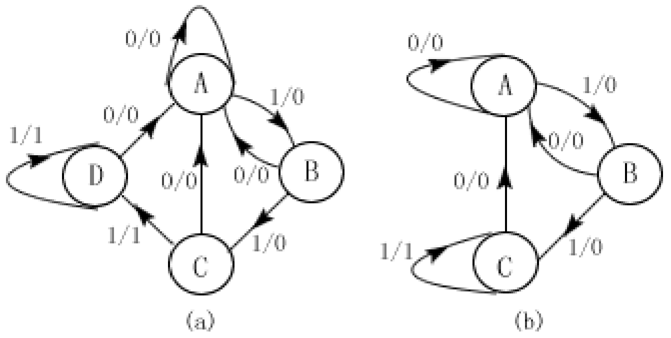
   Example: Design a "111" sequence picker whose output *Z* will be "1" when input X is "1" with 3 or more than 3 times in succession, otherwise *Z* will be "0".

   (1) Logic description

   There are 4 possible situations:

   a. Initial state named state A: all inputs are "0", and then output *Z* is "0".
   b. State B: Only one "1" comes and one "0" follows from input, and then output *Z* is "0".
   c. State C: Two "1" come in succession and one "0" follows, then output *Z* is "0".
   d. State D: Three or more "1" come in succession, then output *Z* is "1". Once "0" comes, system will return to initial state A.

   State flow chart can be drawn based on requirements, shown as figure 5.1. Each letter in the circle stands for a state. Lines and arrows between each state stand for states transition. The numbers beside lines are "inputs/output" showing state transition conditions and results.



(a) Basic chart (b) Simplified chart

Figure 5.1 State flow chart

From figure 5.1(a), we can see that there are same input/output relations for state C and D, so they are the same and can be simplified as figure 5.1(b).

(2) States naming

Because N=3, 2 flip-flops will be used. See table 5-1 for specific states naming.

(3) Flip-flop choosing

Here, D flip-flops are chosen.

Table5-1 States naming

State	Q_1	Q_2
A	0	0
B	0	1
C	1	0
None	1	1

(4) Getting driving equations

Table 5-2 lists states transitions based on states flow chart and function table of D flip-flop. Therein Q_1 and Q_2 are current states of flip-flops; Q_1' and Q_2' are next states, X and Z are input and output respectively. D_1 and D_2 are for next state:

$$D1=f(X, Q_1, Q_2)$$

$$D2=f(X, Q_1, Q_2)$$

These are driving equations.

Table 5-2 States transition table

X	Q_1	Q_2	Q_1'	Q_2'	D_1	D_2	Z
0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0
0	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
0	1	0	0	0	0	0	0
1	1	0	1	0	1	0	1

From table 5-2, we can get:

$$D_1= XQ_2\overline{Q}_1 + XQ_1\overline{Q}_2$$

$$D_2= X\overline{Q}_1\overline{Q}_2$$

And

$$Z= XQ_1 \bullet \overline{Q}_2$$

As the state $Q_1= Q_2=1$ is not used here, above three expressions can be simplified as:

$$D_1= X\left(Q_1+Q_2\right)=X\overline{\overline{Q_1}\bullet\overline{Q_2}}$$

$$D_2= X\overline{Q_1}\,\overline{Q_2}$$

$$Z= XQ_1$$

(5) Circuits construction

The logic circuit diagram of this synchronous timing sequence network is shown in figure 5.2, wherein Φ is system CLOCK.

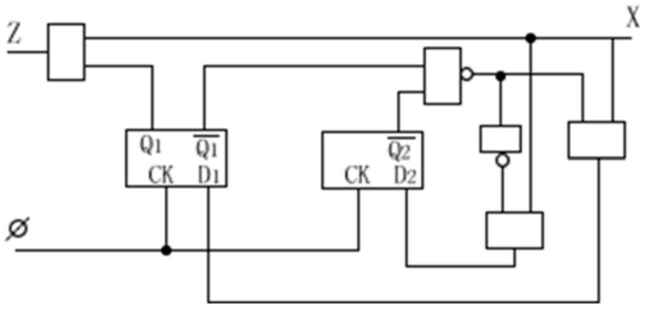


Figure5.2 Logic circuit diagram of synchronous timing sequence network

(6) Static and dynamic testing

Static testing: get *a* single pulse signal as Φ, and change value of X while observing Z. The main purpose is to test and verify the correctness of design. Dynamic testing: get two pulse signals as Φ and X to satisfy some logic relations. Observe Z on CRO, and change signal frequency to get relation between the circuit and work frequency.

2) Work principle of sequential lock

Figure 5.3 is the principle sketch of sequential lock, which consists of four parts: synchronous timing sequence network, codes setting, comparison, open and alarm.

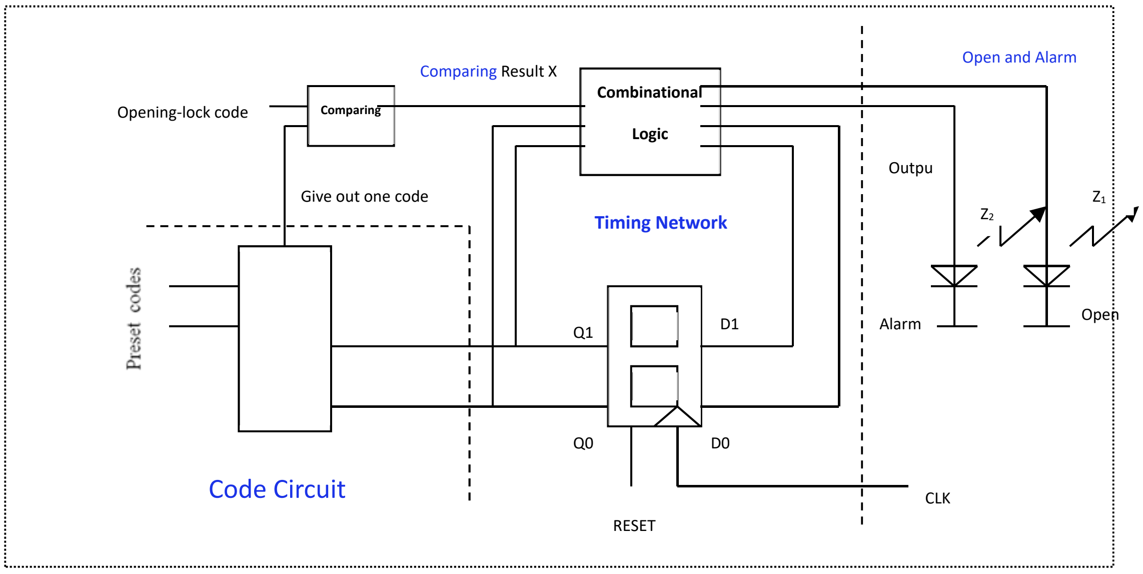


Figure5.3 Sequential lock constitution principle sketch

(1) Code circuit: firstly, lock codes are preset. Via comparison, when opening code and presetting code are identical, the lock will be opened, otherwise the system will give an alarm. Presetting codes can be changed.

(2) Comparing circuit: by CLOCK, opening codes compare with presetting codes one bit by one, and forward the results to the input of timing network. Once the result is not identical, timing network will give a alarm signal. If the result is identical, next bit will go on comparing till the last bit. If all opening codes are identical to presetting codes, timing network will give a signal of opening lock.

(3) Timing network: Its input is the comparison result, and it gives a opening lock signal or a alarm signal by states transmission. When a bit of opening code is identical to the corresponding presetting code, it will transfer to next state, controlling code circuit to output next bit which will compare with opening code again. Only until each bit of opening code is identical to presetting code, can timing network give a signal of opening lock. Once one bit is not identical, it gives an alarm signal immediately.

Now we will illustrate by flow chart and state chart, see figure 5.4 and figure 5.5.

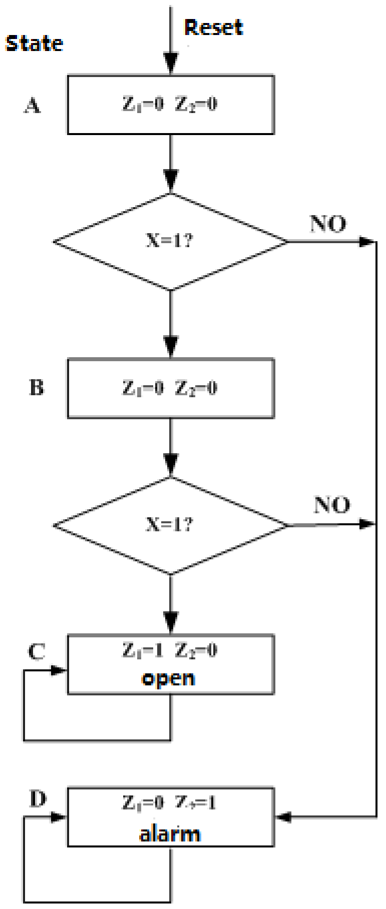


Figure 5.4 Flow chart

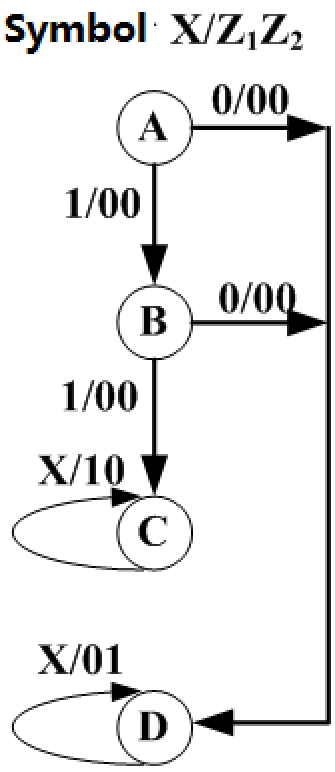


Figure 5.5 State chart

The state chart shows that the initial state is A. When input X is “1”, state transfers to B and C in sequence. Finally output Z_1 is “1”, and lock opens. In above process, once input X is “0”, state will transfer to D at once making output Z_2 is “0”, giving an alarm. In state C and D, state keep steady no matter what X is.

The comparison result of one of the two bits of presetting codes and its corresponding opening code determines the value “0” or “1” of X. If it is identical, X will be “1”, otherwise “0”. Clock is given out by single pulse.

3) Introduction to shift registers

Shift registers consist of arrangements of flip-flops and are important in applications involving the storage and transfer of data in a digital system. A register has no specified sequence of states, and is used solely for storing and shifting data entered into it from an external source and typically possesses no characteristic internal sequence of states.

The 74LS194 is an example of a universal bidirectional (the data can be shifted either left or right) shift register in integrated circuit form. A universal shift register has both serial and parallel input and output capability. Table 5-3 is the function table of 74LS194.

Table 5-3 74LS194 function table

Function	**Inputs**							**Outputs**
	Reset	**Mode**		**Clock Pulse**	**Serial**		**Parallel**	Q_A Q_B Q_C Q_D
		S0	**S1**		**L**	**R**	**A B C D**	
Reset	L	X	X	X	X	X	X X X X	L L L L
Keep	H	X	X	L	X	X	X X X X	Q_A Q_B Q_C Q_D
Send	H	H	H	↑	X	X	a b c d	a b c d
Shift	H	L	H	↑	X	H	X X X X	H Q_A Q_B Q_C
Right		L	H	↑	X	L	X X X X	L Q_A Q_B Q_C
Shift Left	H	H	L	↑	H	X	X X X X	Q_B Q_C Q_D H
		H	L	↑	L	X	X X X X	Q_B Q_C Q_D L
Keep	H	L	L	X	X	X	X X X X	Q_A Q_B Q_C Q_D

3. Preparation Requirements

1) Design a sequential digital lock with two bits. Write down your design process and draw logic circuit diagram, indicating pin numbers. Select what you need from given ICs.
2) Know logic function and applications of 74LS194. Make clear the work principle of figure 5.6 and figure 5.8. Draw timing logic diagrams, and study out steps.

4. Experiment Requirements

1) Two-bit sequential lock circuit
   (1) Build comparing circuit, checking its function.
   (2) Construct timing network, checking if state transition is correct.
   (3) Construct code circuit, checking if it can give out a bit of code with the

change of timing network state.

(4) Connect above circuits together, testing and debugging. Inputs and outputs should be connected to LEDS.

2) Realize 7-bit parallel in/serial out by 74LS194. Circuit diagram is as following:

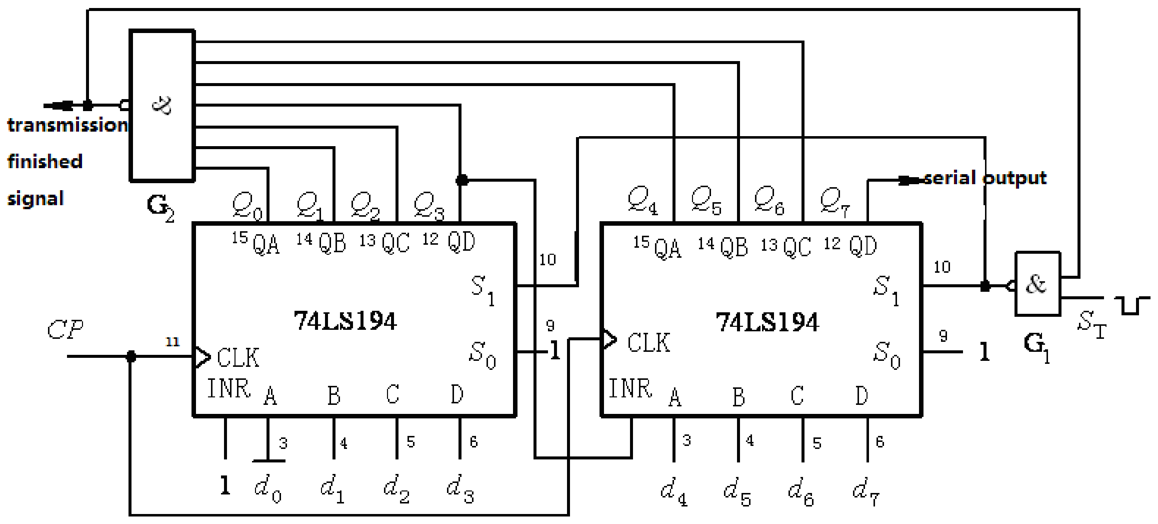


Figure 5-6 7-bit parallel in/serial out circuit diagram

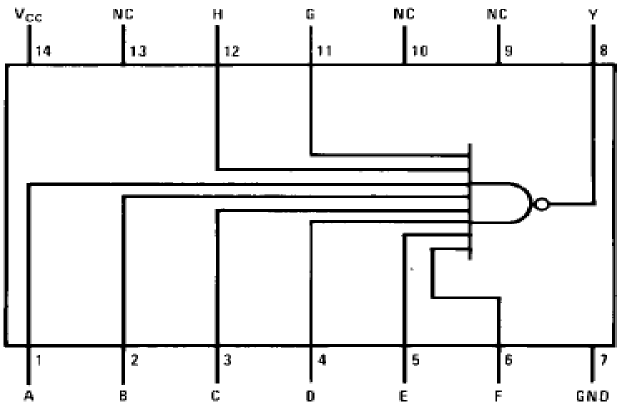


Figure 5-7 74LS30 pin out diagram

3) Realize serial in/7-bit parallel out by 74LS194. Circuit diagram is as following:

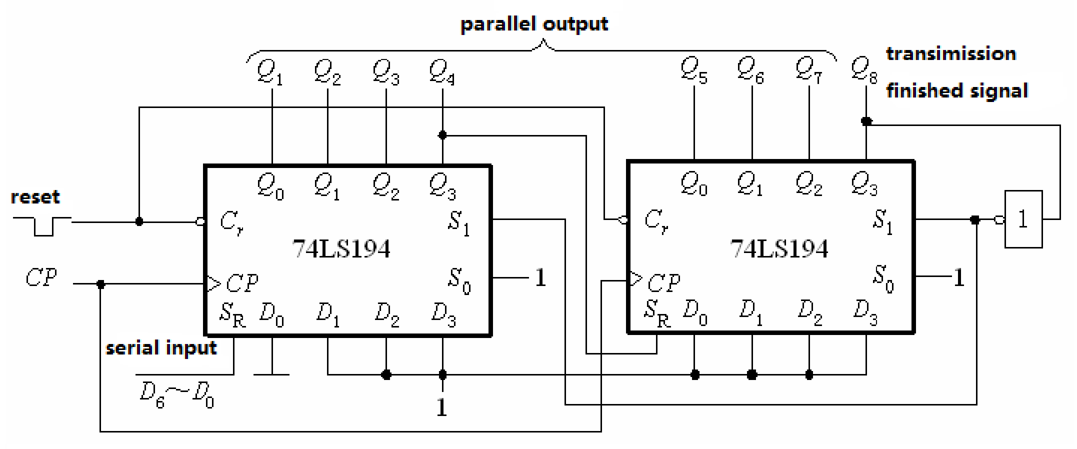
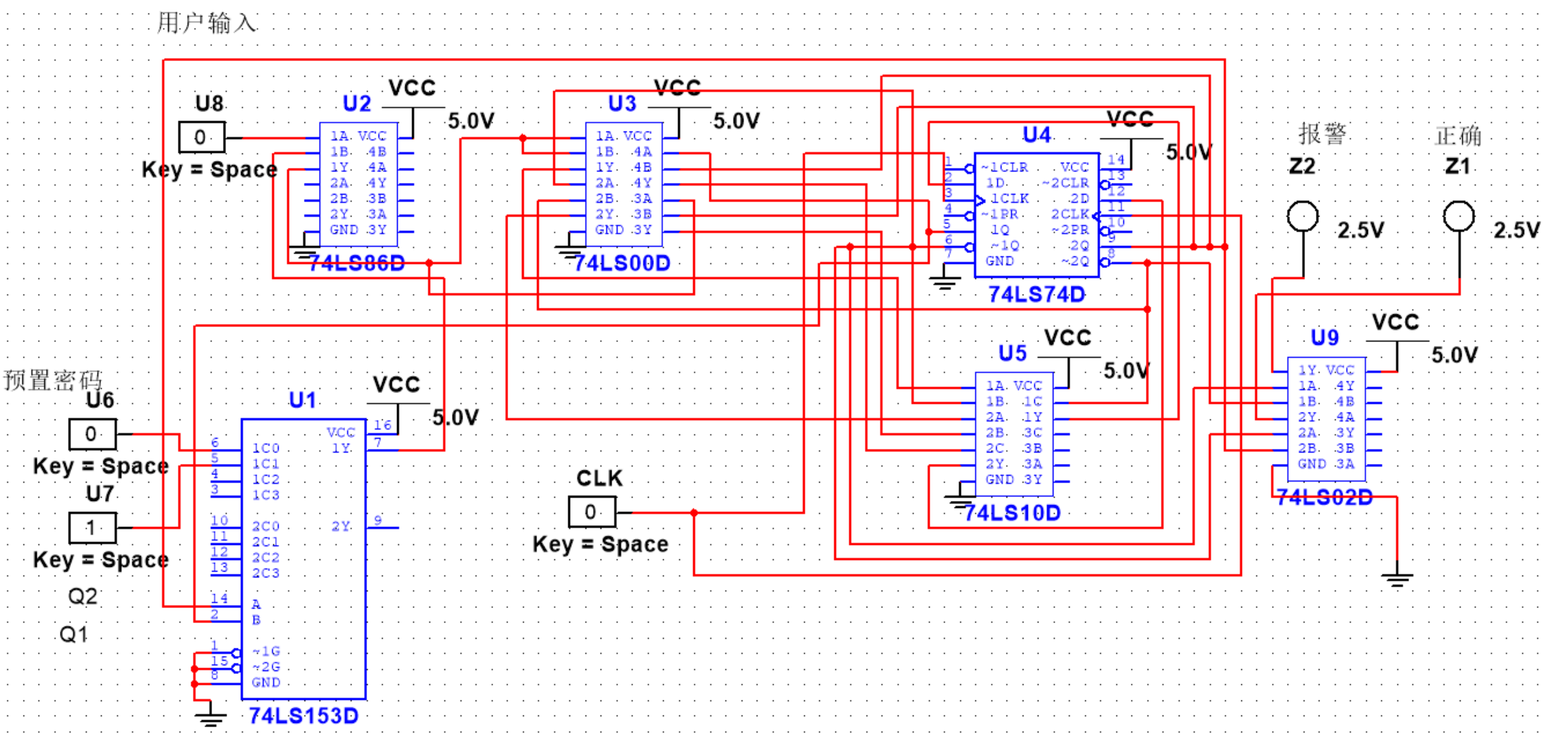

parallel output
Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 Q_8
transimission finished signal
reset
CP
74LS194
serial input
$D_6 \sim D_0$
1

Figure 5-8 serial in/7-bit parallel out circuit diagram

5. Circuit



Sequential digital lock circuit

6. Data record and analysis

1. Sequential digital lock.

①**Compare circuit**: if two input signals are the same (0 or 1), then the output signal X is high; if 2 input signals are different, the output signal X is low. From test result, compare circuit works well.

Input 1-Y	Input 2	Output X
0	0	1
0	1	0
1	0	0
1	1	1

②**Code select circui**t: in this part, IC should select code based on input signal A(14) B(2), and transfer signal Y to next part. I change preset password, connect Y to LED, and change signal input A(14) B(2), experiment data is record as follow

Code1	Code2	A	B	Y
0	1	0	0	0
0	1	1	0	1
0	0	0	0	0
0	0	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	1
1	1	1	0	1

③**Total circuit**:

After connecting whole circuit, I switched on the power, gave low level to reset the circuit and then changed the level to high. My input code was 1(IC1) 0(IC0). Then I gave low signal to pin1 of 74LS86, pressed the rebound switch to give IC a pulse, and nothing happened. Then I gave high signal to pin1, gave a pulse signal, then LED light of Z1 started to brighten, the lock opened! After that, no matter how I changed signal I input, the statement of two LED lights was stationary, without change.

Second time, I reset the circuit, and gave '0' signal first. Still, nothing happened. This time, I chose '0' as second signal to input the whole circuit. This time, light Z2 began to brighten, inferring I entered the wrong code. Then no matter how I changed input signal, Z2 brightened and Z1 not.

Third time, after resetting the circuit, I gave '1' signal to IC, and Z2 brightened. This told me I had entered a wrong number. Then the statement can't be changed.

In the end, I detected other code number, the circuit all worked well. I made it!

2) 7-bit parallel in/serial out by 74LS194

First, I made d1~d7 into 1,0,1,0,1,0,1, pressed reset button to clear the system. Then I made ST to low level, changing 74LS194 to SEND statement, and LED for transfer finished light, implied circuit haven't finished. Then I give IC a pulse, now Q0~Q7 equaled to D0~D7. Next, I changed ST to high lever, changing 74LS194 to SHIFT RIGHT statement. When I gave the circuit one pulse, data shifted right for 1 bit, so the output light would brighten in the order of "1,0,1,0,1,0,1,0". After I input 8 pulses, the "transferring" light didn't brighten, inferring all parallel data had serial outed.

Change numbers parallel in, output LED could show the data one by one. So, it worked successfully.

3) Serial in/7-bit parallel out by 74LS194

First, I reset the circuit. Then, I gave Pin-SR '0' signal, press rebound switch to make a pulse. After that, signal '0' was input and shown on pin Q0. Then I repeated operation above to input signal '0,1,0,1,0,1,0,1'. When I entered numbers, the numbers entered before shifted right at the same time.

In the end I tried many different number combinations, all could be performed correctly by this circuit. So, it worked successfully!

2. Testing experience and timing diagram

My testing experience is: test every part of circuit separately, and connect them together only after ensuring they works well. If there is some wired phenomenon, don’t be panic, miracle happens after thinking calmly.

Timing diagram of 7-bit parallel in/serial out has been drawn as follow: