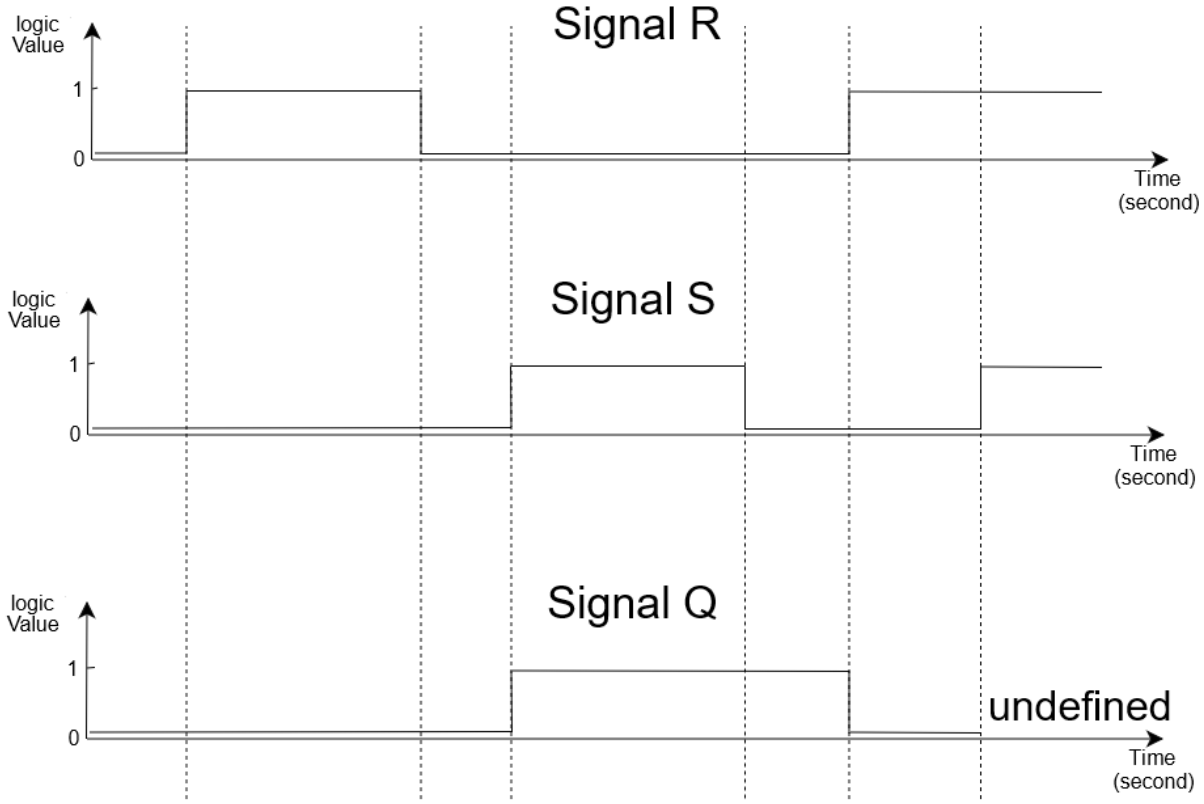
Sétif 1 University
Faculty of Sciences
Common Core Mathematics and Computer Science

2nd semester (year 2023/24)
Machine Structures 2 course
Kara Abdelaziz professor

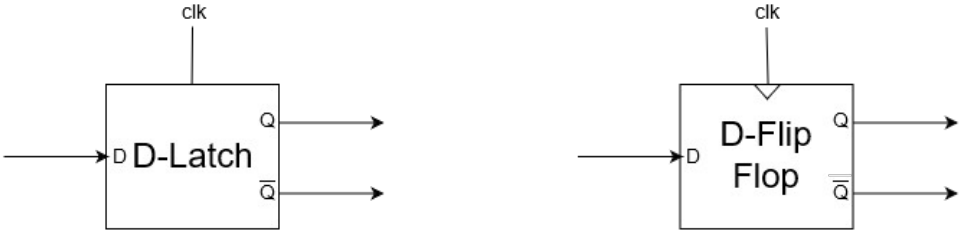
Problem set 2 solution
(Sequential Circuits)

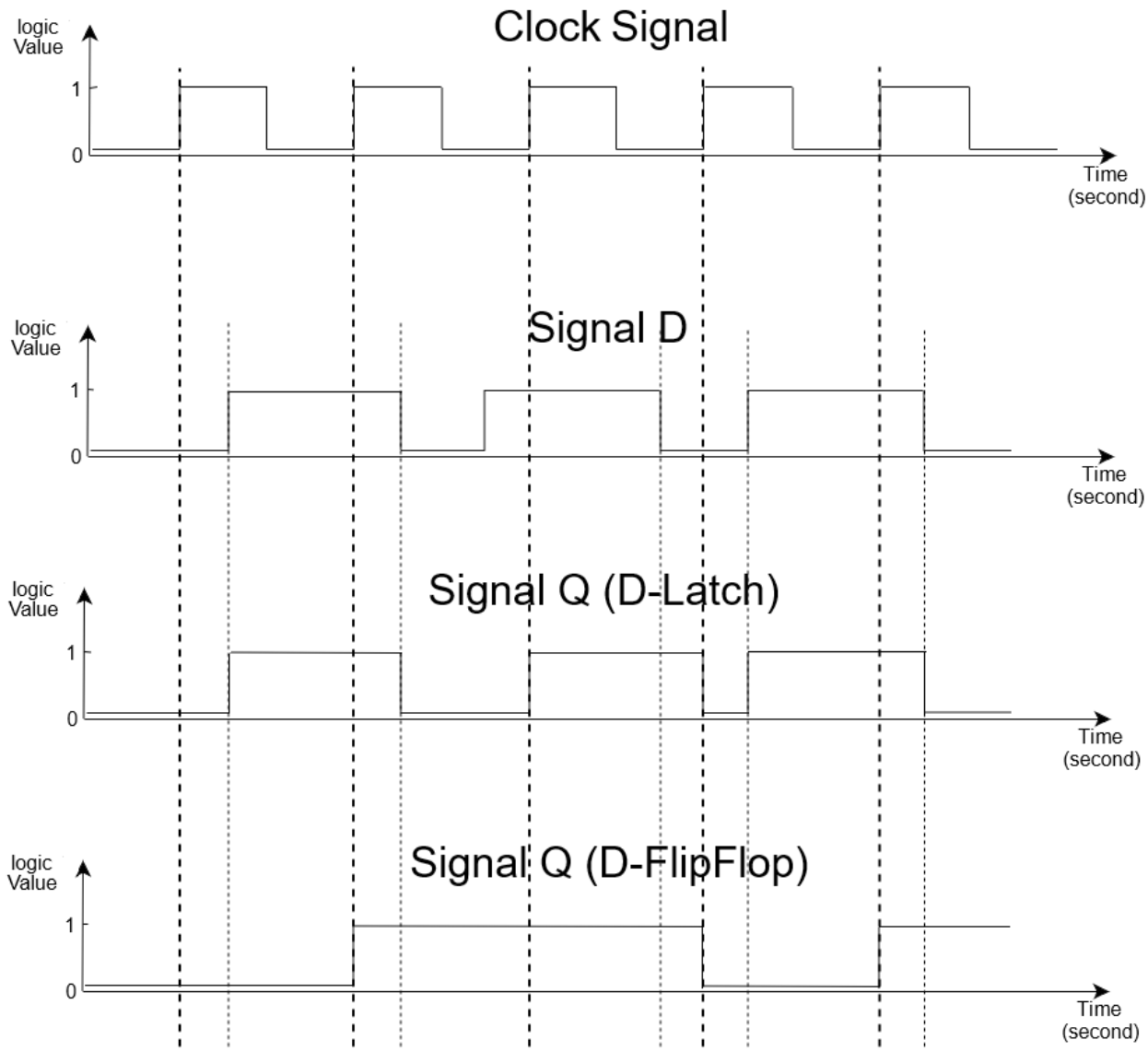
Exercise 01 :

1) The timing diagram of the RS-Latch is as follows :



2) The timing diagram of the D-Latch and the D-FlipFlop :

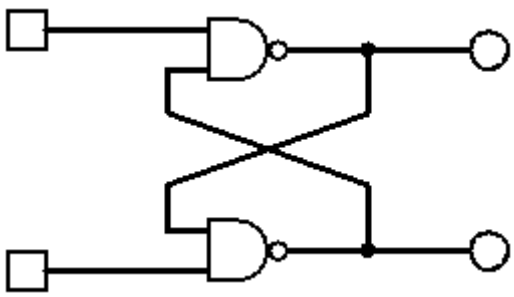




Exercise 02 :

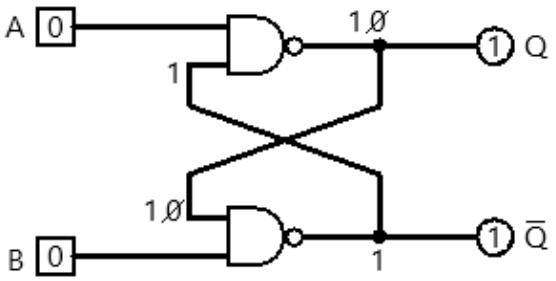
1)

1.

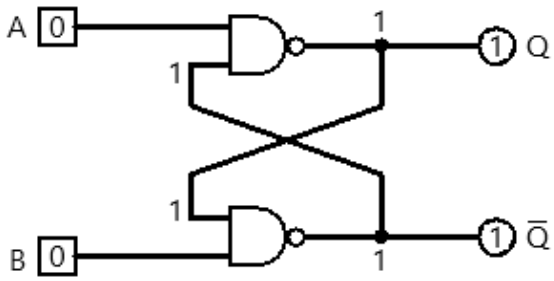


For A=0 and B=0

Initially Q=0

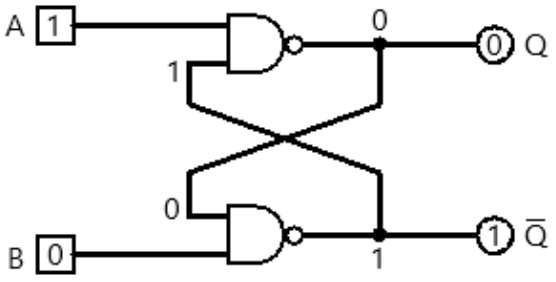


Initially Q=1

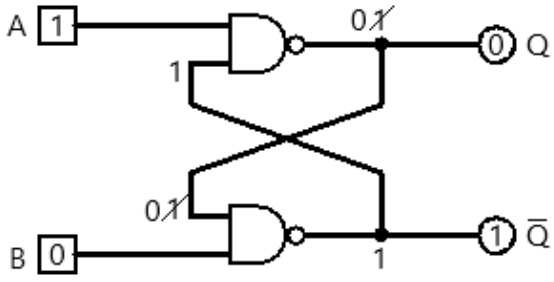


For A=1 and B=0

Initially Q=0

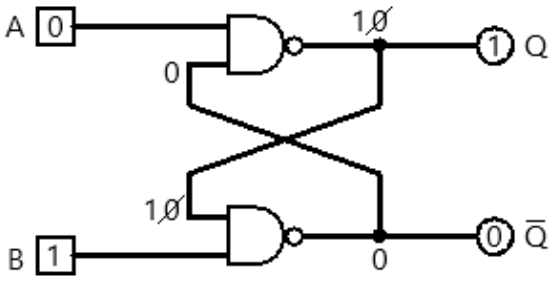


Initially Q=1

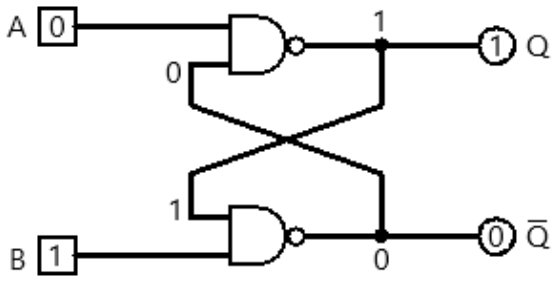


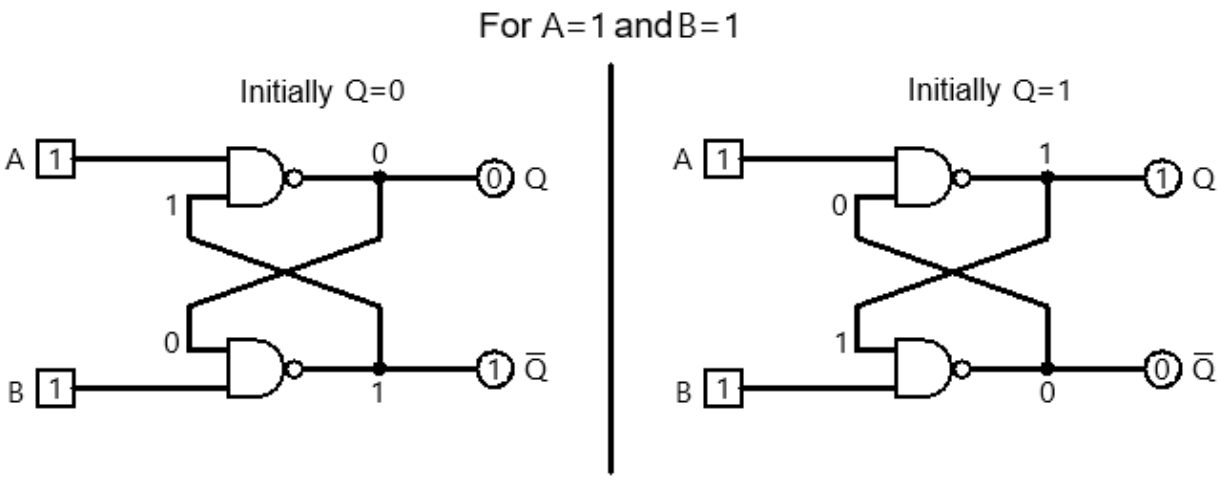
For A=0 and B=1

Initially Q=0



Initially Q=1

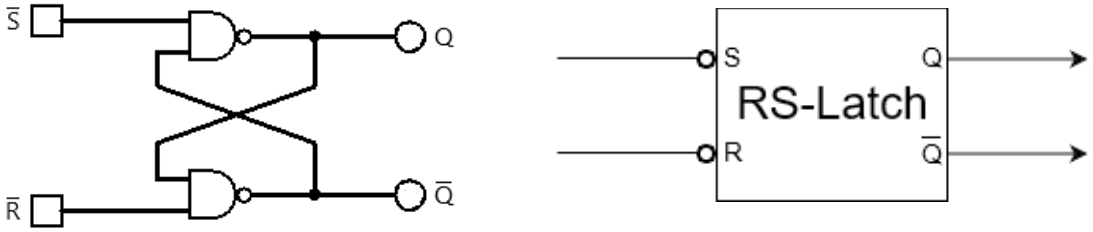




Then we would have a Truth Table :

A	B	Q	$\overline{Q}$	
0	0	1	1	Not defined
0	1	1	0	Set to 1
1	0	0	1	Reset to 0
1	1	Q`	$\overline{Q}$`	Memorize

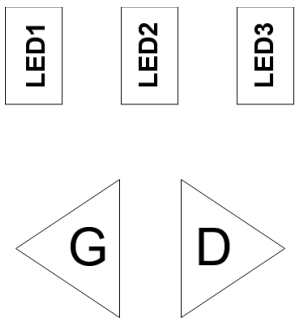
By observation, the truth table represents a RS-Latch circuit with A=$\overline{S}$ et B=$\overline{R}$



This is called a circuit $\overline{R}\,\overline{S}$-Latch

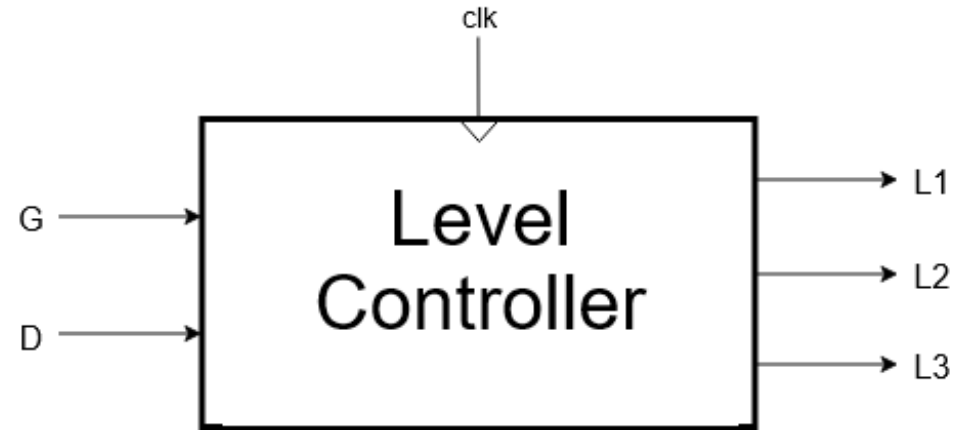
Remark : In modern circuits and for greater optimality, FlipFlops are implemented at the transistor level, so they are even smaller.

Exercise 03 :

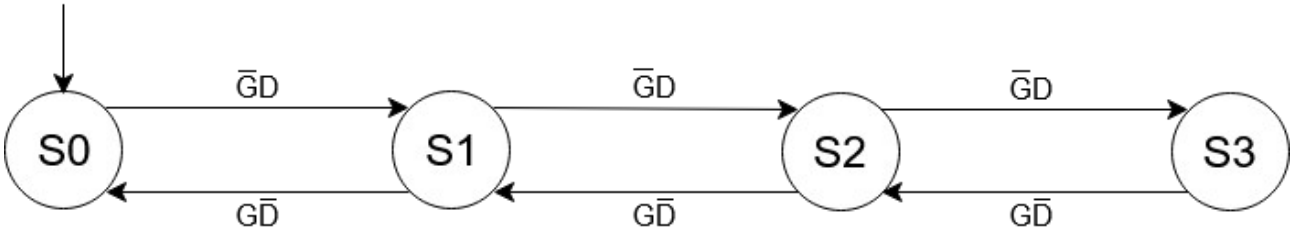


1)

Step 1 : Global Scheme



Step 2 : F.S.M.



S0 → All LEDs are off
S1 → LED1 is on
S2 → LED1 and LED2 are on
S3 → All LEDs are on

S0 → L1 = 0, L2 = 0, L3 = 0
S1 → L1 = 1, L2 = 0, L3 = 0
S2 → L1 = 1, L2 = 1, L3 = 0
S3 → L1 = 1, L2 = 1, L3 = 1

Remark 1: The outputs should normally be represented inside the State circle, they have been listed below the F.S.M. for simplicity.

Remark 2: Events are described by Boolean expressions instead of binary values ($\bar{G}D \iff G=0$, $D=1$). This allows them to be reduced in writing and brings together several combinations of values under the same expression.

Step 3 : Transition Table

Current State	G	D	Next State
S0	0	0	S0
	0	1	S1
	1	0	S0
	1	1	S0
S1	0	0	S1
	0	1	S2
	1	0	S0
	1	1	S1
S2	0	0	S2
	0	1	S3
	1	0	S1
	1	1	S2
S3	0	0	S3
	0	1	S3
	1	0	S2
	1	1	S3

Remark : Transition Table must contain all possible combinations of inputs.

Step 4 : Encoded States Table and Outputs Table

State	S_1	S_0	L1	L2	L3
S0	0	0	0	0	0
S1	0	1	1	0	0
S2	1	0	1	1	0
S3	1	1	1	1	1

Remark : The formulas of L1 and L2 and L3 can be extracted visually.

Step 5 : Encoded Transition Table

S_1	S_0	G	D	S'_1	S'_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	1	1

Step 6 : Logic Formulas

$L1 = S_1+S_0$
$L2 = S_1$
$L3 = S_1 \cdot S_0$

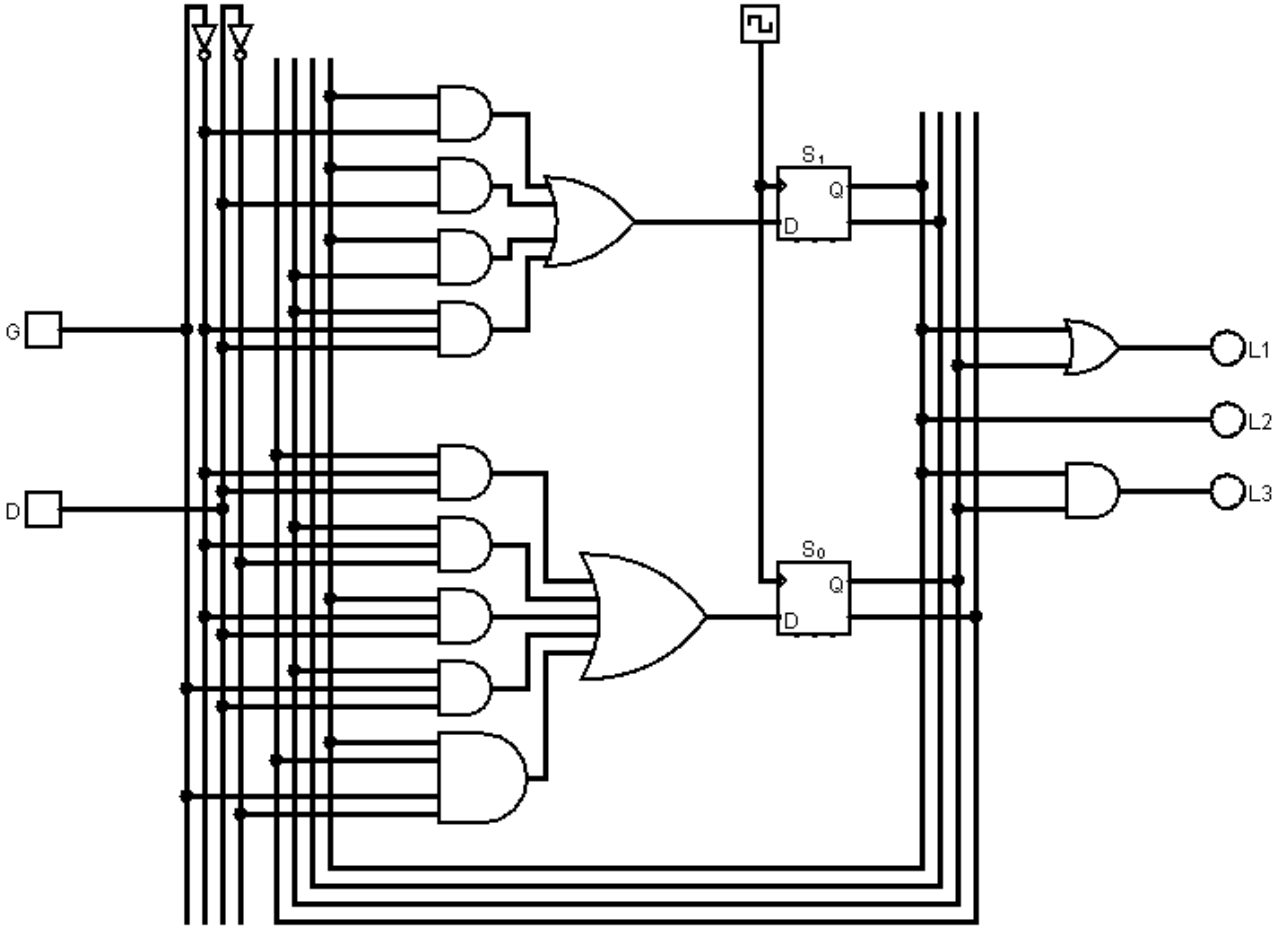
GD \ S_1S_0	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	0	0	1	1
10	0	0	1	0

$S'_1(S_1,S_0,G,D) = S_1 \cdot \overline{G} + S_1 \cdot D + S_1 \cdot S_0 + S_0 \cdot \overline{G} \cdot D$

GD \ S_1S_0	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	0	1	1	0
10	0	0	0	1

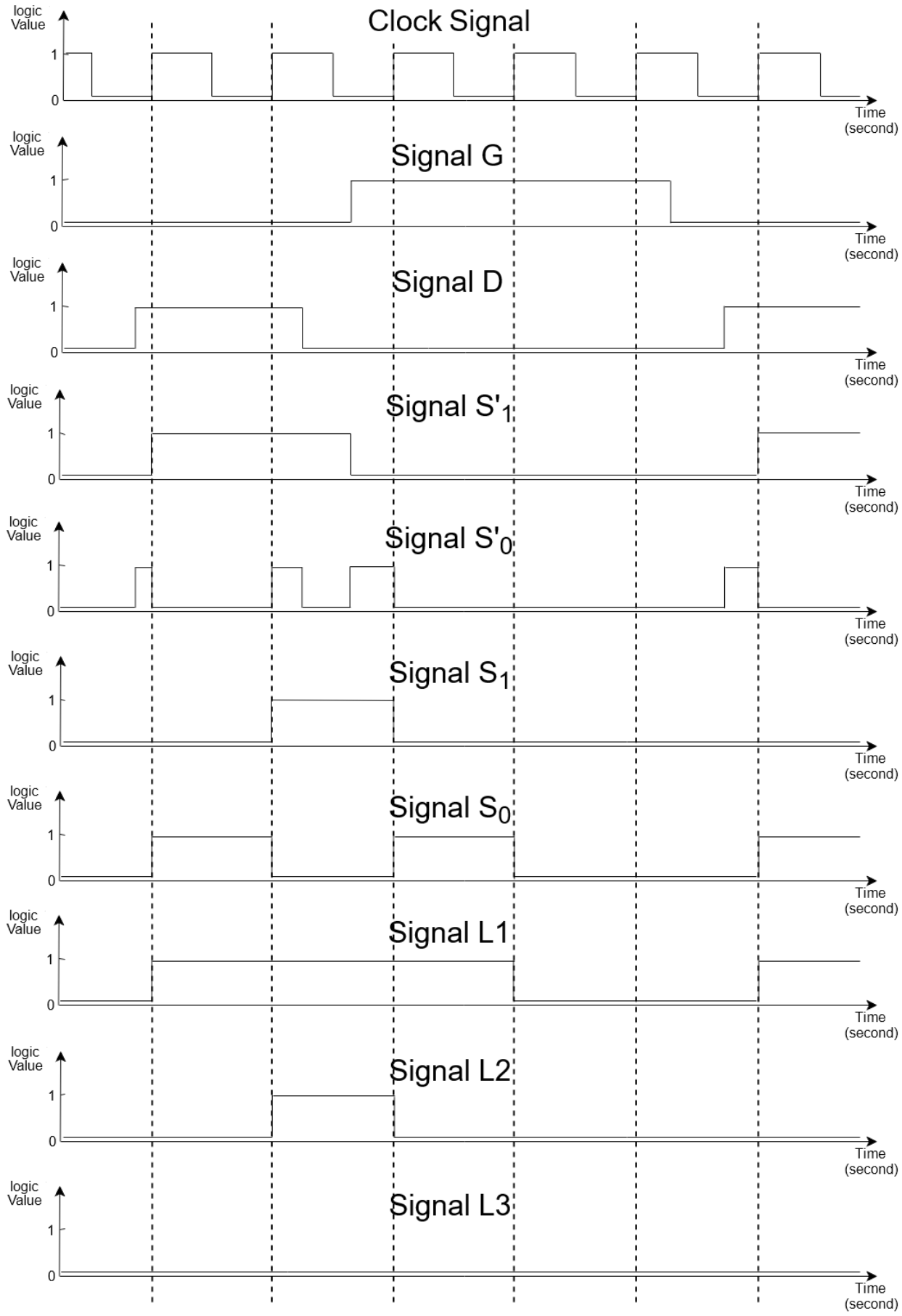
$S'_0(S_1,S_0,G,D) = \overline{S_0} \cdot \overline{G} \cdot D + S_0 \cdot \overline{G} \cdot \overline{D} + S_1 \cdot \overline{G} \cdot D + S_0 \cdot G \cdot D + S_1 \cdot \overline{S_0} \cdot G \cdot \overline{D}$

Step 7 : Schematics



2) When the user presses both buttons at the same time the State will not change, it is also equivalent to when the user does not press any button. This operation is not shown on the F.S.M. Usually, a Transition looping upon the same State must be drown to illustrate these 2 cases, but to not clutter up the F.S.M. too much they have been omitted.

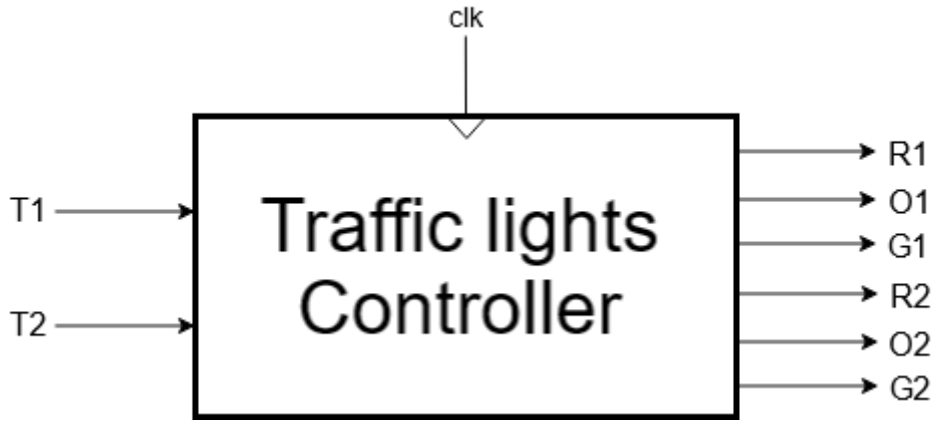
3)


Clock Signal
Signal G
Signal D
Signal S'1
Signal S'0
Signal S1
Signal S0
Signal L1
Signal L2
Signal L3
logic Value
1
0
Time (second)

Exercise 04 :

1)

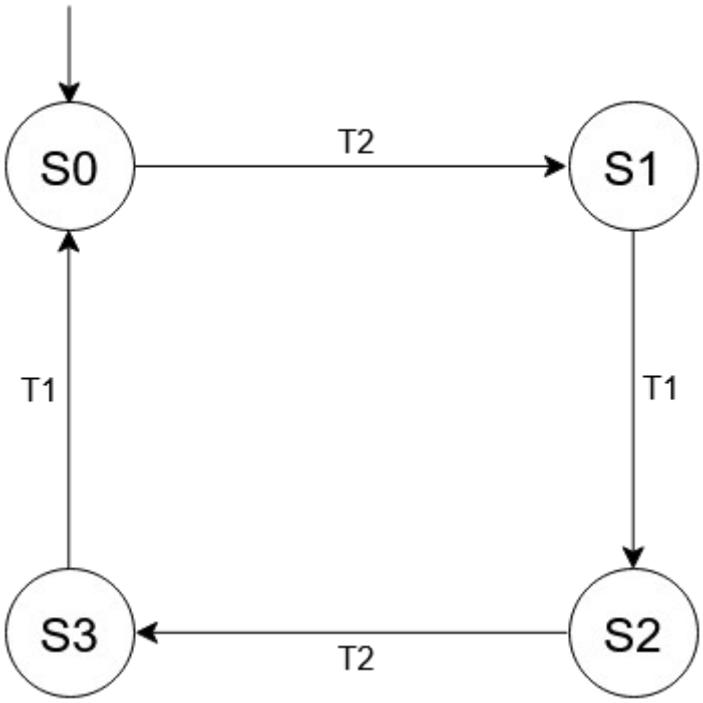
Step 1 : Global Scheme



T : Timer
R : Red color
O : Orange color
G : Green color

R1, O1, G1 : lights of F1 and F4
R2, O2, G2 : lights of F2 and F3

Step 2 : F.S.M.



S0 → R1=1, O1=0, G1=0 (F1 and F4 are red)
R2=0, O2=0, G2=1 (F2 and F3 are green)

S1 → R1=1, O1=0, G1=0 (F1 and F4 are red)
R2=0, O2=1, G2=0 (F2 and F3 are orange)

S2 → R1=0, O1=0, G1=1 (F1 and F4 are green)
R2=1, O2=0, G2=0 (F2 and F3 are red)

S3 → R1=0, O1=1, G1=0 (F1 and F4 are orange)
R2=1, O2=0, G2=0 (F2 and F3 are red)

Remark 1: Timer 1 will produce a 1 at the end of each 30 seconds, this will trigger the circuit to switch from red to green and vice versa. Timer 2 is also 30 seconds, it is shifted 27 seconds behind, or 3 seconds ahead (since it is a periodic signal) compared to Timer 1. It is responsible for the timing of the orange light.

Remark 2: The signals R1, O1 and G1 are identical for F1 and F4. And the same for R2, O2 and V2 with F2 and F3.

Step 3 : Transition Table

Current State	T1	T2	Next State
S0	0	0	S0
	0	1	S1
	1	0	-
	1	1	-
S1	0	0	S1
	0	1	-
	1	0	S2
	1	1	-
S2	0	0	S2
	0	1	S3
	1	0	-
	1	1	-
S3	0	0	S3
	0	1	-
	1	0	S0
	1	1	-

Remark 1: T1 and T2 are Timers and it is impossible according to the arrangement of our circuit for the signals T1 and T2 to arrive at the same time, or for T2 to exceed T1. These impossible cases are represented by many *don't care* in the table which will help us later in the minimization.

Remark 2: For simplicity, the loop Transitions for the case T1=0, T2=0 in all 4 States, were not represented on the F.S.M. But these Transitions are real and they must be taken into consideration in the Transition Table .

Remark 3: Often looped Transitions that do not change the State are ignored in the F.S.M. But they are always implmented in the Transition Table.

<u>Step 4</u> : Encoded States Table and Outputs Table

States	S_1	S_0	R1	O1	V1	R2	O2	V2
S0	0	0	1	0	0	0	0	1
S1	0	1	1	0	0	0	1	0
S2	1	0	0	0	1	1	0	0
S3	1	1	0	1	0	1	0	0

<u>Step 5</u> : Encoded Transition Table

S_1	S_0	T1	T2	S'_1	S'_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	-	-
0	0	1	1	-	-
0	1	0	0	0	1
0	1	0	1	-	-
0	1	1	0	1	0
0	1	1	1	-	-
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	-	-
1	0	1	1	-	-
1	1	0	0	1	1
1	1	0	1	-	-
1	1	1	0	0	0
1	1	1	1	-	-

Step 6 : Logical Formulas

$R1(S_1,S_0) = \overline{S_1}$
$O1(S_1,S_0) = S_1 \cdot S_0$
$V1(S_1,S_0) = S_1 \cdot \overline{S_0}$

$R2(S_1,S_0) = S_1$
$O2(S_1,S_0) = \overline{S_1} \cdot S_0$
$V2(S_1,S_0) = \overline{S_1 + S_0}$

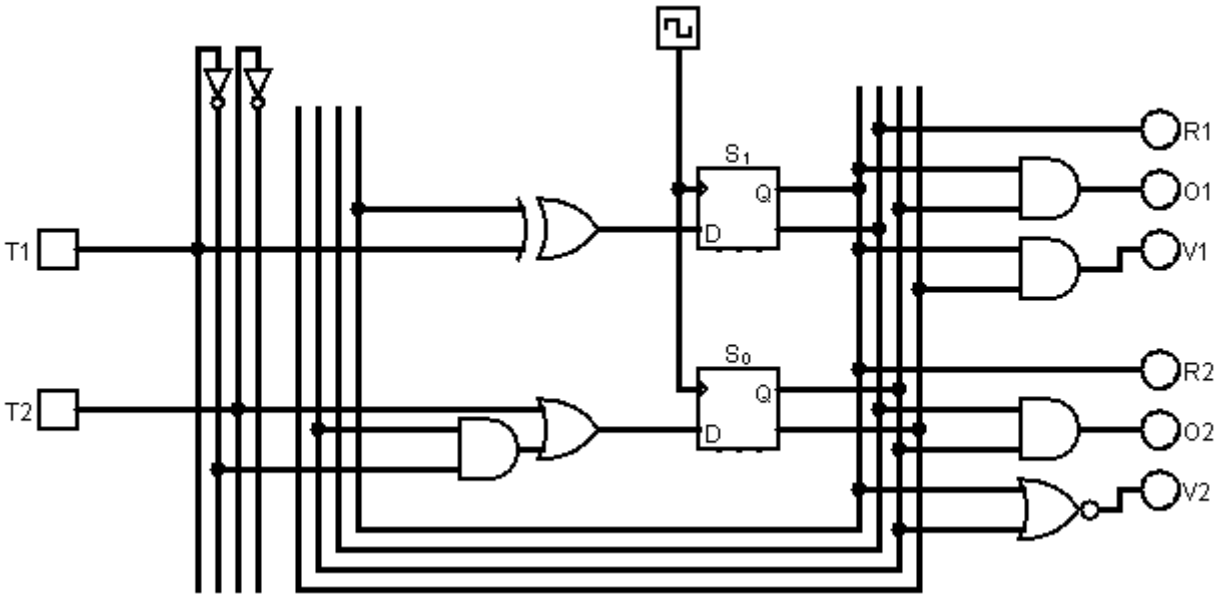
S_1S_0 / T1T2	00	01	11	10
00	0	0	1	1
01	0	-	-	1
11	-	-	-	-
10	-	1	0	-

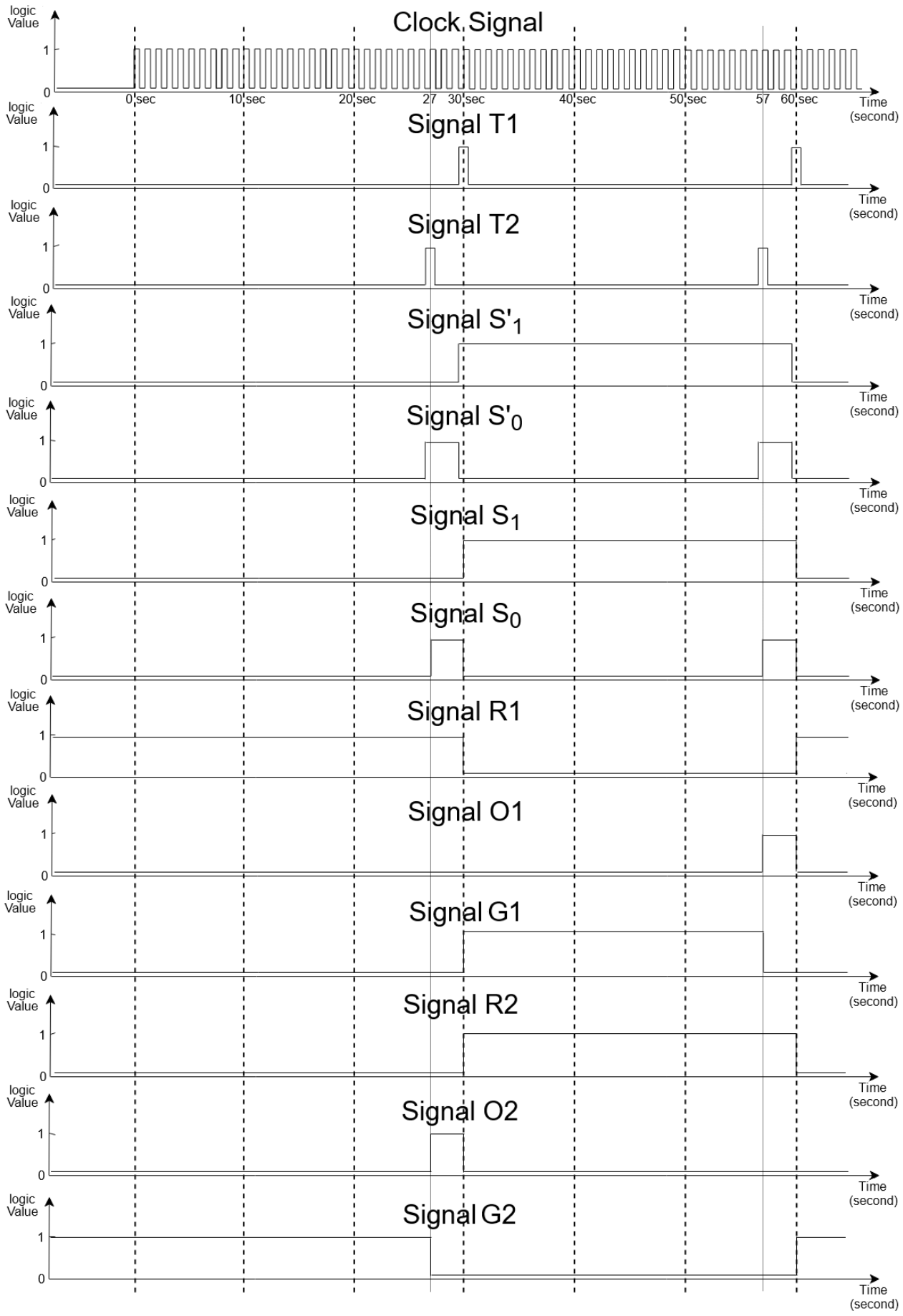
S_1S_0 / T1T2	00	01	11	10
00	0	1	1	0
01	1	-	-	1
11	-	-	-	-
10	-	0	0	-

$S'_1(S_1,S_0,T1,T2) = S_1 \cdot \overline{T1} + \overline{S_1} \cdot T1 = S_1 \oplus T1$

$S'_0(S_1,S_0,T1,T2) = T2 + S_0 \cdot \overline{T1}$

Step 7 : Schematics




Clock Signal
Signal T1
Signal T2
Signal S'1
Signal S'0
Signal S1
Signal S0
Signal R1
Signal O1
Signal G1
Signal R2
Signal O2
Signal G2
logic Value
1
0
0 sec
10 sec
20 sec
27
30 sec
40 sec
50 sec
57
60 sec
Time (second)

Remark 1: The frequency of the clock was not indicated in the exercise, but it can be deduced that the period cannot exceed 3 seconds, because the minimum signal change in the circuit is 3 seconds, that of the orange light. We chose 1 second as the period, therefore a frequency of 1 Hertz.

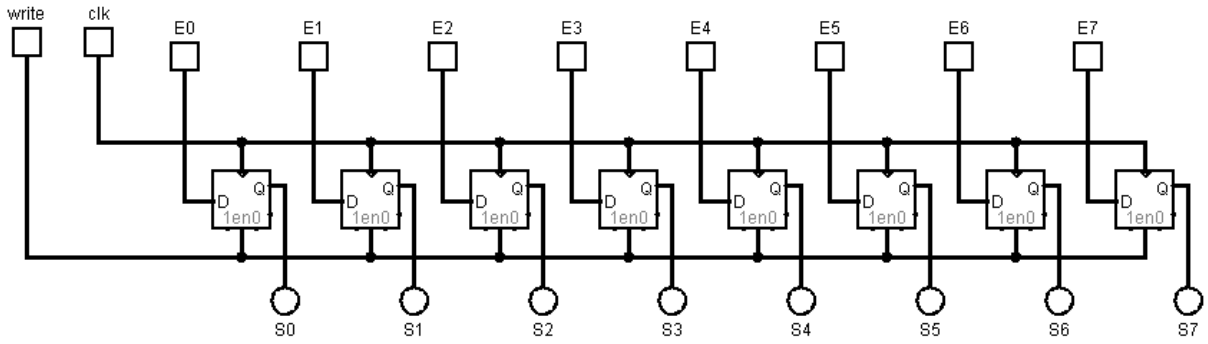
Remark 2: The signals from the 2 Timers are shifted and advanced by half a period compared to the clock signal, this gives enough time for the D-FlipFlop and the signal in the combinational circuits to propagate correctly.

Remark 3: Tracking the signals from the memory cells and the Output Combinational Output Circuit are quite easy, they are fixed over an entire period. On the other hand, the signals S'_1 and S'_0 of the Next State Combinational Circuit are more difficult to follow, because they must be updated with each change in the inputs T1 and T2 and in the memory cells S_1 and S_0.

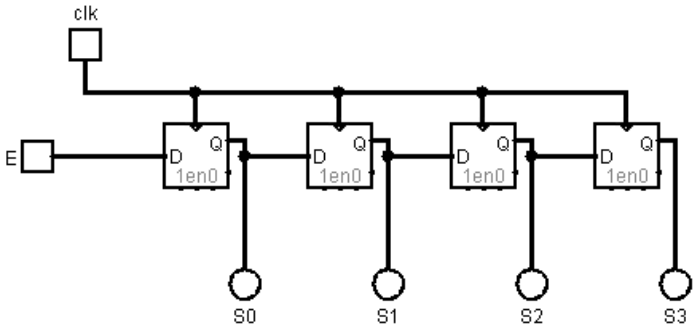
Remark 4: To avoid too many complications in the chronogram, the signals S'_1 and S'_0 were not updated in the small part of the half-period after T1 and T2.

Exercise 05 :

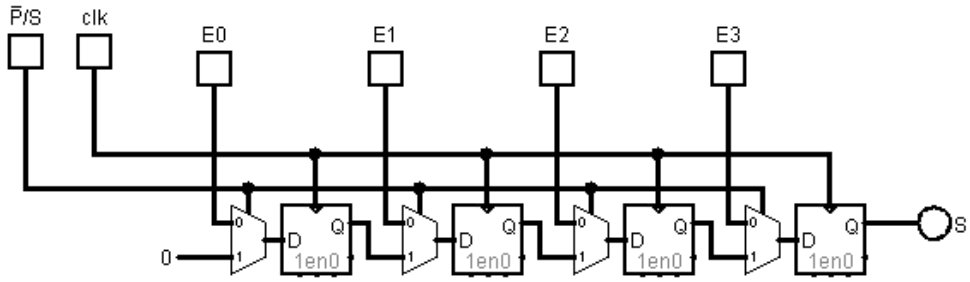
1) The 8-bit Register circuit with the Write command :



2) The 4-bit SIPO Shift circuit :



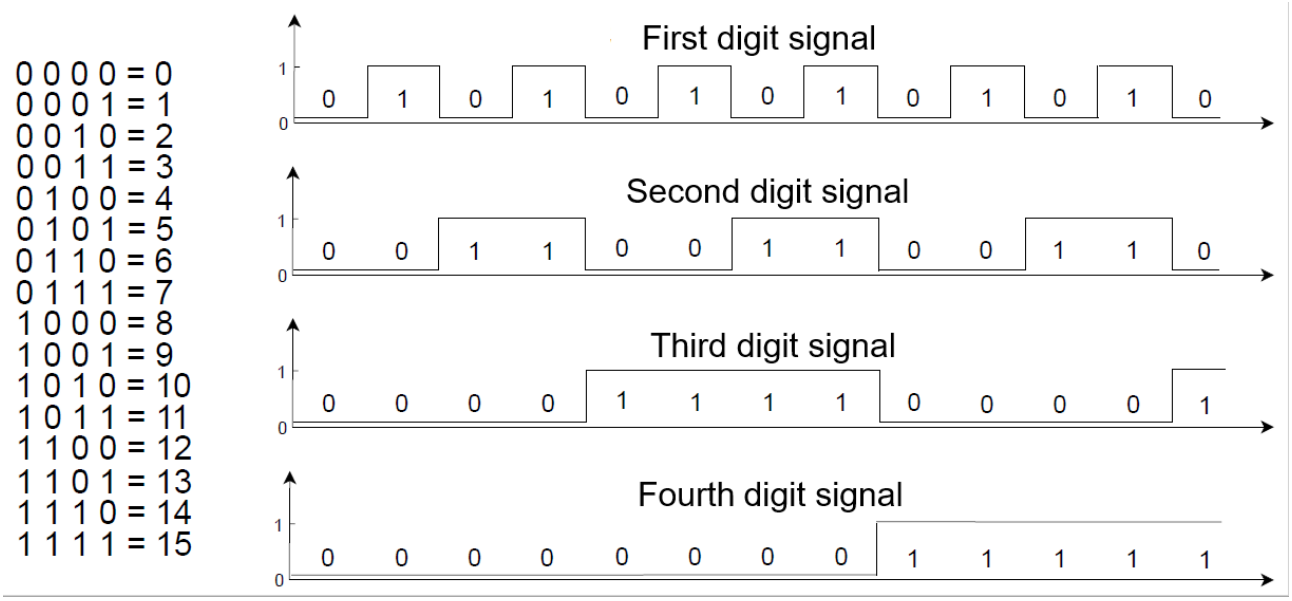
The 4-bit PISO Shift circuit :



Remark 1: The $\overline{P}$/S ($\overline{\text{Parallel}}$/Serial) control input allows to choose between, entering the 4 inputs in parallel, or shifting the values in a chain through the cells to output them in series on the S output.

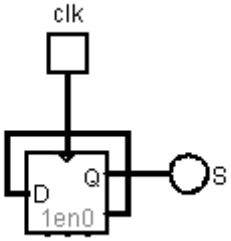
Remark 2: The bar in $\overline{P}$/S indicates which functionality is chosen by the value 0, so if the input is set to 0 the Parallel input will be used.

3) To build a Counter we must carefully study and observe the binary format of the increment operation. We can see the numbers incrementing in the left part of the figure below :

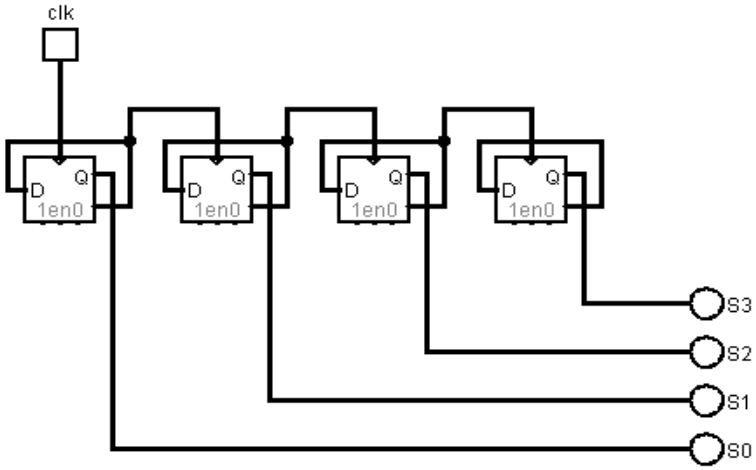


If we focus on a single column for a given digit, for example on the first digit. We can observe that it is a periodic signal identical to the clock signal. As well as for the second column, with the difference that it is double the first. And the third is also double the second, and so on and so forth. The signals of all its 4-bit digits are transcribed in the right part of the above figure.

So to create a Counter, we simply need to generate a clock signal for each digit, with the condition that the next digit must double the period of the previous digit. To create a clock signal it's quite simple, just use a D-FlipFlop as shown in the diagram below. The $\overline{Q}$ output must loop to the D input, so for each Rising Edge the FlipFlop will take the inverse value of the current value.

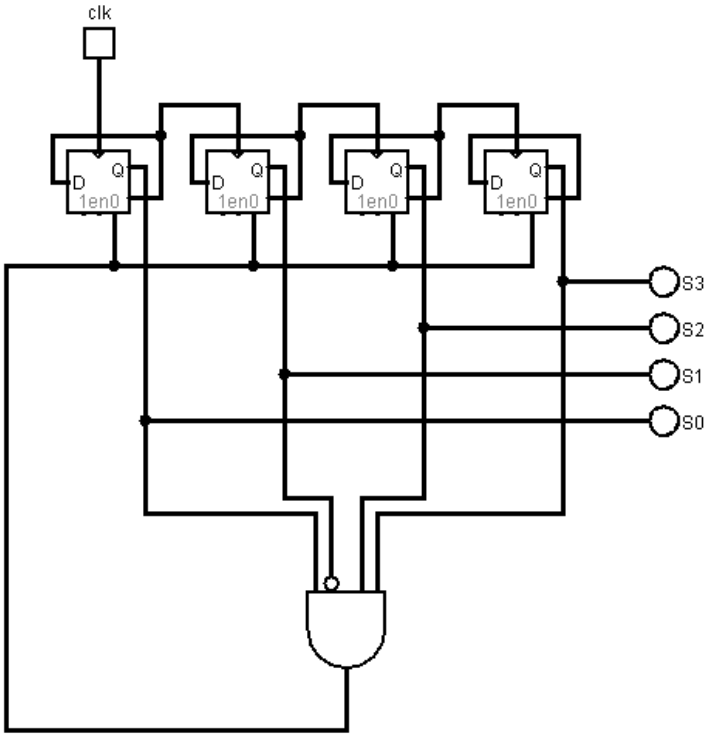


To make the signal of digit taking twice the period of the previous digit, we simply use the output of the previous digit as a clock for the current digit. As in the figure below. You note that in the figure we take the signal $\overline{Q}$ instead of Q. Because by observing previous graph of the signals above, we notice that the digital signal really changes value during the Falling Edge of the previous digital signal. This forces us to use the inverse signal $\overline{Q}$ instead of Q.



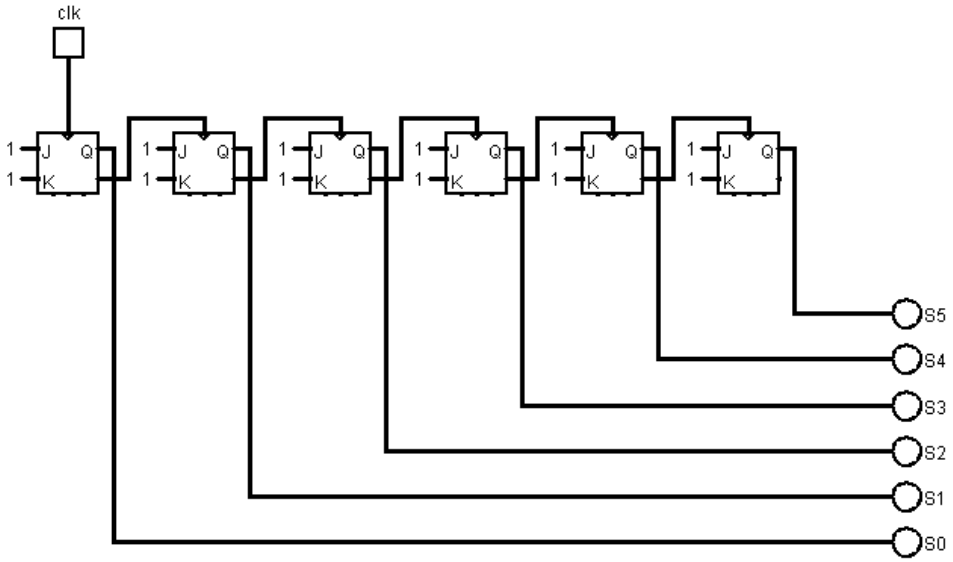
To limit the count to the value 12 (1100), it is wise to use the FlipFlip Reset command which will reset all the FlipFlops to 0 when the circuit detects that it has reached the limit. The limit in our case should be the value 13, the value just after 12, which should simply be detected by an AND gate. The Reset used here is Asynchronous, so it will reset the cell immediately after detecting the value 13. It practically instantaneous, the counter will never get enough time to hold the value 13.

The diagram of a 4-bit counter that counts up to the value 12 :

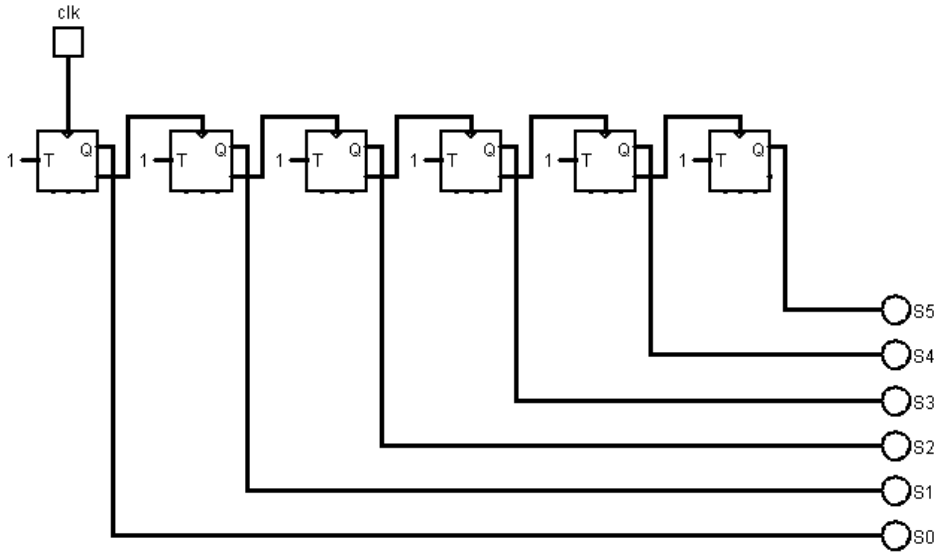


Remark 3: The Reset in this solution is supposed to be Asynchronous, if it was a Synchronous command the solution should be different.

4) The 6-bit counter circuit based on JK-FlipFlop :



The 6-bit counter circuit based on T-FlipFlop :



Remark 4: The Counter has the ability to oscillate its outputs by doubling the period for each output, which gives it the ability to be used very often as a frequency divider. For example, if you apply a frequency of 1 KHz to its clock, the first output will produce a signal of 500 Hz, the second of 250 Hz, the third of 125 Hz, and so on.

Remark 5: The 3 fairly simple types of Counters that we have just built are most often called Asynchronous Counters, or Ripple Counters. Due to the fact that the output of one cell is the clock of the next, which will cause a small clock delay that accumulate across the cells. For low frequency circuits below 1 MHz, normally this should not pose a problem. But for high frequency circuits they become a source of failure, then so-called Synchronous Counters which are more complex are generally more reliable. Synchronous implies that all cells use the same clock signal.