

**EXPERIMENT 5****COMBINATIONAL and SEQUENTIAL LOGIC CIRCUITS****Hardware implementation and software design****I. PURPOSE:**

To familiarize with combinational and sequential logic circuits. Combinational circuits are logic circuits whose outputs respond immediately to the inputs; there is no memory. In sequential logic circuits the outputs depend on the inputs plus its history; i.e. it has memory.

**In the experimental Section-1** you will build an adder (using NAND and NOR gates), as an example of **combinational** logic circuit.

**In the experimental Section-2**, sequential logic circuits are introduced through the construction of a **RS latch** (using NAND gates), which will help us to get an understanding about how **memory** is developed in logic circuits. Stability in the RS latch is obtained by implementing a series of gate controls, all of which lead to the development of the JK flip flop. Commercially available **JK flip flops** will be used to construct an hexadecimal and a decimal ring counter.

[Time permitted, and in order to gain a broad view on the subject, Section-2 may include a demonstrative session on how to use LabView software to design a JK flip flop. The underlying interest in the demonstration is to make students aware of a current technological trend of software designing circuit layouts. After compilation, the design is used to configure ('wire') a programmable logic semiconductor device (FPGA card), the latter subsequently acting as an independent hardware device. <http://en.wikipedia.org/wiki/Fpga>]

To gain hands on experience on the software design, you will be required to LabView design a 3-to-8 decoder using combinational logic circuits.

**II. THEORETICAL CONSIDERATIONS****II.1 How information is coded in electronic digital form?****II.1A Digital levels: transistor switch****II.1B Counting objects: Decimal and binary system****II.1C Digital electronics****II.1A Digital levels**

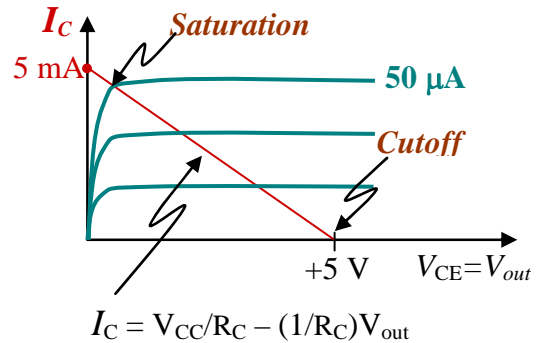
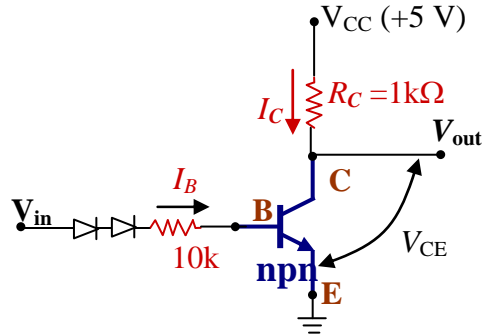
Consider the transistor switch

If  $V_{in} < 2.1$  Volts

- The BE diode would be reversed biased, therefore there will be no flow of electrons from E to B. That is, the transistor is OFF.
- No  $I_B$  current, no collector current. It implies  $V_{out} = V_{CC} = 5$  volts (**Digital 1**)

If  $V_{in} > 2.1$  .

- As  $V_{in}$  increases, the transistor moves out from cutoff along the loading line.
- Further increase of  $V_{in}$  makes the transistor reach the saturation stage,  $I_C = 5 \text{ mA}$ . For a transistor of  $\beta = 100$ , an  $I_B = 50 \mu\text{A}$  will saturate the transistor. Accordingly, a  $V_{in} = 3(0.7) + (10\text{k}\Omega)(50 \mu\text{A}) = 2.6 \text{ V}$  will saturate the transistor.
- From  $2.1 < V_{in} < 2.6$  the transistor works in the active region.

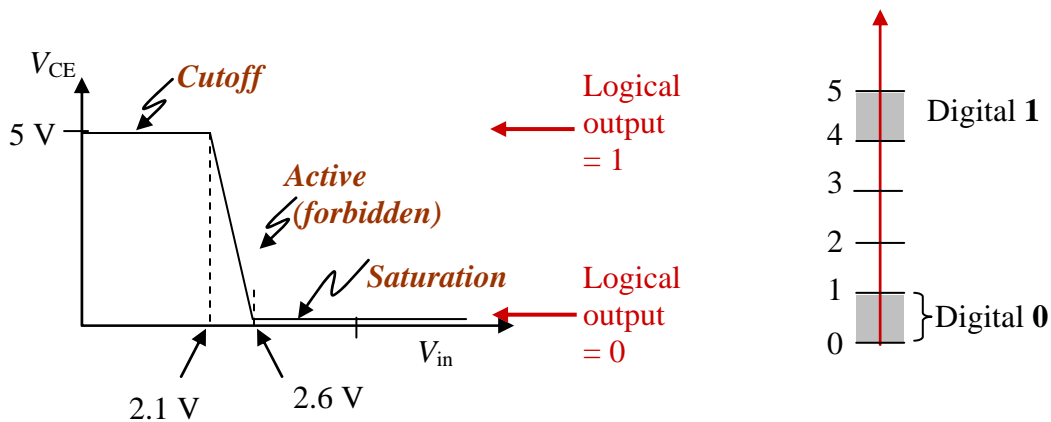


If  $V_{in} = 2.6 \text{ Volts}$

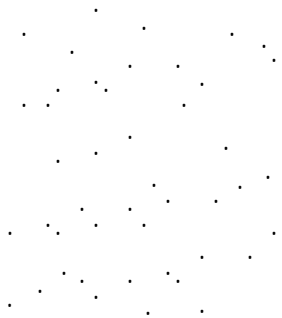
- The transistor is saturated,  $I_C = 5 \text{ mA}$ . The voltage drop across  $R_E$  is then 5 Volts, which make  $V_{out} = 0 \text{ Volts}$ .

If  $V_{in} > 2.6$ .

- The transistor remains saturated.  $V_{out} = 0 \text{ Volts}$  (**Digital 0**)

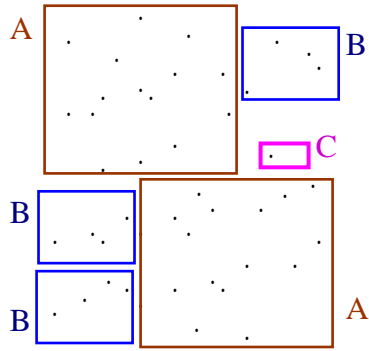


## II.1B Decimal and binary system



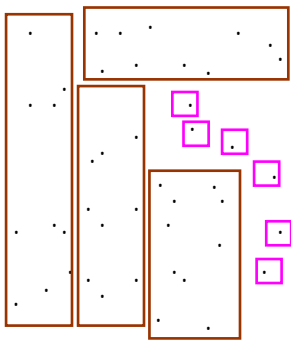
How to systematically count the elements of this system?

Using an arbitrary numerical system



2 groups of "type" A  
 3 groups of "type" B  
 1 group of "type" C  
  
 (2A) (3B) (5C)

Using the decimal system



4 groups of 10  
 6 groups of 1

$$4 (10^1) \quad 6 (10^0)$$

Then, as we assume that the **decimal system** is being used, we just write:

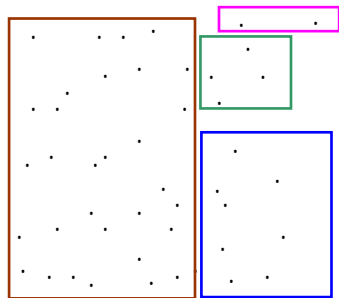
**4 6**

Array of decimal digits

The position of a digit gives the increasing powers of 10 in the number

Binary system

We will count them in sub-groups of  $2^0, 2^1, 2^2, 2^3, \dots$



1 group of  $2^5$   
 0 group of  $2^4$   
 1 group of  $2^3$   
 1 group of  $2^2$   
 1 group of  $2^1$   
 0 group of  $2^0$

$$1 (2^5) \quad 0 (2^4) \quad 1 (2^3) \quad 1 (2^2) \quad 1 (2^1) \quad 0 (2^0)$$

Then, as we assume that the **binary system** is being used, we just write: **1 0 1 1 1 0**

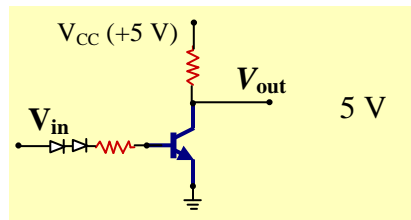
Array of binary digits

The position of a digit gives the increasing powers of 2 in the number

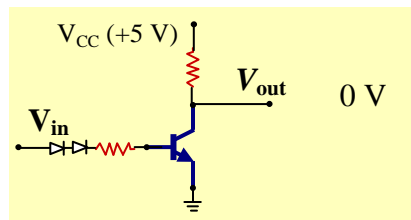
## II.1C Digital electronics

Using an array of transistor circuits

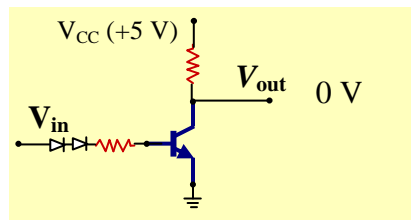
*Interpreted as logic levels*



**1**



**0**



**1**

## III. EXPERIMENTAL CONSIDERATIONS

### III.1 Combinational Logic Circuits

#### III.1A Logic gates

#### III.1B Digital Arithmetic: Adder circuit

### III.2 Sequential Logic Circuits

#### III.2.1 How memory is developed in logic circuits: SR LATCH.

#### III.2.2 Adding control to the SR LATCH: the JK FLIP=FLOP

#### III.2.3 Commercial JK FLIP=FLOP

#### III.2.3 JK Flip-flop applications

### III.3 LabView Design of a Decoder

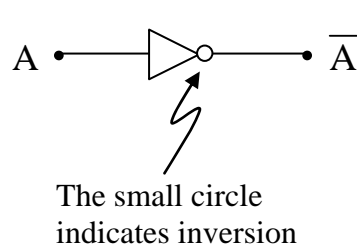
### III.1 COMBINATIONAL LOGIC CIRCUITS

Combinational circuits are logic circuits whose outputs respond immediately to the inputs; there is no memory.

#### III.1A Digital logic gates

Combinational Digital gates are circuits that pass or block signals moving through a logic circuit.

##### NOT gate



Input	Output
A	$\bar{A}$
0	1
1	0

##### OR gate

##### NOR gate

##### EXCLUSIVE OR gate



Inputs		Output
A	B	$A \oplus B$
1	1	0
1	0	1
0	1	1
0	0	0

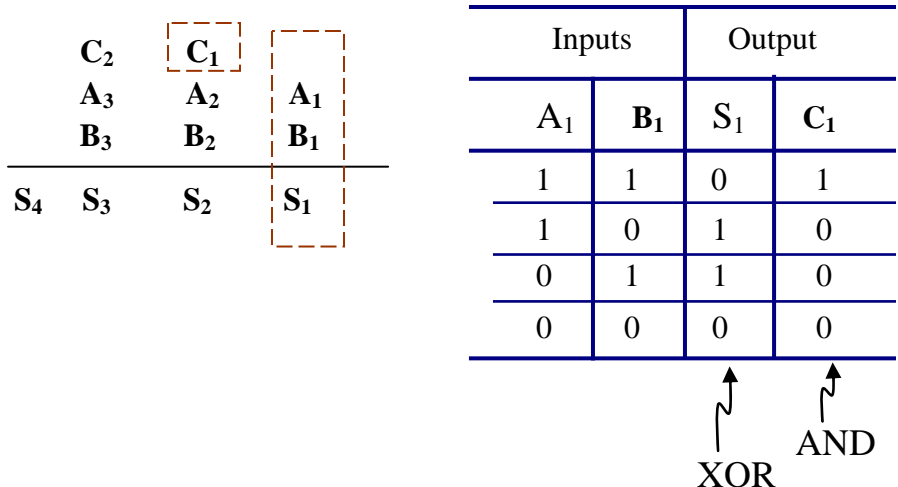
#### III.1B Digital Arithmetic: Adder circuit

The diagram on the left (figure below) indicates an addition operation of two binary numbers:  $A_3 A_2 A_1$  and  $B_3 B_2 B_1$ .

**Task:** To build a simple **half-adder** for adding  $A_1$  and  $B_1$ , as well as the carrier of their sum.

Optionally, you may want to implement a **full adder** for adding  $A_2$ ,  $B_2$ , and the previous carrier  $C_1$ , as well as to produce the forward carrier  $C_2$ .

##### HALF ADDER



The diagram above suggests that all we need is a XOR and AND gates. Since we have available only NAND and NOR gates, a bit a Boolean algebra comes timely to the rescue.

**Design of a XOR out of NAND and NOR gates**

- First, verify explicitly the following properties:

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

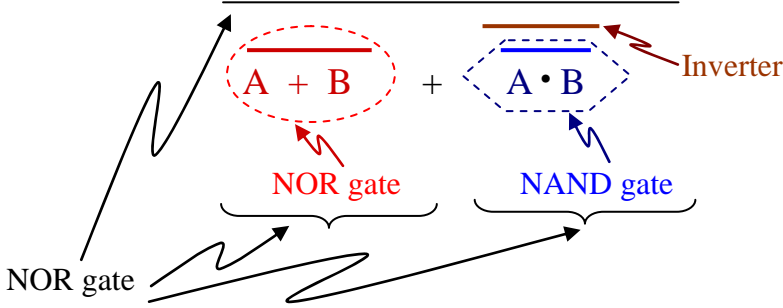
$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

- Verify  $A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$

as well as  $A \oplus B = (A + B)\overline{A \cdot B}$

- Hence,

$$\begin{aligned}
 A \oplus B &= (A + B)\overline{A \cdot B} = \\
 &= \overline{\overline{(A + B)\overline{A \cdot B}}} = \overline{\overline{(A + B)} \cdot \overline{\overline{A \cdot B}}} \\
 &= \overline{\overline{A + B} + \overline{A \cdot B}}
 \end{aligned}$$



**Fig.** XOR design with NAND and NOR gates

- Hence the following implementation

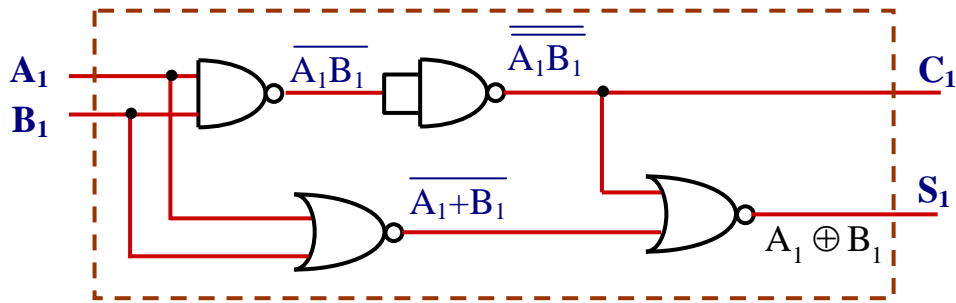


Fig. Half adder circuit

**Optional task:** Build a full adder (it adds two digits plus a previous carrier.).

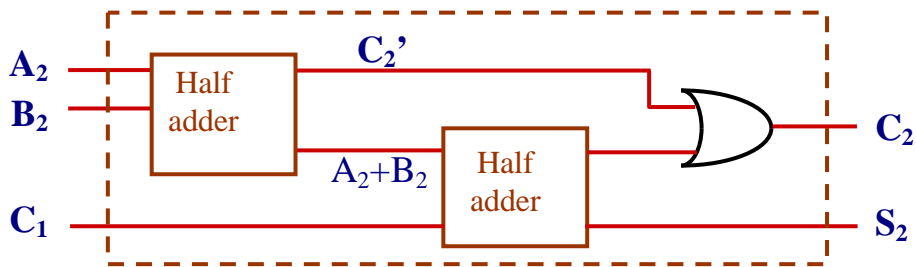


Fig. Full adder circuit

## III.2 SEQUENTIAL LOGIC CIRCUITS

**III.2.1 How memory is developed in logic circuits: SR LATCH.**

**III.2.2 Adding control to the SR LATCH: the JK FLIP=FLOP**

**III.2.3 Commercial JK FLIP=FLOP**

**III.2.3 JK Flip-flop applications**

Logic circuits, like the adder circuit, are called combinational logic circuits. Their characteristics are:

- The output responds immediately to the inputs
- There is no memory

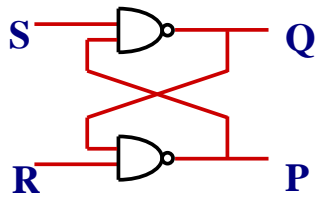
In contrast, in a sequential logic circuit

- The output not only depend on the inputs, but also on the inputs history
- That is, a sequential logic circuit has a memory

Logic circuits, like the adder circuit, are called combinational logic circuits. Their

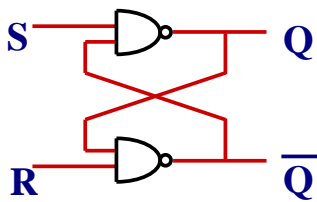
**III.2.1 How memory is developed in logic circuits: SR LATCH.**

Implement a RS latch and verify the table of truth.



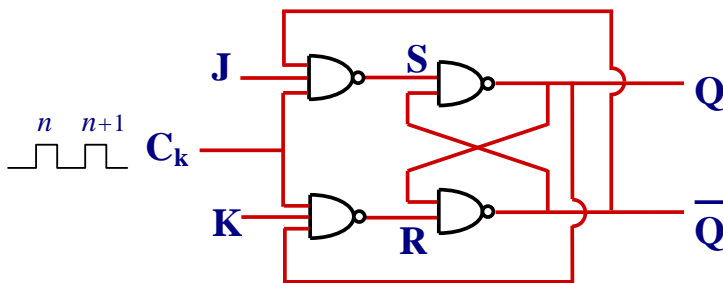
INPUTS		OUTPUTS		
S	R	Q	P	
0	0	1	1	
0	1	1	0	unambiguous output
1	1	1	0	Remembers the previous state
1	0	0	1	unambiguous output
1	1	0	1	Remembers the previous state

Notice, except when  $S=R=0$ , the output satisfies  $P = \overline{Q}$ . Since we want the latter relation to hold, we will forbid the  $S=R=0$  input state.



INPUTS		OUTPUTS		
S	R	Q	$\overline{Q}$	
0	0	1	1	Forbidden
0	1	1	0	Sets $Q \rightarrow 1$
1	1	1	0	Memory
1	0	0	1	Sets $Q \rightarrow 0$
1	1	0	1	memory

### III.2.2 Adding control to the SR LATCH: the JK FLIP=FLOP



**Task:** Fill out the table of truth for the JK flip flop

INPUTS							
J	K	$S_n$	$R_n$	$Q_n$	$Q_n$	$Q_{n+1}$	$Q_{n+1}$
0	0						
1	0						
0	0						
1	1						

### III.2.3 Commercial JK FLIP=FLOP

The JK flip flop is considered a universal flip flop.

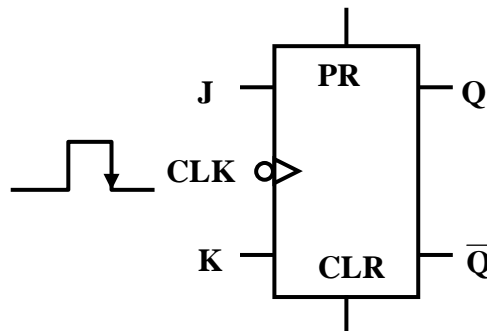
The flip flop is SET when it store a binary 1 ( $Q=1$ )

This is obtained by applying momentarily a LOW at the PR input.

The flip flop is CLEARED (also known as RESET) when it store a binary 0 ( $Q = 0$ )

This is obtained by applying momentarily a LOW at the CLR input.

J	K	$Q_{n+1}$	
0	0	$Q_n$	MEMORY
0	1	0	RESET or CLEAR
1	0	1	SET
1	1	0	TOGGLE



Use a commercially available JK flip flop chip (IC DUAL JK EDGE-TRIG F/F 16 DIP) and familiarize with the its functioning

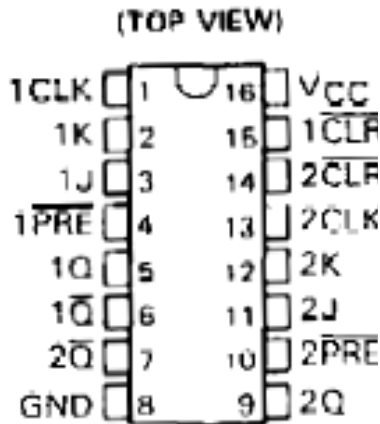
Clear first the flip flop and then check the different mode of operations:

**SET MODE:** Place  $J=1$  and  $K=0$  and verify it causes the flip flop to set ( $Q=1$ ) when the clock transits from high to low.

**RESET MODE:** Place  $J=0$  and  $K=1$  and verify it causes the flip flop to clear (or reset; i.e.  $Q=0$ ) when the clock transits from high to low.

**HOLD MODE:** Place  $J=0$  and  $K=0$  and verify it the out does not change upon the arrival of clock pulses.

TOGGLE MODE: Place J=1 and K=1 and verify changes back and forth to the high and low levels upon the arrival of clock pulses.



FUNCTION TABLE (each flip-flop)

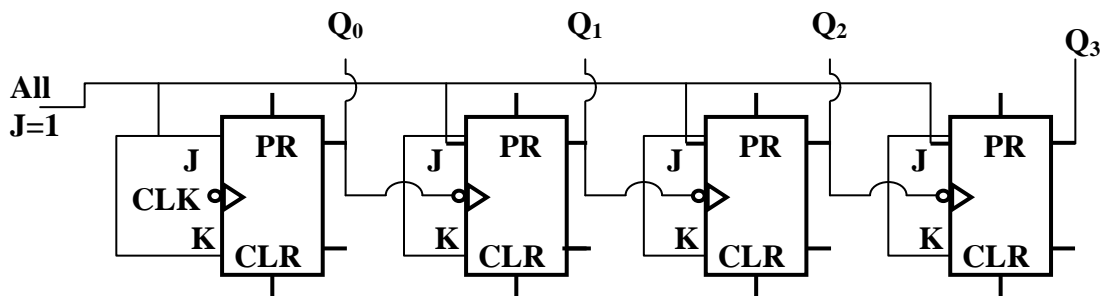
INPUTS					OUTPUTS	
PRE	CLR	CLK	J	K	Q	Q̄
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H <sup>†</sup>	H <sup>†</sup>
H	H	↓	L	L	Q <sub>0</sub>	Q̄ <sub>0</sub>
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q <sub>0</sub>	Q̄ <sub>0</sub>

<sup>†</sup> The output levels in this configuration are not guaranteed to meet the minimum levels for  $V_{OH}$  if the lows at preset and clear are near  $V_{IL}$  minimum. Furthermore, this configuration is nonstable; that is, it will not persist when either preset or clear returns to its inactive (high) level.

### III.2.3 JK FLIP APPLICATIONS

#### Hexadecimal Ring Counter

Construct a hexadecimal ring counter exploiting the toggle mode of the JK flip flop. Implement into the counter the capability to be reset (or clear) at any arbitrary time.

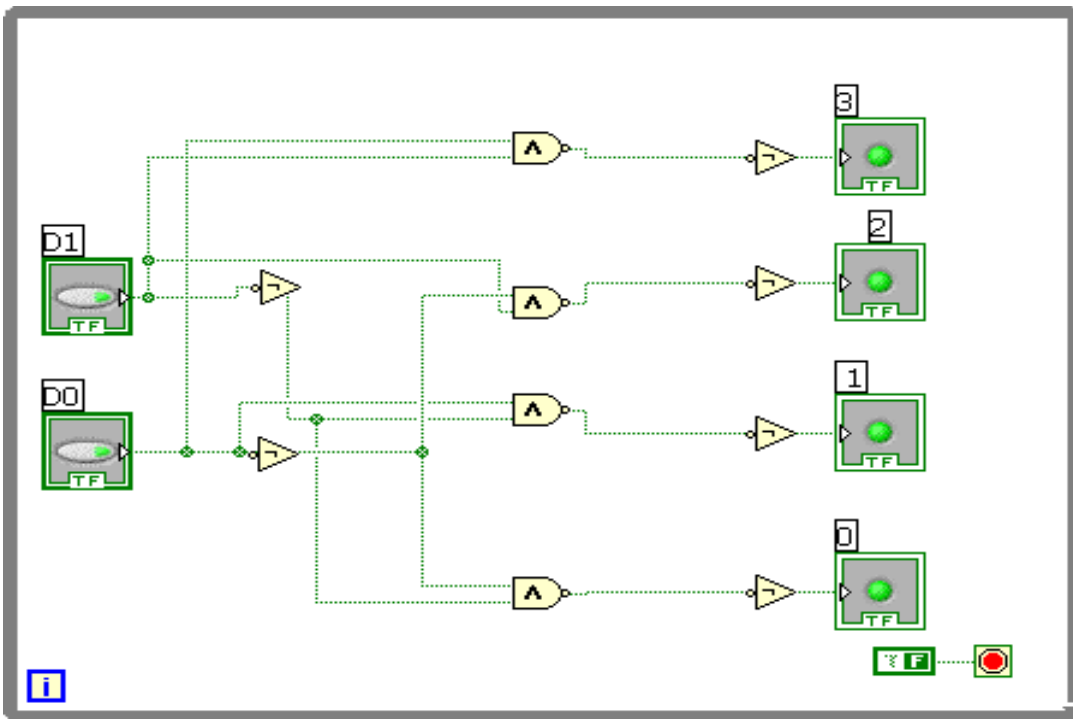


### Decade Ring Counter

It is often more convenient to have counters based on 10 rather than 16. The ring counter you built above can be converted to a decade counter by providing a RESET or CLEAR every time the system reaches 10. Since  $10_{10} = 1010_2$  a NAND gate with inputs  $Q_3 \bar{Q}_2 Q_1 \bar{Q}_0$  could make the trick. Such gate will output 1 when the input varies from 0=0000 to 9=1001, but will transition to zero at 1010. Such output can be feedback to the CLEAR input of the JK flip flops. Implement a decade ring counter. Implement the CLEAR feature described above using the 2-input NAND gates.

### III.3 LabVIEW Design of a Decoder

The figure below shows a LabVIEW to design a 2-to-4 decoder (see figure below).



**TASK:** Use LabVIEW software to build a 3-to-8 decoder using combinational logic circuits.

Helpful reference: Getting started with LabView <http://www.ni.com/pdf/manuals/373427b.pdf>