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 a sequential logic circuit the outputs depend on the inputs plus its history; i.e. it has memory.

Experimental Section-1
You will build an ADDER (using 7400-NAND gates), as an example of combinational logic circuit.

Experimental Section-2.
Sequential logic circuits are introduced through the construction of a RS latch (using NAND gates), which will help us to attain 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.

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 is information coded in electronic digital form?
   II.1A Defining the digital levels using a transistor switch
   II.1B Counting objects: Decimal and binary system
   II.1C Digital electronics

II.1A Digital levels
Consider the transistor switch circuit shown in Fig. 1

Notice, 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 would be OFF.
- No \( I_B \) current, no collector current. It implies \( V_{out} = V_{CC} = 5 \) volts (Digital level 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, \( I_C = 5 \) mA. For a transistor of \( h_{FE} = 100 \), a base current equal to \( I_B = 50 \) μA will saturate the transistor. Thus, by applying an input voltage equal to, for example \( V_{in} = 3(0.7) + (10k\Omega)(50 \mu A) = 2.6 \) V the transistor will be saturated.
- So, we expect that for input voltages in the range \( 2.1 V < V_{in} < 2.6 V \) the transistor will work in the active region.
Note: In these digital electronics applications the transistor is not used in the active region.

\[ V_{CC} (+5 \text{ V}) \]
\[ I_c \]
\[ R_C = 1k\Omega \]
\[ 10k\Omega \]
\[ B \]
\[ C \]
\[ E \]
\[ V_{CE} \]
\[ I_C \]
\[ V_{CE} = V_{out} \]

**Fig. 1** Transistor switch. For \( V_{in} < 2.1 \text{ V} \) the output level is 5V; for \( V_{in} > 2.6 \text{ V} \) the output levels is close to 0 V.

If \( V_{in} = 2.6 \text{ V} \):
- As indicated above, for an input voltage of 2.6 V the transistor will be saturated, and the collector current would be \( I_c = 5 \text{ mA} \). The corresponding voltage drop across \( R_E \) is then \( 1k\Omega \times 5 \text{ mA} = 5 \text{ Volts} \), which makes \( V_{out} = 0 \text{ Volts} \).

If \( V_{in} > 2.6 \text{ V} \):
- The transistor remains saturated and \( V_{out} = 0 \text{ Volts} \) (Digital level 0)

**Fig. 2** Switch transistor response and corresponding definitions of digital output signal levels.
II.1B Decimal and binary systems

How to systematically count the elements of this system?

Using an arbitrary numerical system

We will count them in sub-groups of sizes $A$, $B$, and $C$.

- 2 groups of “size” $A$
- 3 groups of “size” $B$
- 1 group of “size” $C$

which can be expressed in the following notation:

$$2A \ 3B \ 1C$$

Fig. 4 Grouping under an arbitrarily given numerical system.

Using the decimal system

We will count them in sub-groups of $10^0, 10^1, 10^2, 10^3, \ldots$
When the binary system is assumed implicitly being used, we just write: \[10110\]

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

5 V 1

0 V 0

5 V 1

Fig. 7.

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: GATED FLIP=FLOP

III.2.3 Reducing the gating time: EDGE TRIGGERED FLIP FLOPS

III.2.4 Eliminating the forbidden sates: JK FLIP FLOP

III.2.5 JK Flip-flop applications

III.3 LabView Design of a Decoder

III.4 Registers

III.5 Memory Circuits

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 (Integrated circuit 7404 INVERTER)

\[
\begin{array}{c|c}
\text{Input} & \text{Output} \\
A & \overline{A} \\
0 & 1 \\
1 & 0 \\
\end{array}
\]

The small circle indicates inversion

Note: The overscore on the symbol A means NOT or logical complement.

AND gate

\[
\begin{array}{c|c}
\text{Input} & \text{Output} \\
A & B & Q = A \cdot B \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

NAND gate (Integrated circuit 7400 NAND)

\[
\begin{array}{c|c}
\text{Input} & \text{Output} \\
A & B & Q = \overline{A} \cdot \overline{B} \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
OR gate (Integrated circuit 7402 OR)

\[
\text{Inputs} \quad \text{Output} \\
\begin{array}{cc|c}
A & B & Q = A + B \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

NOR gate (Integrated circuit 7402 NOR)

\[
\text{Inputs} \quad \text{Output} \\
\begin{array}{cc|c}
A & B & Q = A + B \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

EXCLUSIVE OR gate

\[
\text{Inputs} \quad \text{Output} \\
\begin{array}{cc|c}
A & B & Q = A \oplus B \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
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 \( B_3 B_2 B_1 \).

\[
\begin{array}{c|c|c|c|c|c}
  & C_2 & C_1 & A_3 & A_2 & A_1 \\
\hline
A_3 & A_2 & A_1 & B_3 & B_2 & B_1 \\
\hline
S_4 & S_3 & S_2 & S_1 & & \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
<td>( B_1 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 8 Table of truth for implementing an adder circuit.

**TASKS:**

- To build a simple **half-adder** for adding \( A_1 \) and \( B_1 \), as well as the carrier of their sum \( C_1 \), using only NAND gates. (Suggested procedure is given below, leading to the design shown in Figs. 9 and 10).
- Subsequently, implement a **full adder** for (in addition to adding \( A_1 \) and \( B_1 \)) also adding: \( A_2, B_2, \) and the previous carrier \( C_1 \), as well as to produce the forward carrier \( C_2 \). (Suggested procedure is shown in Fig. 11).

**About the HALF ADDER**

The diagram above (table of truth for the adder) suggests that all we need is a XOR and AND gates. Assuming we have have available only NAND and NOR gates, a bit a Boolean algebra comes timely to the rescue (see below).

Note: If you do not have a NOR gate, implement one based on NAND gates.

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

**TASKS** First, verify explicitly (making a corresponding table of truth) the following properties:

- \( A + B = \overline{A} \cdot \overline{B} \)
- \( A \cdot B = \overline{A} + \overline{B} \)
- \( A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B \)
\[ A \oplus B = (A + B) \overline{A \cdot B} \]

Experimental implementation of \[ A \oplus B = (A + B) \overline{A \cdot B} \]

\[ A \oplus B = (A + B) \overline{A \cdot B} \]

\[ = A + B + A \cdot B \]

**Fig. 9** XOR design with NAND and NOR gates

Hence the following implementation constitutes a half adder circuit.

**Fig. 10** Half adder circuit.

**FULL ADDER**

**Task:** Build the circuit below and verify that it works as a full adder (it adds two digits plus a previous carrier).

In particular, explain in detail how the OR gate makes the trick for the full-adder to work.
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: GATED FLIP FLOP

III.2.3 Reducing the gating time: EDGE TRIGGERED FLIP FLOPS

III.2.4 Eliminating the forbidden states: JK FLIP FLOP

III.2.5 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

III.2.1 How memory is developed in logic circuits: S-R LATCH.

**TASK:** Implement the circuit shown in Fig 10 and verify the table of truth.

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S' R'</td>
<td>Q P</td>
</tr>
<tr>
<td>0 0</td>
<td>1 1</td>
</tr>
<tr>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
</tr>
</tbody>
</table>

Unambiguous output

Remembers the previous state

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>S' R'</td>
<td>Q P</td>
</tr>
<tr>
<td>1 0</td>
<td>0 1</td>
</tr>
<tr>
<td>1 1</td>
<td>0 1</td>
</tr>
</tbody>
</table>

Unambiguous output

Remembers the previous state

**Fig. 10** Latch circuit displaying electronic memory properties. Verify your circuits works as expected.
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. Hence, the above result is equivalently expressed as follows:

\[
\begin{array}{c|c|c}
\text{INPUTS} & \text{OUTPUTS} \\
\hline
S' & R' & Q & Q' \\
\hline
0 & 0 & 1 & 1 & \text{Forbidden} \\
0 & 1 & 1 & 0 & \text{Sets } Q \rightarrow 1 \\
1 & 1 & 1 & 0 & \text{Memory} \\
1 & 0 & 0 & 1 & \text{Sets } Q \rightarrow 0 \\
1 & 1 & 0 & 1 & \text{memory} \\
\end{array}
\]

\[\text{Fig. 11} \quad \text{S-R latch with complementary outputs.}\]

**III.2.2 Adding control to the SR latch: GATED FLIP FLOP**

The SR latch requires a few refinements. For example, it responds to its input signals immediately and at all times. Problems can occur when logic signals that are supposed to arrive at the same time actually arrive at slightly different times due to separate delays. Such timing problems can create short unwanted pulses called glitches. The gated flip flop shown below corrects this problem.

\[\text{Fig. 12} \quad \text{Gated latch. (No need to EXPERIMENTALLY implement this circuit in this lab session). But verify its corresponding table of truth (as given in the see text below).}\]

Notice:
- The circuit responds to input logic signals only when the clock input \( C_k \) is in state 1.
- When \( C_k \) is in state 0, the outputs of the NAN gates on the left become equal to 1 and, thus, the outputs \( Q \) and \( \overline{Q} \) remains in memory state.
- The table of truth for the circuit in Fig. 12 can be obtained directly from the table of truth of the circuit in Fig 11 by simply interchanging the levels 1 and 0.
II.2.3 Reducing the gating time: EDGE TRIGGERED FLIP FLOPS

To even further protect the flip flops from glitches, the gating time (the time during which the input signals affect the output signals) can be reduced by making the circuit sensitive only when the clock signal makes transitions from either high to low or vice versa. This is known as edge triggering.

![Leading edge triggering](image1.png)

![Trailing edge triggering](image2.png)

**Fig. 13** Symbols for edge triggered flip flops. Triggering at the edges limits the time during which the inputs are active.
### III.2.4 Eliminating the forbidden sates: JK FLIP FLOP

A problem with the S-R latches is the forbidden state at the inputs. The circuit below shows an alternative to correct such shortcoming.

![Diagram of a J-K flip flop](image)

**Fig. 14** Version of a J-K flip flop. (No need to implement this circuit in this lab session).

The corresponding table of truth is,

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>J 1</td>
<td>K 1</td>
</tr>
<tr>
<td>J 1</td>
<td>K 0</td>
</tr>
<tr>
<td>J 0</td>
<td>K 1</td>
</tr>
<tr>
<td>J 0</td>
<td>K 0</td>
</tr>
</tbody>
</table>

**Fig. 15** J-K flip flop and its standard table of truth. (We will provide a flip flop chip).

When the inputs J and K are equal to 1, the outputs Q and \(\overline{Q}\) will change to its complementary value after each clock pulse.

The toggle feature reveals the advantage of edge triggering for the JF flip flop: if the gating time were extended in time, the output state would oscillate back and forth and the eventual final output (when the gating is off) would be undetermined.

The JK flip-flop is a very versatile device, and is probably the most commonly used form of flip-flop in digital electronic and control circuits.
**D- FLIP FLOP:** Transferring the input to the output at the active clock edge.

![D Flip Flop Diagram](image)

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>(Q_{n+1})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Fig. 16* D flip flop. Notice it has the effect of transferring the input to the output at the active clock edge.

**T- FLIP FLOP**

![T Flip Flop Diagram](image)

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>(Q_{n+1})</td>
</tr>
<tr>
<td>1</td>
<td>(\overline{Q}_n)</td>
</tr>
<tr>
<td>0</td>
<td>(Q_n)</td>
</tr>
</tbody>
</table>

*Fig. 17* The T flip flop toggles with the clock pulse when T=1 and does not toggle when T=0.

**Commercial JK FLIP FLOP**

Use a commercially available JK flip flop chip (IC DUAL JK EDGE-TRIG F/F 16 DIP) and familiarize with the its functioning. Download the data sheet of the flip flop provided in the lab.

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.

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=1) 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.
III.2.3 JK FLIP APPLICATIONS

Hexadecimal Ring Counter

TASKS:
- 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.
- Make a diagram displaying the digital signals of the clock and the four Q-outputs as a function of time.

![Fig. 18 Asynchronous counter. Notice, the JK flip flops are operating in Toggle mode.](image)

Hints: It may occur that when connecting the Q-outputs to the monitoring LEDs, the latter may affect the functioning of the counter (the Q-output not being able to drive the clocks.) As potential solutions, you may:
- Opt to display the output of the counter by monitoring the \( \overline{Q} \) outputs instead (thus relieving the Q-outputs to do its job driving the clock of the next flip-flop).
- Opt to keep using the same design of Fig 18, but inserting a resistor (try 1k\( \Omega \), or 10k\( \Omega \)) between the Q-output and the corresponding LED.

Decade Ring Counter

It 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 \( 10_{10} = 1010_2 \) an NAND gate with inputs \( Q_3 \overline{Q}_2 \overline{Q}_1 \overline{Q}_0 \) 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.

TASK: Implement a decade ring counter. Implement the CLEAR feature described above using the 2-input NAND gates.

III.3 LABVIEW DESIGN: 3 to 8 Decoder (This section is required in 2016).

The desktop computer in our lab have the LabView software. Your instructor will guide you with some initial tip on how to use that software. To optimize the lab time get familiar with the software with information available at https://www.youtube.com/watch?v=Em5R_RM8E08

The figure below shows a LabVIEW design of a 2-to-4 decoder (see figure below.) That is, for a binary input 00 only the “0” LED lights up; for the binary input 01 only the “1” LED lights up; etc.
Feel free to use your own strategy to build a 2-to-4 decoder (as an alternative to the one in Fig. 19). The only requirement we impose is that you make your design in such a way that it can be straightforward extrapolated to build a 3-to-8 decoder, then to a 4-to-16 decoder, … , and so on.

**Fig. 19** LabView design of a 2 to 4 decoder.

**TASK:**

i) Use LabVIEW software to build a 3-to-8 decoder using combinational logic circuits.

ii) Once step i) is done, try to implement the same task using the **loop function**.

   (Address the advantage or difficulties).

iii) Use the **case structure** option to implement the same task.

Helpful references:

https://www.youtube.com/watch?v=Em5R_RM8E08


https://www.youtube.com/watch?v=0Ea2lQeCIMY&ecb=ANyPxKqAlmx6xwlb-iiLSGqUjppRRbBa6seDKhO6BSEXz0njX1zEFygvV48c242AIKH3sLtmSnSE_8oLDCDzEq1nj3NuyNChA


**III.4 REGISTERS** (No experiment is needed to implement in this section).

A **register** is a series of flip flops arranged for organized storage or processing of binary information.

Information is represented in a computer by groups of 0’s and 1’s called **words**.

A 8-bit word is called a **byte**.

Large computers work with words of 32 or more bits.

A register in a computer with 8-bit words would require 8 flip flops to store or process simultaneously the 8 bits of information.

Words of information are moved around in a computer on a **bus**.
The bus consists of a number of conducting paths connecting all potential source-registers with all potential destination-registers.

**Fig. 20 Parallel input and parallel output.** Loading a register of 4 D-type flip flops from a bus. At the trailing edge of the LOAD signal, the information on the bus is stored in the register.

**Shift register**

Sometimes digital information must be sent over one channel. In this case, bits are sent in serial form.

When digital information must be received in serial form, a shift register may be used to accept the serial information and convert it to parallel form.

**Fig. 21 Shift register.** The input at the D flip flop is shifted to the output at the action of a clock pulse.

**III.5 MEMORY CIRCUITS (No experiment is needed to implement in this section).**

**Read-Only Memories**

The decoder alluded in section III.3 above are an example of what has come to be called a read-only memory, or ROM. A ROM associates a specific output binary number with each input binary number according to its fixed internal logic.

The fixed relationship between input and output distinguishes the ROM from other memory circuits.
An important application of ROMs is to provide look-up tables for mathematical functions, such as trigonometric, exponential, square root, and logarithmic functions.

In certain applications, most notably in microprocessors circuits, it proves useful to be able to enter the information in a ROM after the fabrication of the device. In such a programmable ROM, or PROM, the desired memory bits are stored by electrically altering the circuit connections. Similarly, erasable PROM are available in which information is stored as charge on stray capacitance at the gate electrodes of a MOSFET ROM without actually destroying the gate electrodes. These bit patterns can be erased by irradiation with ultraviolet light to discharge the gate capacitors or other electrical signals.[Ref 3.]

**Shift Register Memories**

In many applications it proves useful to store digital information temporarily for recall at later time. This is a memory into which information can be rapidly written and changed, as well as read out. Shift registers are convenient and effective memory circuits for this purpose.

**Random-Access Memories**

The access time in a shift-register memory depends upon the word address and upon the word storage capacity of the memory since information is only available sequentially at the shift register outputs. In a random-access memory (RAM) the access time is independent of the location of information in the memory; addressing logic permits immediate access to any information stored in the memory. A RAM is organized into words lines and bit lines, and information is stored at each intersection by the state of a flip flop memory cell.

**References**