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 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 and 7402-NOR 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 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.
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 \( \beta = 100 \), a base current equal to \( I_B = 50 \) \( \mu \)A 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 $2.1V < V_{in} < 2.6V$ the transistor will work in the active region.

![Transistor and circuit diagram](image1)

**Fig. 1** Transistor switch. For $V_{in} < 2.1V$ the output level is 5V; for $V_{in} > 2.6V$ the output levels is 0 V.

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

If $V_{in} > 2.6V$.
• The transistor remains saturated and $V_{out} = 0$ Volts **(Digital level 0)**

![Graph showing switch transistor response and corresponding definitions of digital output signal levels](image2)

**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 $C$.

Fig. 3

Using the decimal system
We will count them in sub-groups of $10^0$, $10^1$, $10^2$, $10^3$, ...

Fig. 4 Grouping under an arbitrarily given numerical system.

Fig. 5. Grouping under the decimal numerical system
**Binary system**

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

- 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 $2^1$

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

When the binary system is assumed implicitly 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

**Fig. 6.** Grouping under the binary numerical system

---

**II.1C Digital electronics**

Using an array of transistor circuits

Interpreted as logic levels

<table>
<thead>
<tr>
<th>Input Voltage</th>
<th>Output Voltage</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{in}$</td>
<td>$V_{out}$</td>
<td>1</td>
</tr>
<tr>
<td>$V_{in}$</td>
<td>$0$ V</td>
<td>0</td>
</tr>
<tr>
<td>$V_{in}$</td>
<td>$0$ V</td>
<td>1</td>
</tr>
</tbody>
</table>

**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 states: 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

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

III.1A Digital logic gates

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

NOT gate (Integrated circuit 7404 INVERTER)

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( \overline{A} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The small circle indicates inversion.

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

AND gate

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( B )</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>
NAND gate  (Integrated circuit 7400 NAND )

\[ Q = \overline{A \cdot B} \]

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

OR gate (Integrated circuit 7432 OR)

\[ Q = A + B \]

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

NOR gate  (Integrated circuit 7402 NOR)

\[ Q = \overline{A + B} \]

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
EXCLUSIVE OR gate

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

III.1B Digital Arithmetic: Adder circuit

The diagram on the left (figure below) indicates an addition operation of two binary numbers: A3 A2 A1 and B3 B2 B1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</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 A1 and B1, as well as the carrier of their sum C1, using only NAND and NOR 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 A1 and B1) also adding: A2, B2, and the previous carrier C1, as well as to produce the forward carrier C2. (Suggested procedure is shown in Fig.11).

HALF ADDER
The diagram above (table of truth for the adder) 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 gate out of NAND and NOR gates**

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

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

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

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

**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).

![Full adder circuit](image)

**Fig. 11** Full adder circuit. (Notice, you will have to design an OR gate out of NAND gates).

### 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'</td>
<td>R'</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fig. 10** Latch circuit displaying electronic memory properties.
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:

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

**Fig. 11** 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.

![Gated latch diagram](image_url)

**Fig. 12** Gated latch..

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.
### III.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](Fig. 13)

![Trailing edge triggering](Fig. 13)

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.

![Fig. 14 Version of a J-K flip flop.](image)

The corresponding table if truth is,

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>K</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. 15 J-K flip flop and its standard table of truth.](image)

When the inputs J and K are equal to 1, the outputs Q and Q̅ 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

![D Flip Flop Diagram]

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

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Q&lt;sub&gt;n+1&lt;/sub&gt;</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

T- FLIP FLOP

![T Flip Flop Diagram]

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

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Q&lt;sub&gt;n+1&lt;/sub&gt;</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Commercial JK FLIP FLOP

Use a commercially available JK flip flop chip (IC DUAL JK EDGE-TRIG F/F 16 DIP) and familiarize with its functioning. The data sheet is available on the website of this course. [http://www.physics.pdx.edu/~larosaa/PH-315/Datasheet_IC_DUA_%20JK_EDGE-TRIG_FF_16-DIP.pdf](http://www.physics.pdx.edu/~larosaa/PH-315/Datasheet_IC_DUA_%20JK_EDGE-TRIG_FF_16-DIP.pdf)

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

**TASK:** 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. Also, make a diagram displaying the digital signals of the clock and the four Q-outputs as a function of time.

![Hexadecimal Ring Counter Diagram](image)

**Fig. 18** Asynchronous counter.

**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 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 1kΩ, or 10kΩ) 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 \(Q_3 \overline{Q}_2 Q_1 \overline{Q}_0\) 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

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.
**III.4 REGISTERS**

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.
III.5 MEMORY CIRCUITS

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

Field-programmable gate array

Task: Make a two-page (or more) description of the working principle of the FPGA


References