كلية الهندسة - جامعة الزقازيق

الفرقة: الثالثة هندسة الحاسبات والمنظومات

المقرر: دوائر الحاسب المتكاملة

الإسم: احمد محمد عبدالمنعم محمد

الرقم في السكشن: ١٨

رقم الجروب وإسم الموضوع: ٦

Fastest Push First Indicator

Overview

Fastest Push First Indicator is a well-known system which is used in quiz type games. It is an indicator that displays the number of the fastest player who pushes the button. Also, there is a speaker to notify the players and the referee that the button has been pressed. It is implemented by adding simple blocks and ICs together.

Components

7805 Voltage Regulator

74LS75 4-Bit Bistable Latch

74LS20 Dual 4-Input NAND Gate

74LS147 9-Lines To 4-Lines Priority Encoder

74LS04 Hex Invertor

74LS47 BCD To 7-Segment Decoder

NE555 Timer

Common Anode 7-Segment Display

Capacitors

Resistors

Speaker 8Ω

Implementation

We used 4 **push buttons** to receive player's reactions and also, we used a **7-segment** to identify the number of the first player to push the button. And **the speaker** which generates a notification depending on **IC NE555** which works as a timer to generate the clock signal. When a player presses his button, the **latch** transmits the signal to the **priority encoder** which outputs the code in binary representation for the number of the pushed button. Then, for design purposes we put **NOT** gate to invert the state. After inverting the signal we used **7-segment decoder** to transform the binary code (which we took from the **priority encoder** from the last stage) to an input for the **7-segment**.

Options & Additions

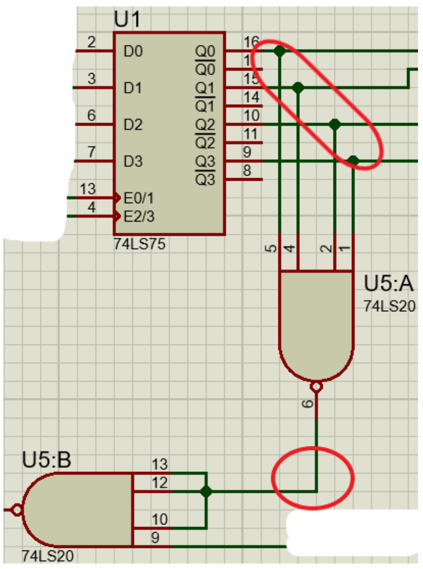
Definitely, this system will not work properly if we implement that design only. This system should prevent the users to push two or more buttons at the same time in order to display only one player number each game which will be the number of the fastest player to push the button. Also, the system should have a method to notify the players and the referee that there is a pushed button.

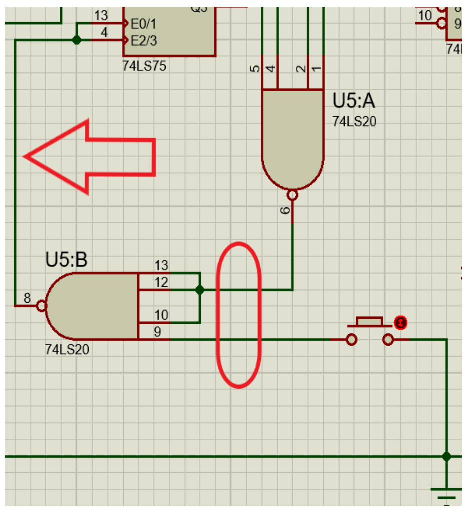
Only one Button: In order to find a way to display only the fastest player number, we will make use of our latch enable pins. As we will put in the input of our latch enable pins logic zero to disable the **latch** function immediately after pushing any button.

Now we have received the fastest push. But How will we deactivate the **latch** enable immediately after pushing one of the buttons to prevent any other pushes??

We will use 2 **NAND** gates.

The *first gate input pins* will be connected to the *latch output pins* and its output will be connected to the second gate.





The ***second gate output*** will be connected to the ***latch enable*** and it has two inputs.

The ***first input*** is the ***output of the first gate*** and the ***second input*** will be a ***reset button*** to reset the whole system after the game has finished. Now, it is clear that the first **NAND** gate will sense any change in the **latch** output which means that there is a pushed button and will also change the output of the second gate and disable the **latch**. This means any other pushed buttons will not change anything in the **latch** output.

So, we have pushed a button and its number have been displayed on the **7-segment** and any other push buttons will not change anything because **latch** enable has been deactivated after the fastest push due to our **NAND** gates.

Reset Button: Now we want to start a new game and remove the current player number. So, we will use reset button.

But How can we design this button?

We will connect it to the input of the second **NAND** gate and the ground from the other side. So, when we push it, we put logic zero on the second gate input and our latch enable will be activated again and the latch is ready for receiving a new push now.

Speaker & Timer NE555: We want to add a notification method to the system. So, we will use an audio notification.

We will use **NE555** timer to generate clock signal in order to get audio signal so that we could use the speaker. The main pins in **NE555** timer in our application are ***reset pin (pin 4)*** and ***output pin (pin 3)***. Reset pin is an active low pin. So, if we want to reset the timer, we put logic zero to it. Also, we will connect the output pin to the speaker.

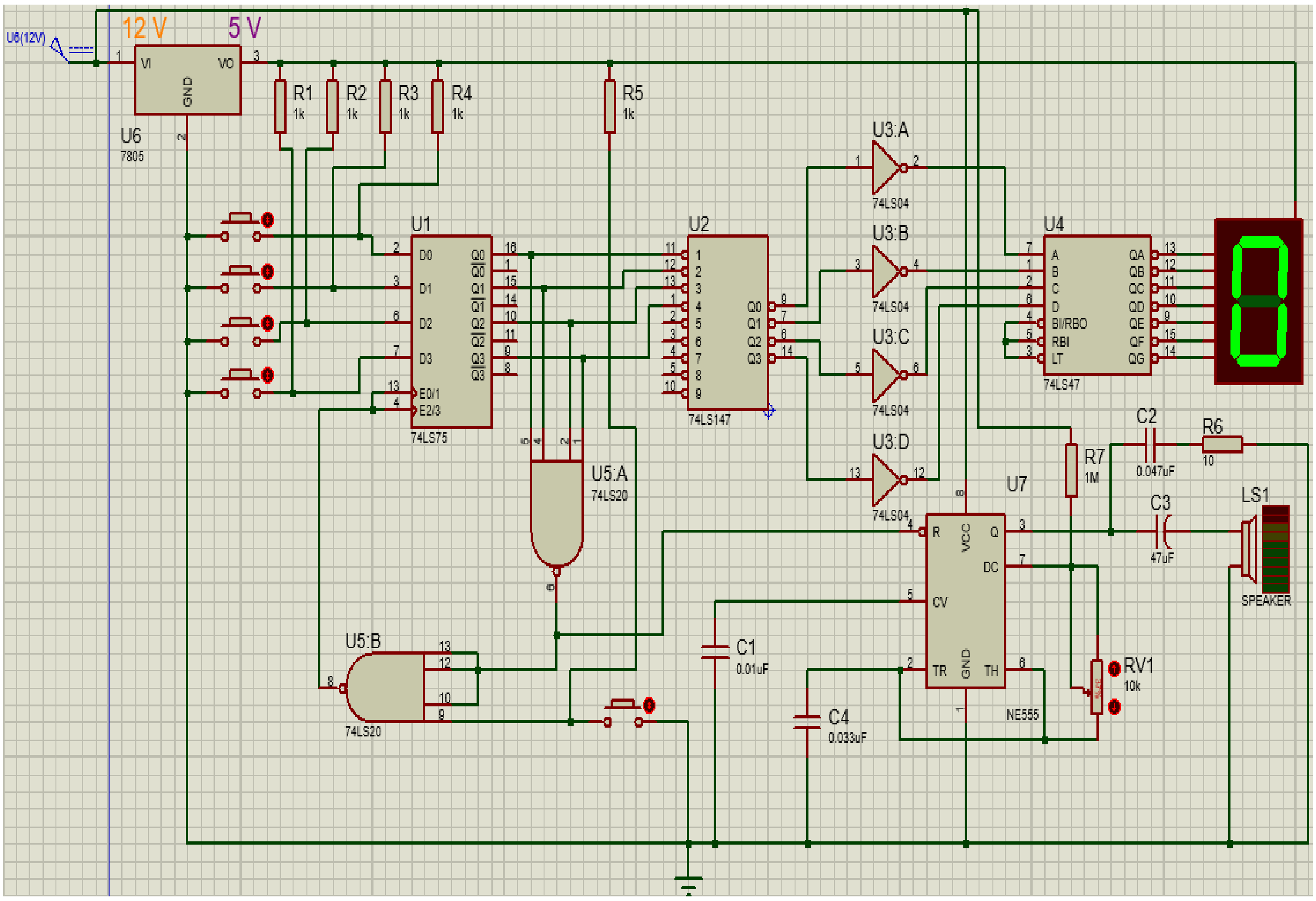
But How will we control the timer? Or How will the speaker work when we push any button?

As I mentioned, to identify any changes on the push buttons we used a **NAND** gate connected to the output of the **latch**. The gate changes its output state from logic zero to logic one when we push any of the buttons. So, we can connect the output of this gate to the timer reset which is active low reset. And there will be 2 cases.

Case #1 when there are not any buttons pushed, the output of the first NAND gate will be logic zero. So, the timer will reset permanently and there will be no output and the speaker will not work.

Case #2 when we push a button, the output of the gate will be logic one. So, the timer will not reset, the oscillations will be generated and the speaker will work.

Circuit Diagram



My Role

As a team, we have divided ourselves into *3 sub-groups* and I was a member in the *first sub-group*. Our mission was to design, implement and run the *simulation on Proteus*. I have helped my classmates in *additions design*. I have helped in the logic design using 2 **NAND** gates to activate/deactivate the **latch enable** and **reset button** function. Also, deep searching over the internet about how we can adjust **NE555** timer to generate oscillations in order to run the **speaker**.