



---

# TIMER

---

DSD first year project



MAY 31, 2023

ANDREICAN RARES

PROJECT SUPERVISOR: Ing. Diana Pop

# Table of Contents

1	Specifications .....	2
2	Design.....	3
2.1	Black Box .....	3
2.2	Control and Execution Unit.....	3
2.2.1	Mapping the inputs and outputs of the black box on the two components .....	4
2.2.2	Resources (breakdown of the Execution Unit).....	4
2.2.3	Block Diagram for first breakdown .....	6
2.2.4	State diagram of the Control Unit .....	6
2.2.5	Detailed diagram of the project .....	10
3	User manual .....	10
4	Technical justifications for the design.....	10
5	Future developments .....	11
6	References.....	11

# Timer

## 1 Specifications

This paper presents the implementation of a device that solves the needs of a timer, as it follows:

- The device has a set of 6 BCD 7-segment device. They are divided in 3 groups, 2 for minutes, 2 for seconds and other 2 for the alarm timer(still seconds)
- The maximum value that can be shown is 99 minutes and 59 seconds
- The device uses a set of 3 main buttons : M, S, START/STOP and another set of 2 buttons that are implementing the alarm and the Command Unit: set\_alarm and start
- Supposing a initial state is ZERO (00:00), if the button START/STOP is pressed, the timer starts to count up. Pressing the same button stops the timer. A third press on the button makes the timer count up again- and so on. At 99:59, the times resets to 00:00 and continues to count up; If both M/S buttons are pressed together, the timer resets to a initial state of „Wait”;
- In any state, if M is pressed, the counter stops and the M counter counts up. The device then waits for the M button to be pressed, meaning the minutes have been set. Pressing the START/STOP button Pressing the S button works exactly the same and they can be used at the same time(set S and set M, then make the device count down).
- Once the S/M has been set by following the above behaviour, the timer starts to count down. Once it reaches 00:00, it activates an alarm buzz(described by a blinking light) that is up for a certain period of time. That period of time is decided when a user starts using the timer with the help of alarm and START/STOP button. Further information on using the device will be found in the „user manual” section;

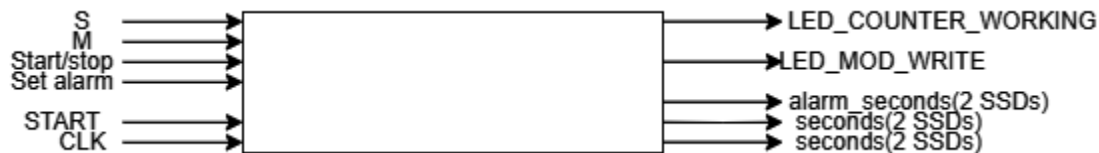
## 2 Design

### 2.1 Black Box

Inputs :

- CLK, clock from the FPGA board
- M, the minutes button described by the text
- S, the seconds button described by the text
- Start/Stop, the start/stop button described by the text
- SET\_ALARM, button described by the text
- START BUTTON, described by the text
- LED\_MOD\_WR ,a led signaling that the automata is waiting for the user to set alarm
- LED\_COUNTER\_WORKING, a led signaling that the counter is in mode “count”
- Seconds, minutes, alarm , 3 7 segments BCD display showing the desired numbers
- alarm buzz, a led signaling the alarm is working

*Figure 1 Black Box of the system*



. All components are bonded to the same RESET input signal.

### 2.2 Control and Execution Unit

The system's black box must be further broken down in order to find implementable components. We will do a **top-down** breakdown of the problem until we get to known circuits, and then we will implement **bottom-up**.

The first breakdown of any system is one in which we will differentiate between the **control logic** in the system and the **system resources**. The control logic is represented by the Control Unit (CU) and the resources are represented by the Execution Unit (EU). Any algorithm can be broken down in this way (*the abstract representation of an algorithm is done through a flow-chart*).

### 2.2.1 Mapping the inputs and outputs of the black box on the two components

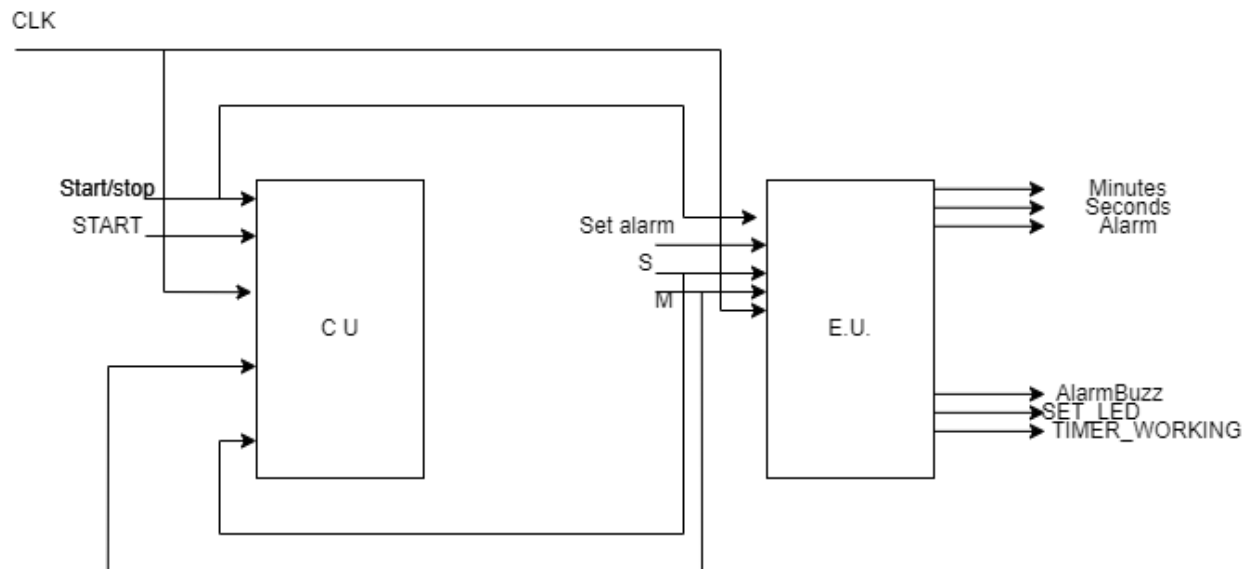


Figure 2 Mapping the inputs and outputs of the black box on the inputs and outputs of the units

We can divide both inputs and outputs into 2 categories: **data** and **control**. This separation is essential at the beginning.

**Data inputs:** values for different things (ticket cost, distance traveled, pin card number, number of tickets, etc.).

**Control inputs:** confirmation button, M, S buttons, reset button, etc.

**Data outputs:** values to be displayed to the user (time remaining in a program, rest due, calculated cost of a ticket, etc.)

**Control outputs:** warning or warning signals to the user, through which we can control and guide the user through the operation of the system (LEDs, audible signal).

### 2.2.2 Resources (breakdown of the Execution Unit)

In order to further establish the links between the CU and the EU, we must first identify **the resources on the basis of which we make decisions**. These resources can **generate signals to the control unit** and can be **controlled by the CU** via Enable or Reset signals.

Any decision-making information must come from a resource that generates that information and passes it on to UC.

Resources can be **simple circuits**, which can be implemented directly (counter, register, etc.) or **complex resources** (remainder algorithm, multiplication algorithm, etc.). These complex resources may appear in the first breakdown with black boxes to which we must establish inputs

and outputs, but later they must be further broken down (usually also in CU and EU) until we reach known circuits.

## RESOURCES

We are indeed facing a problem that must use TIME, count up to 99 and 59 respectively. For the SECONDS and MINUTES units, we will use 2 bidirectional counters with special properties. This unit will be called “BIG\_CNT” in the diagram; For simplicity, although the needs of the automata are less, I’ve decided to implement the width of the counters output as 8. Below will be shown the resources I’ve found useful for implementing this device:

1. Frequency divider – 1sec

This resource has a single *CLK* input received from the clock. It transforms the received 100Mhz frequency into 1Hz frequency or 1 second. The single output will be the new frequency and it will be used by other resources E.U.

2. 60 seconds TIMER counter(alarm)

This is a 8-bit modulo 60 counter. As inputs, it gets the clock, a signal that decides its internal state, a input from the control unit saying whether it should work or not; As outputs, it provides signals to CU, the register(when it has to be reloaded) and the internal counter output.

3. Minute Timer

This is a 8-bit modulo 100 invert counter. This circuit will have different inputs (such as mode, load, CU/CD, RESET(the and between M and S at the same time will be reset)) and outputs for the 7-segment BCD display.

4. Seconds timer

This is a 8-bit modulo 60 invert counter. The circuit will have different inputs(such as mode,load, CU/CD, RESET(the and between M and S at the same time will be reset)) and outputs for the 7-segment BCD display.

5. Register

This is a 8-bit register that helps the alarm reset itself. It has as input the clock, a write input(communicates with the alarm) and the reg\_a input/output, what should be loaded inside the register and what should be extracted from it.

6. Binary to decimal resources

They are resources needed for converting the bitstream from the seconds timer/alarm/minutes timer to a 7-segment BCD display.

7. Anodes/Catodes unit

This unit helps the FPGA show the correct outputs on the display.

8. Button debouncer

This unit debounces the buttons and prevents misinterpretation of the input signals from the user.

### 2.2.3 Block Diagram for first breakdown

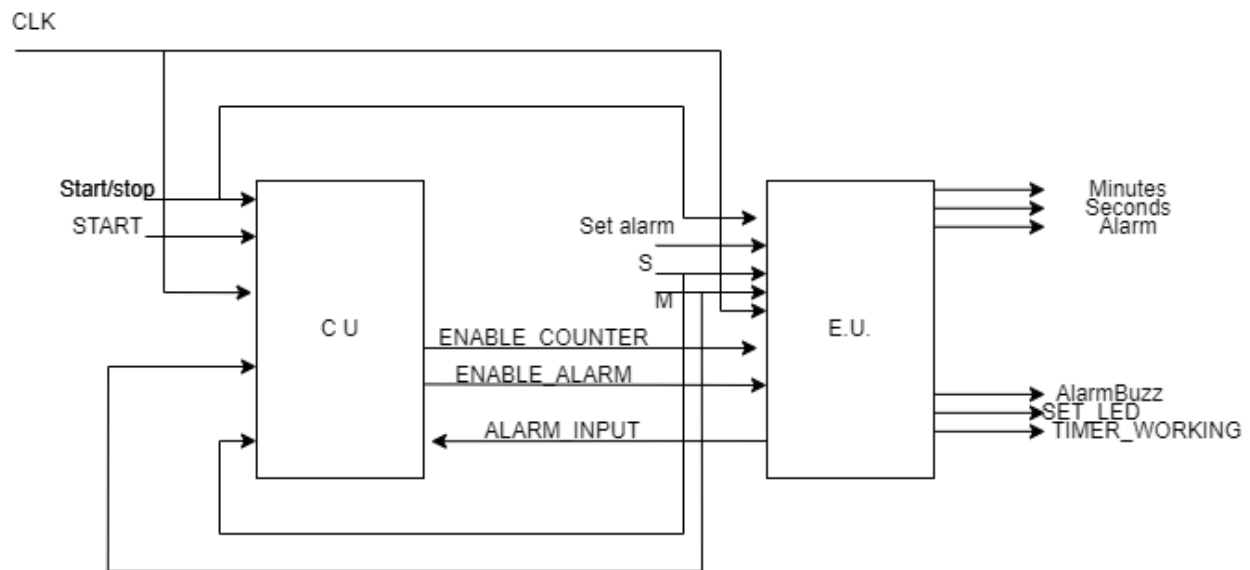



Figure 3 Mapping the connections found between the control and execution units


---

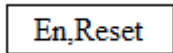
*It is worth noting that CLK has a higher input frequency (100Mhz for FPGAs), so there should be a frequency divider within the EU as another resource to generate the 1 min period for the 2 counters.*

---

### 2.2.4 State diagram of the Control Unit

**States** are represented by . A state represents a moment of time (a period).

The **decisions** made in each state are represented by .

The **outputs** generated in each state are represented by . Inside the rectangle are the outputs that are true at that time.

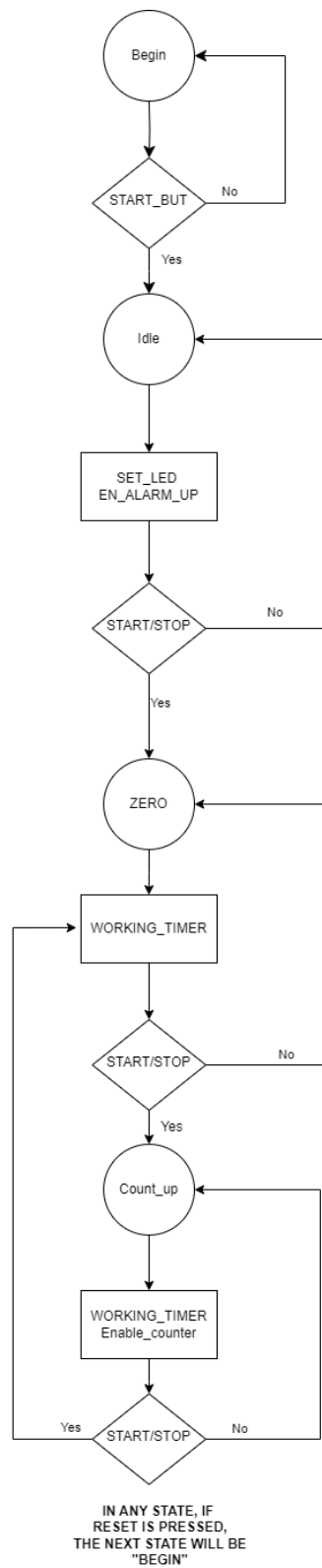


Figure 4.1 State diagram



This state diagram is somehow simplified for the reader. The “Count\_up” state actually holds more hard-coded states, that I’ve called “internal states of a counter”. This state also holds the working mode of the alarm, although its only interaction with the user is in state “Idle”.

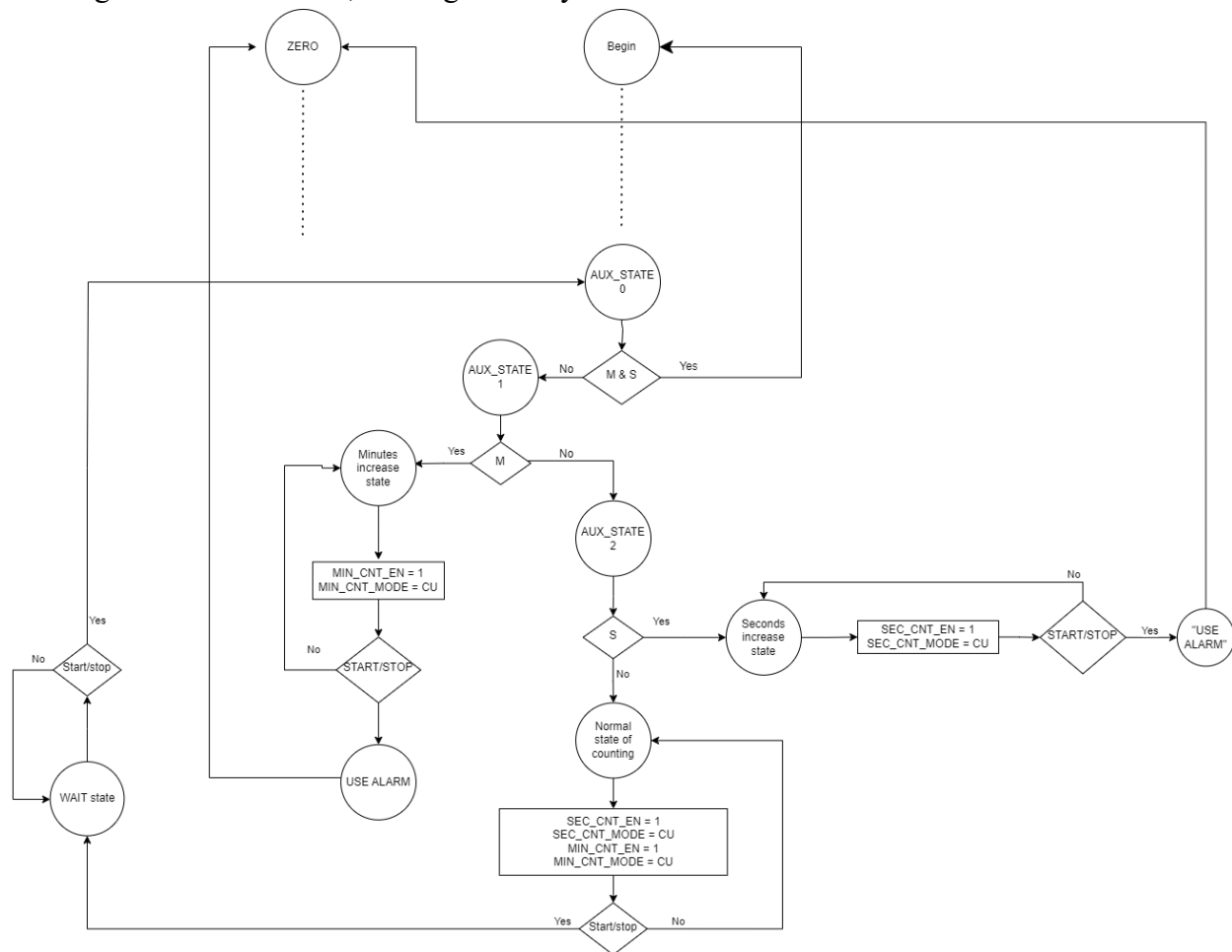


Figure 5.2 State diagram for the counter unit

The last undescribed state called “USE ALARM” looks like this:

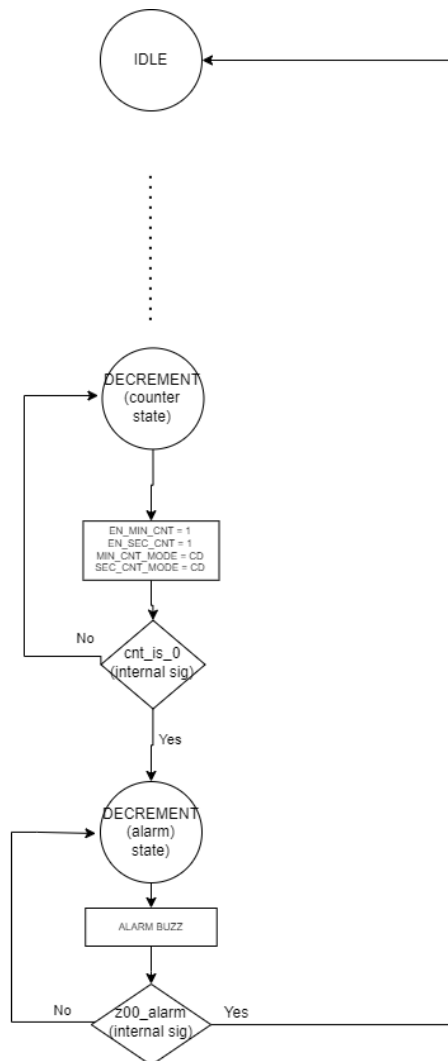


Figure 6.2 State diagram of the alarm

## 2.2.5 Detailed diagram of the project

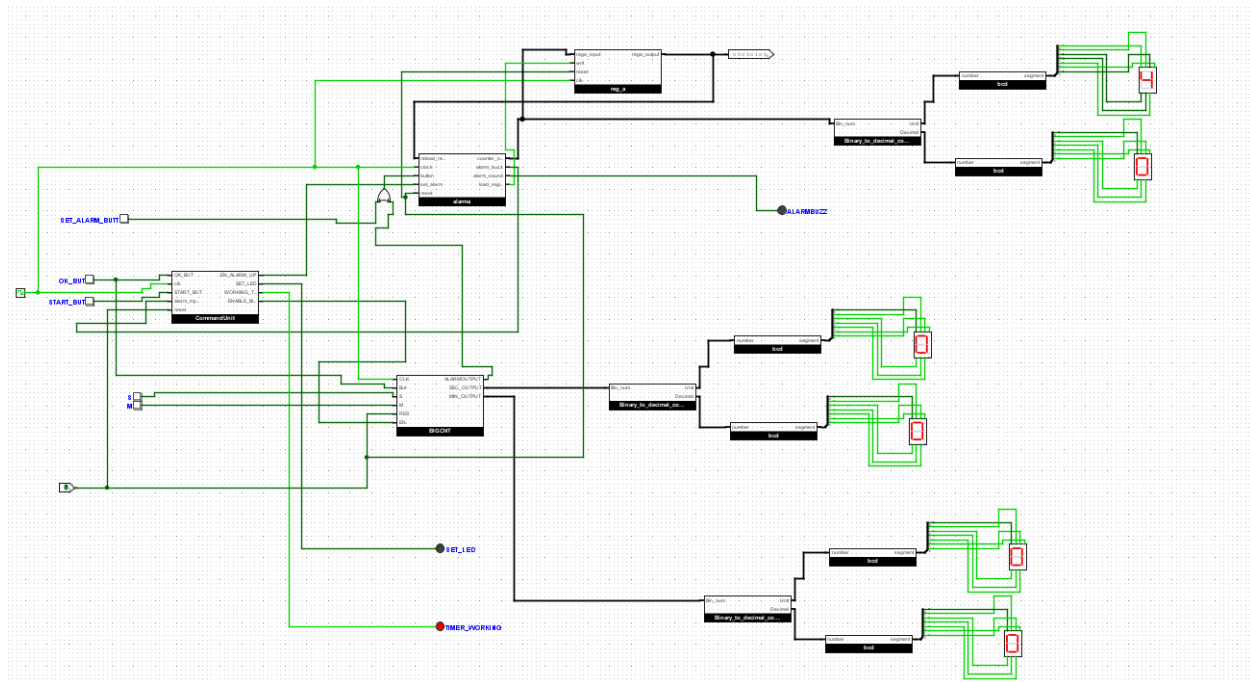


Figure 7 Detailed diagram of the project as shown in Logisim

## 3 User manual

This user manual provides an overview of the FPGA-based timer. The hardware used is a Nexys 4 FPGA. In order to use the timer as expected, a user would use the buttons in the following order:

- 1. Firstly, press the START button in order to start the automata. The SET\_LED led should be lit.
- 2. You are now expected to set the alarm. Press the SET\_ALARM\_BUT button and wait until the desired number is displayed on the SSD. Pressing the SET\_ALARM\_BUT button again will stop the alarm counter.
- 3. Here, the TIMER\_WORKING led should be lit. Pressing the START/STOP button will make the state transition as described in the state diagram. Pressing START/STOP again will make the „big” counter count up. Pressing it again will make the counter count up again, and so on.
- 4. In any state, pressing M or S works as described before. Pressing S once makes the seconds go up (same for M, it makes the minutes go up). Pressing S again makes the seconds timer stop (same for M). Now, when pressing again the START/STOP button, the timer goes into a count-down state. When it reaches 00:00, the alarm activates and buzzes for the set time in the beginning.

- 5.Further, if wanted, the timer can be used again starting with step 3(once the alarm resets itself to the value the user chose at step 2) or reset and make it back to step 1.

## 4 Technical justifications for the design

For this project, I've selected a set of 8 resources. The hardest unit to design was the big timer(composed of 2 interconnected counters: seconds and minutes) . I've decided to simulate the cascading of the 2 counters(when seconds are at 60, the minutes counter gets +1, and the seconds reset) synchronous. All other connections were made in a easy-to-understand manner, making it easy for new users to catch up with how the system works. Processes are triggered by CLK and the state transitions are determined by the user input. Following the requirements (see Specifications), the system is designed in a top-bottom manner and fulfills the needs of the task.

## 5 Future developments

Here, I've noted some ideas that struck my mind when working at this project:

- Enhancing the precision, such as implementing milliseconds
- Multiple timers functionality – develop the automata in such a way it could work with more set times(therefore, a user wouldn't have to reset the timer to a given value whenever it has to count down for something)
- Data saving – maybe it'd be interesting if the user could set some time stamps during the countdown state, e.g. in a running race, to be able to press a button that sets the time stamps according to the arrival of contestants.

## 6 References

[Nexys 4 - Digilent Reference](#)

[VHDL Tutorial – 19: Designing a 4-bit binary counter using VHDL \(engineersgarage.com\)](#)

[\[FPGA Tutorial\] Seven-Segment LED Display on Basys 3 FPGA - FPGA4student.com](#)