EEET2475 – Advanced Digital Design 1

Assignment (35%)

# A. System 1 - Stopwatch Design with Finite State Machine Technique (10%)

## 1.1 Description

In this exercise you will design a simple stopwatch system using SystemVerilog. You can check how such a system works using a Stopwatch app on your phone or on the website.

The system includes the following I/Os:

|  | Signals |
| --- | --- |
| One button for both Start/Stop | **start\_stop** (1 bit) |
| One button for both Lap/Reset | **lap\_reset** (1 bit) |
| Eight of 7-segment LEDs to display the total time elapsed (4 digits) and lap time (4 digits). We will track **the precision to 1/10 second.** | **HEX0** [7:0]  **HEX7**[7:0] |
|  | **rstn** – synchronous active low |
|  | **clk** – 50Mhz |

*I use the buttons and 7 segments in the above table, so it is easier for you to relate with an actual stopwatch. However, we won’t have to do FPGA prototpye in this system. Therefore you should focus on using the signal name and length in the 2nd column.*

When you press Start/Stop button, the timer starts counting. The first 4 digits show the time elapsed. During the run:

* 1. If you press Start/Stop, the timer pauses, then:
* If you press Start/Stop button again, the timer resumes.
* If you press Lap/Reset button, the timer resets to 00:00
  1. If you press Lap/Reset button, the lap time is recorded and displayed on the other 4 digits. We will record only one lap time, so the previous lap time recorded if any will be erased. The timer still runs and displays on the first 4 digits.

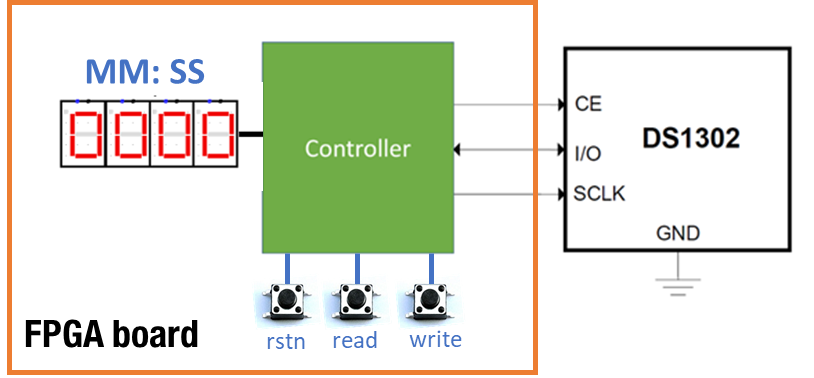
## 1.2 Requirements

* You need to derive a state diagram from the above description.
* You must design the system using Finite State Machine approach in SystemVerilog
* Verify the system using an auto-testbench.

# B. System 2 - Clock system with TimeKeeper Chip (25%)

## 2.1 Description

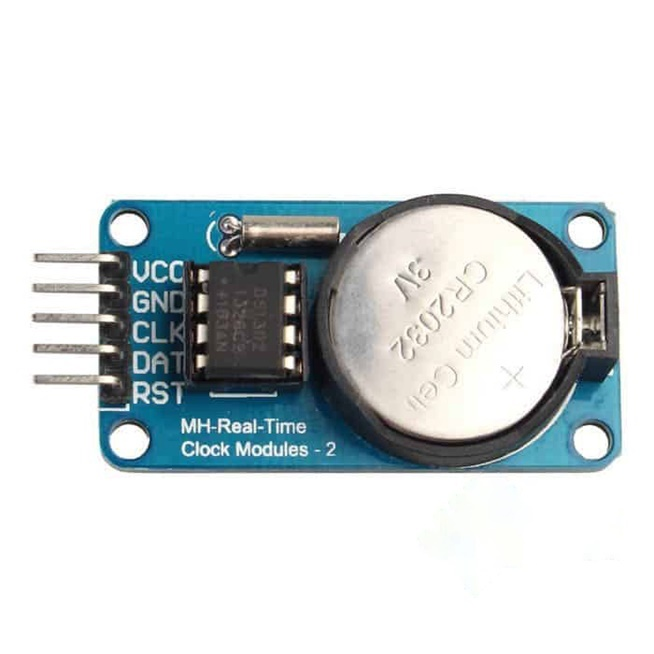
You are required to design a Clock system and implement it on a FPGA board. The system diagram can be shown below:



*Figure 1 – Clock System*

The clock relies on a Timekeeper DS1302 to keep time (battery-powered) even as the power is disconnected. We can operate your clock as follows:

* We will write some initial time to the DS1302 module at the beginning of the operation – *In this assignment, we will write* ***a fixed time (for example 13:24:30)*** *that you define in your SystemVerilog design. We don’t have to worry about changing it during the operation*. Everytime we press the **write button**, the write process starts.
* Once the timekeeper is loaded with a time, we can read this time from the module by pressing the **read** button. The time will be then display on 4 seven-segment LEDs on the FPGA board. We will display Minutes and Seconds in this Assignment. *Note the clock is not running, nor update the time at all until you press the read button another time.*
* The system can be reset by pressing the button **reset** button. After reset, the digits shows 00:00



*Figure 2 – TimeKeeper module with DSC1302 chip built-in*

*Datasheet -* <https://datasheets.maximintegrated.com/en/ds/DS1302.pdf>

## 2.2 Technical Requirements

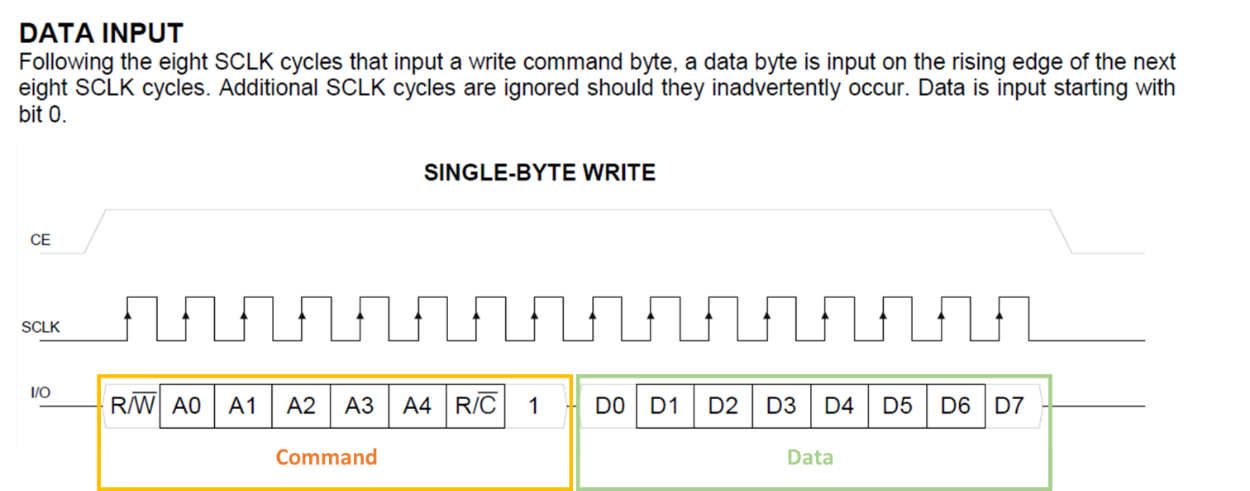
Your main task in this Assignment is to design and verify a Controller that can communicate with the DSC1302, as well as with buttons and LEDs.

The controller interfaces with the DS1302 via synchronous serial communication. The interface has three wires **Chip Enable (CE)**, **I/O** (data line) and **SCLK** (serial clock). The Controller need to send control signals and data to the DS1302 module for two major working modes:

* Controller writes data to the DS1302 to set the time for the chip (e.g., could be the initial time)
* Controller reads data from the DS1302 (this is the time the module keeps track of)

### 2.2.1 Write to the DS1302 module

We will use a **single-byte Write mode** which can be described below (you can read this information in the datasheet). Everytime the write button is pressed, the signals will be generated following the timing diagram for this mode



The data in this case will be our initial time, that you can hard code in your System Verilog file in this Assignment. The command will be corresponding with the mode.

In term of detailed timing parameters for the Write mode, you can find them in the below figure and table:

Diagram, engineering drawing

Description automatically generated

Table

Description automatically generated

To start the design of this mode, you should study the working mode together with the above information. Then you can come up with how you generate control signals accordingly.

### 2.2.2 - Read from the DS1302 module

Similarly, we will need to implement the **Single-Byte Read mode** which is summarized below. Everytime the read button is pressed, the signals will be generated following the timing diagram for this mode

Graphical user interface, application

Description automatically generated

The data in this case is the time stored inside the DS1302 module. The command is corresponding with the control for the read mode.

The detailed timing diagram for this mode is also given in the datasheet.

Diagram

Description automatically generated

To start the design of this mode, you should study the working mode together with the above information. Then you can come up with how you generate control signals accordingly. Finally, the data will be decoded to be shown on 7 segment LEDs.

## 2.3 Other Notes

* Button will be sampled only on the rising edge of the clock
* Reset is synchronous active low

## 2.4. Verification

You must verify the Controller above against the requirements especially for both Write and Read modes. Auto-testbench is required for top-module only. Sub-modules you can verify using manual testbench if possible.

## 2.5. FPGA prototyping

Finally implement the whole system on FPGA and test out the function of the clock

# C. Demonstration

During week 13, we will arrange a demonstration session. During the demo, you will be asked to show the working prototypes, answer questions regarding the two systems.

# D. Assessment

Marking Rubric is found on Canvas

**Workload**

| Group of 2 | Group 3 |
| --- | --- |
| All the above | ALL the above  Burst mode  Input the initial time |

## Burst mode

Besides the single write and read more, the DS1302 supports the burst-write of the data into the module. Group of 3 is required to explore and implement this mode together with the single-write above.

## Input Initial time

As FPGA has other I/Os such as buttons or switches. Group of 3 is required to choose these peripherals, and implement additional decode to allow users to input the initial time, instead of using a fixed initial time as in the Group of 2.