Alarm Clock with Buzzer

ECE 2700, Daniel Llamocca

Brendan Alkevicius, Samuel Bejko, Andrew McGhee, Trey Plichta

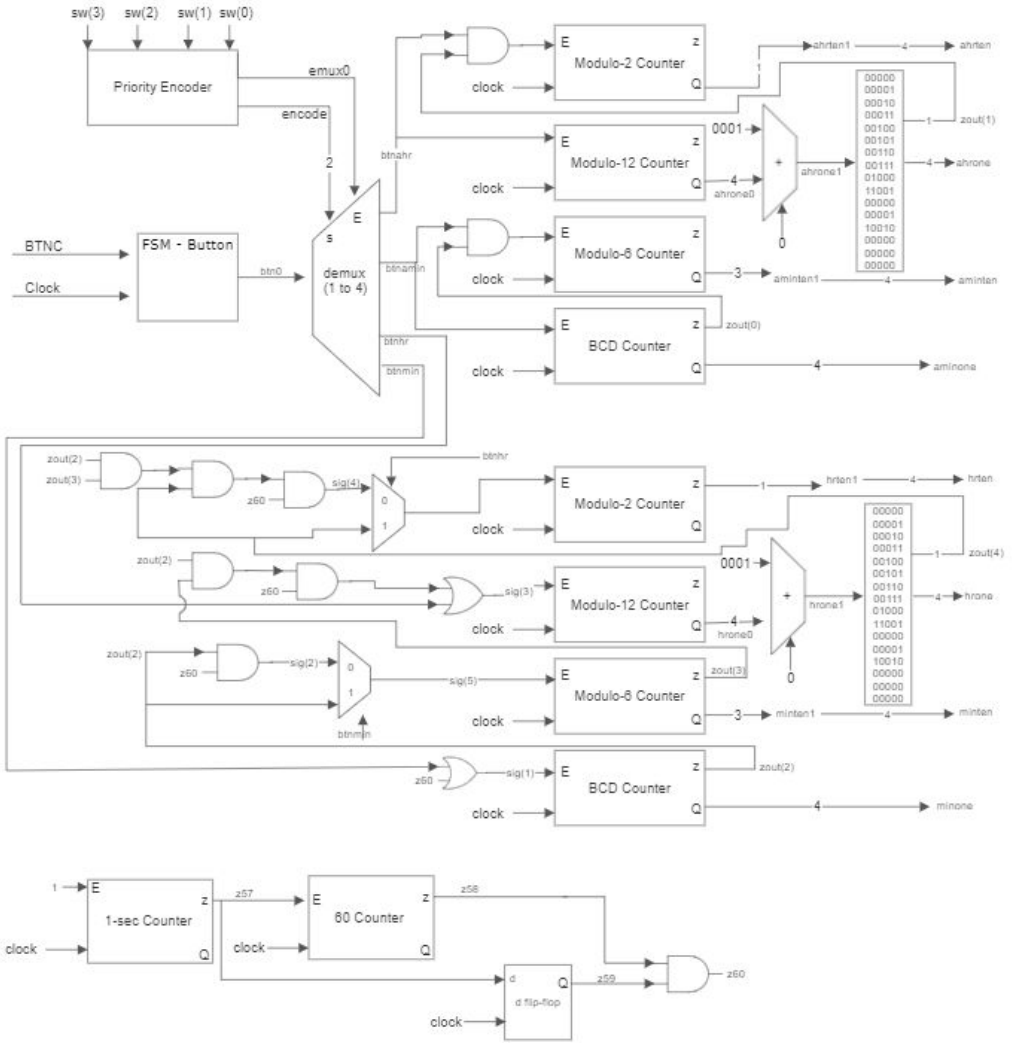
Our design aims to create an alarm clock using 8 separate 7 segment displays. We utilized FSMs, adders, decoders, encoders, multiplexores, Hex-to-7-segment decoders, comparators and LUTs to list a few. By using switches we are able to select whether we want to modify the time clock or change/set the time for the alarm. A button press is utilized to increase the time in either the clock or the alarm time. Once, the alarm time equals the clock time, a buzzer is triggered. The buzzer is deactivated by a button press or when 5 minutes has passed.

Main Circuit

Inputs: btnc and sw0 - sw3 on the fpga board, clock, resetn

Outputs: ahrten, ahrone, aminten, aminone, hrten, hrone, minten, minone which are all 4 bits and represent the number which is to be displayed on each of the eight 7-segment displays

Purpose: “The brains of the alarm clock”. This is the portion of the circuit that keeps track of the clock time and keeps track of the alarm time. It also takes input from the user to set the time based on which switches are activated.

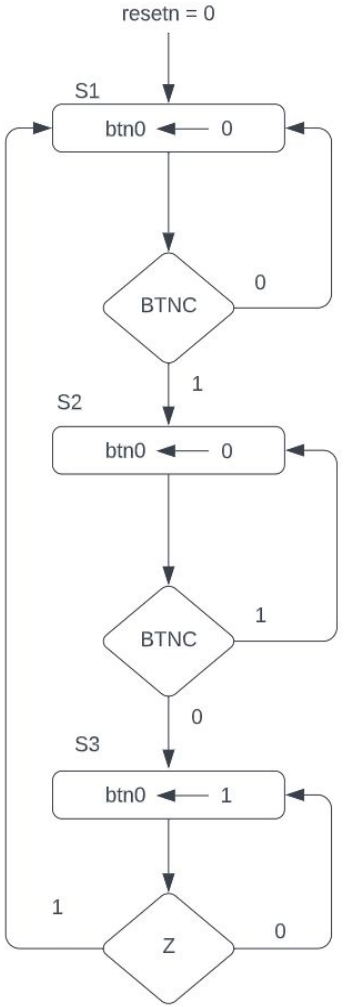


FSM #1 Button Input

Inputs: BTNC, clock(internal signal z), resetn

Outputs: btno

Purpose: This is the portion of the project that receives user input in regards to setting the clock and alarm times. This FSM has three states. First, it is waiting for btnc to be pressed. Once btnc is pressed, then it is waiting for it to be released. After btnc is released, the FSM is going to output a one clock cycle pulse to the rest of the circuit with the name btno before returning to state 1 and searching for the next instance where btnc is pressed.

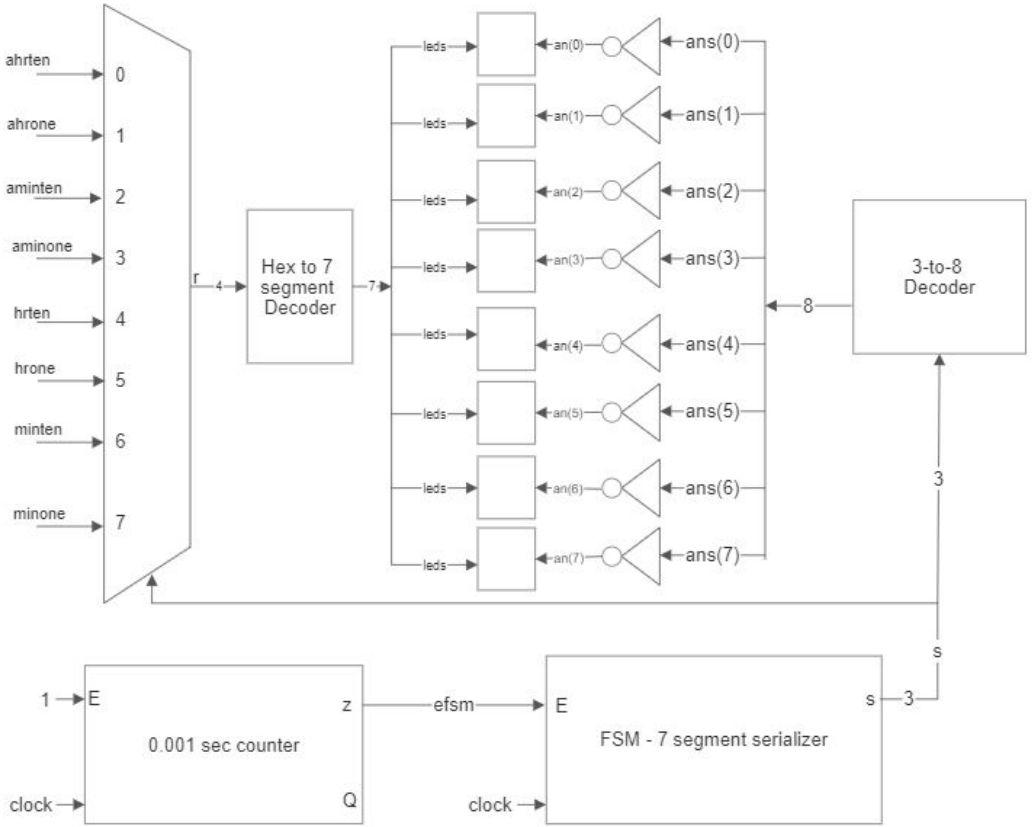


7-Segment Serializer Circuit

Inputs: ahrten, ahrone, aminten, aminone, hrten, hrone, minten, minone, clock, resetn

Outputs: leds(8 bits), an(8 bits)

Purpose: This part of the circuit takes the binary numbers that need to be displayed on each 7 segment display and filters through them every 0.001 sec with a control FSM in order for the human eye to perceive a constant output on each 7-segment display. This is a necessary workaround since the 7-segments display do not run in parallel. The common selector ‘s’ selects both the incoming data as well as deciding which display to turn on.

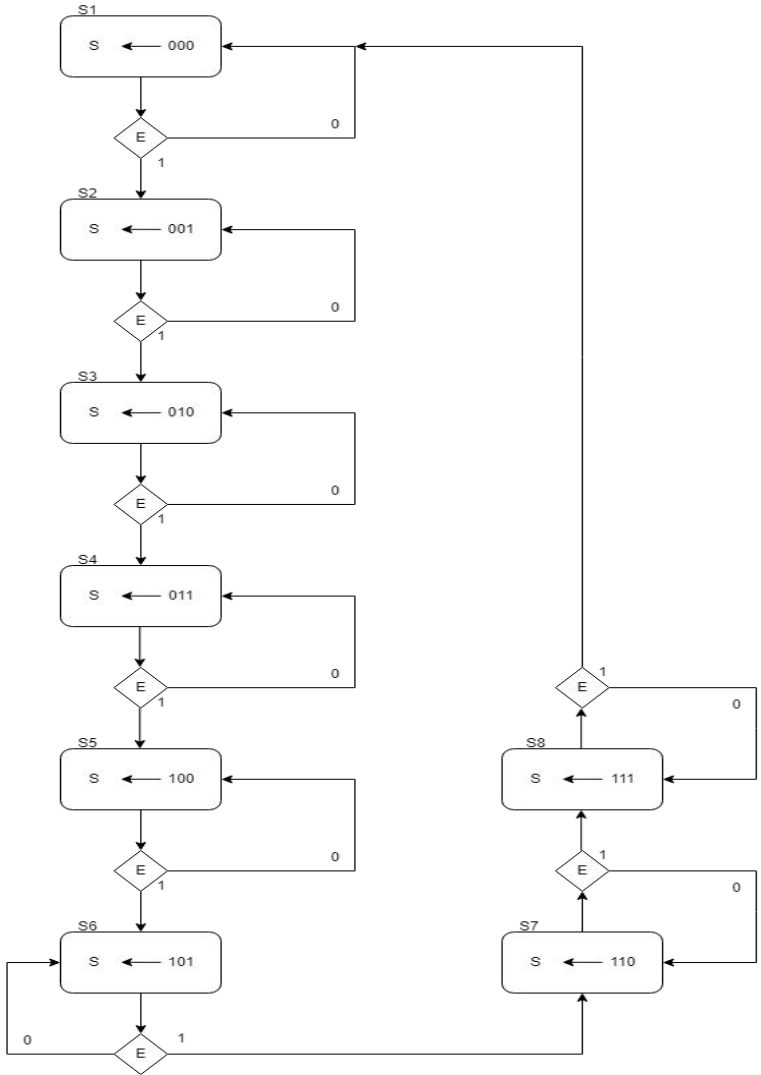


FSM #2 - 7 Segment Serializer Control

Inputs: efsm(internal signal E), clock, resetn

Outputs: s(3 bits)

Purpose: This FSM simply increments to the next state everytime E is 1. E comes from the z output of a counter, therefore, E is 1 when that counter hits it maximum value (every 0.001 seconds). Therefore, this FSM changes between each state every 0.001 seconds and the output s is increasing by 1. s is then provided to the multiplexor as a 3 bit select line and to a 3-to-8 decoder in order to complete the circuit.

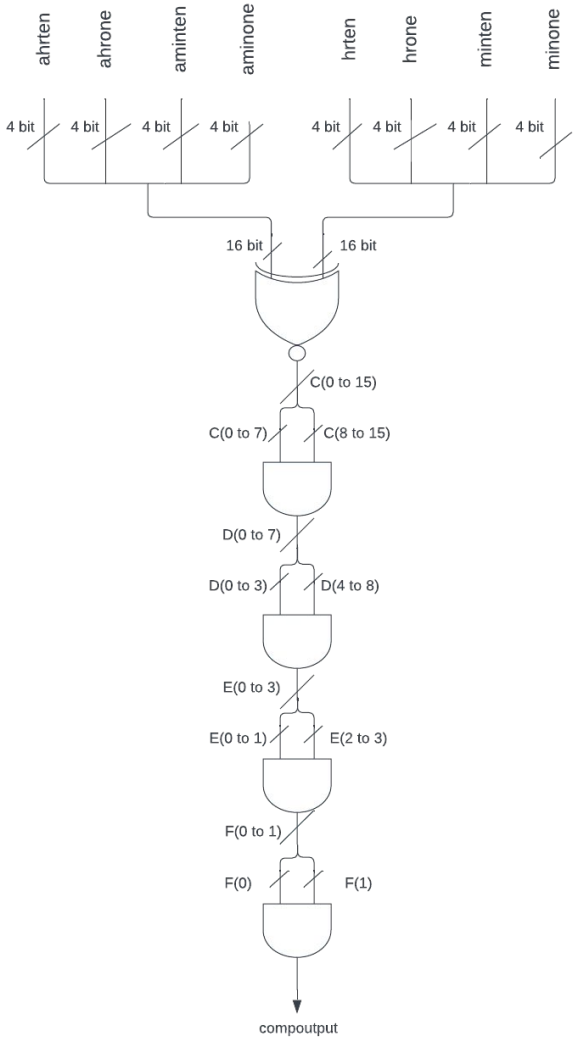


Comparator

Inputs: ahrten, hrten, ahrone, hrone, aminten, minten, aminone, minone (4 bits each)

Outputs: compoutput (1 bit)

Purpose: The comparator takes the current time and the set time for the alarm, and compares the two, triggering when they are both the same.



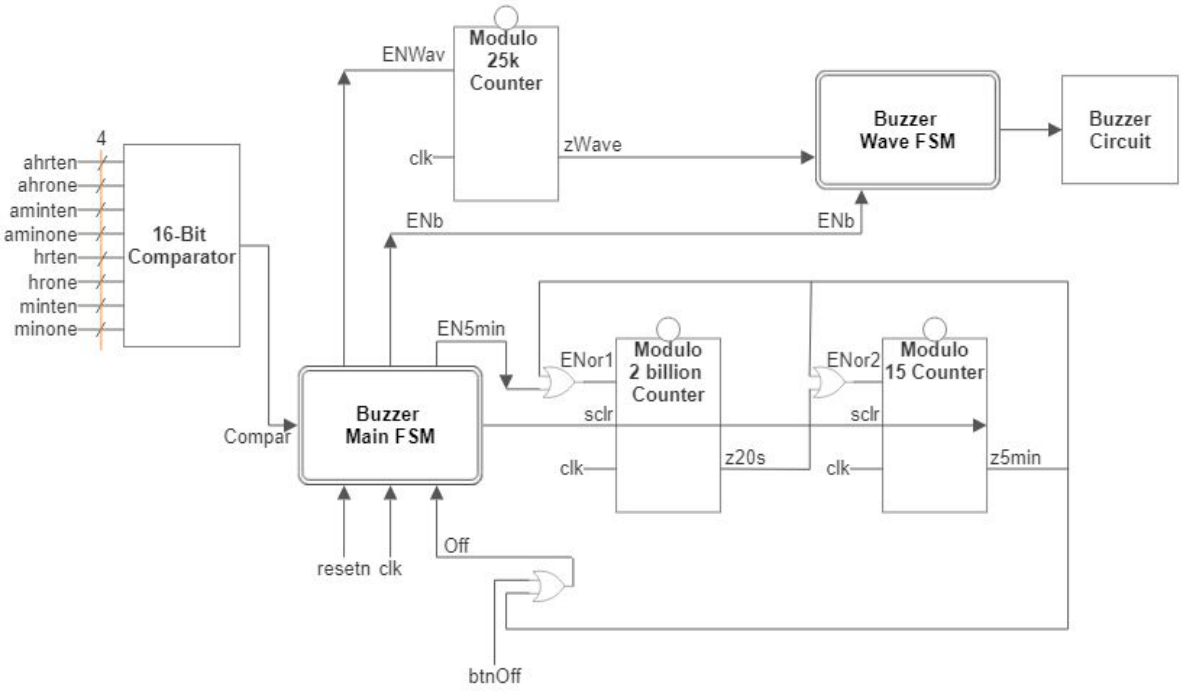
Buzzer Circuit

This is the circuit representation for the activation of the buzzer.

Inputs: Comparator output, btn, resetn, clk

Outputs: Buzzer going off.

Purpose: the circuit utilizes the FSM outputs and counters to use as a buzzer activation system.

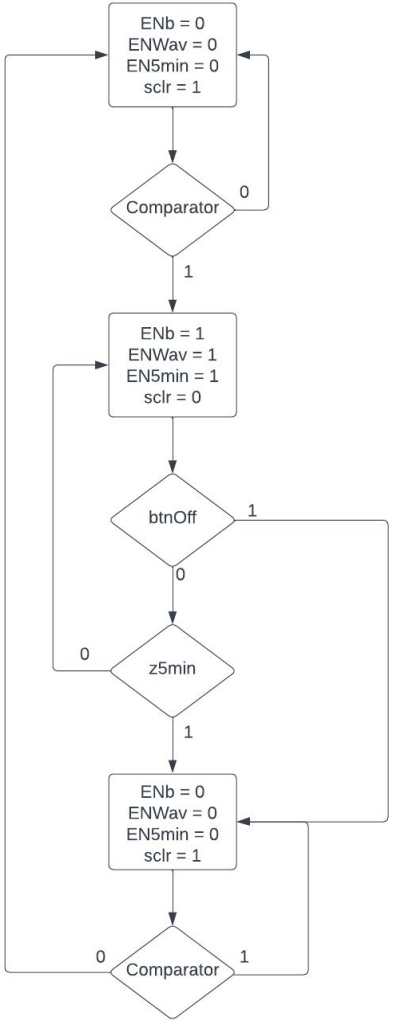


FSM #3 Buzzer Control

Inputs: Comparator, btnOff, z5min, clock, resetn

Outputs: ENb, ENWav, EN5min, sclr

Purpose: This FSM controls whether the buzzer will go off or not by utilizing the comparator, btnOff, and z5min



FSM #4 Wave Generator

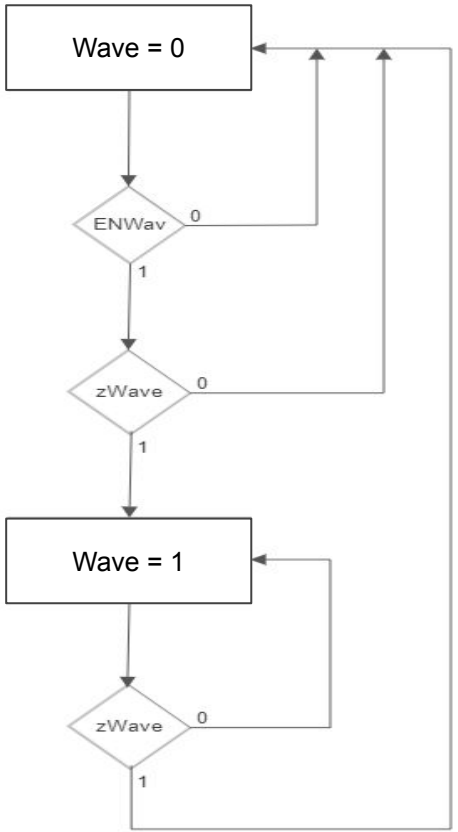
Input: -”ENWav” from Buzzer Control FSM

-4kHz 10ns pulse “zWave” from Modulo - 25k counter

-resetn

- clock

Output: -2kHz full square wave to the buzzer circuit “Wave”



Problems Encountered

- Cascaded Counters keeping the final output at a value for longer than one on board clock cycle
    - A d flip-flop was used to delay the z output of the first counter, that was then sent through an and gate with the z output of the second counter
        - This allowed the output of the and gate to be high when both of the counters hit there max value and then held this for one onboard clock cycle
- How to get a counter for the ones digit of the hour for both the alarm time and the clock time since the one digits value increments as follows:
    - 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, reset
    - Additionally, properly resetting the hours tens digit counter was a major issue
        - To solve all these problems, a 4 bit adder and a 4-to-5 LUT were used