Programming and Designing a Microwave Controller

Brian Lam, Michael Tolmajian, Sarah Su, Michael Nava
lam@cpp.edu, matolmajian@cpp.edu, sssu@cpp.edu, mnava1@cpp.edu
The Department of Electrical and Computer Engineering
College of Engineering
California State Polytechnic University, Pomona

***Abstract*—This project introduces a very common and popular piece of technology used in everyday life: a microwave. Any microwave today includes basics such as a number pad to set a timer, a kind of visual for both the timer and heating status, and a start/stop button. Most modern microwaves include a power/heat level button as well as various preset cooking settings. Here, we explore all the different ways to implement the functionality of a microwave into the hardware we are currently using, an FPGA board.**

I. Introduction

In this project, we are building a controller that is capable of doing nearly everything a modern microwave can. Using a total of 4 buttons, we allow users to start/stop the microwave heating process, cycle upwards/downwards through the three power levels, and reset the heat timer. This project also utilizes 16 LEDs, and 6 seven-segment displays to visualize the current status of the microwave, such as whether or not it is heating, what its power level is, and the timer. As for the timer, we use 12 switches to set a specific time up to 63 minutes and 63 seconds.

Unconventional to a normal microwave, our controller is capable of connecting to a VGA monitor to display an image of a microwave that is ON and OFF, as well as the ability to be started/stopped by an outside peripheral (a computer and keyboard) through serial communication.

II. Requirements

For this project, the group uses both programmable logic on software to output results to hardware devices.

A. Software

The entirety of the microwave controller exists on Xilinx's Vivado Design Suite using the Verilog Hardware Description Language. To activate the six peripherals in use, the group uses Verilog.

B. Hardware

Because the project focuses on not only using buttons, switches, and diodes but also VGA and UART, additional hardware requirements are considered. While the project uses Verilog on Xilinx Vivado, the output is on Digilent's Nexys A7-100T Field Programmable Gate Array (FPGA). For the Video Graphics Array, the project also requires an external desktop monitor for display. For the UART data transmission, the Nexys A7 must have a USB-JTAG adapter for the user to transmit and receive data through the computer's I/O devices. The adapter can connect one USB port on a computer, a laptop for instance, to the JTAG port of the Nexys A7.

III. Methodology

A. Buttons

This design uses a total of 4 buttons. One for cycling upwards through the power levels, one for cycling downwards through the power levels, one to stop or start the microwave, and a last button to reset the entire machine.

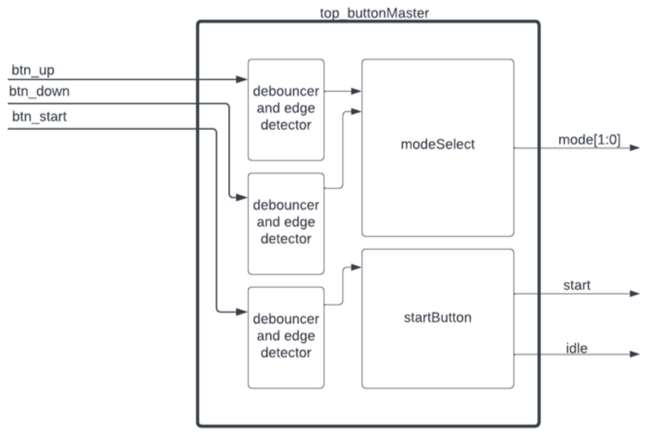
top_buttonMaster
btn_up
btn_down
btn_start
debouncer and edge detector
debouncer and edge detector
debouncer and edge detector
modeSelect
startButton
mode[1:0]
start
idle

Fig. 1: Master design of the button modules.

Figure 1 shows the overall design of the button's mechanics. The design is relatively simple. Using a debouncer, synchronizer, and edge detector, an ideal button can be created. This ideal button can then be used in a Mealy-type finite state machine that will jump from one mode to another based on which button is pressed. If the cycle up button is pressed, the mode will jump up one level based on which level it is currently at (normal mode is default).

In Figure 2, the 'btn_start' signal, when in sync with the system clock, 'sys', initiates the microwave's timer by raising the 'start' variable to 1. Notice that there are other moments

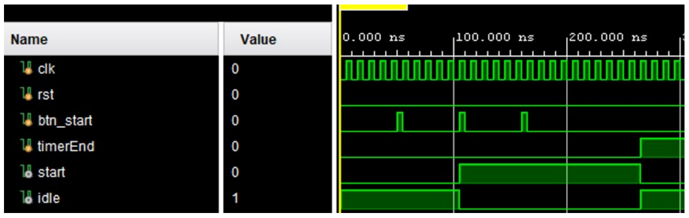

Fig. 2: Simulation of button debouncer.

in the simulation, where 'btn_start' was set to 1 but had no effect on the status of the 'start' signal. Synchronization, in this case, requires both the clock edge and button signal edge to be rising at the same time.

B. Light-Emitting Diodes (LEDs)

From the available 16 light-emitting diodes on our Nexys-A7 FPGA, we had used all of them for this project. The 16 LEDs serve as outputs of a four-to-sixteen decoder controlled by a 4-bit up/down counter.

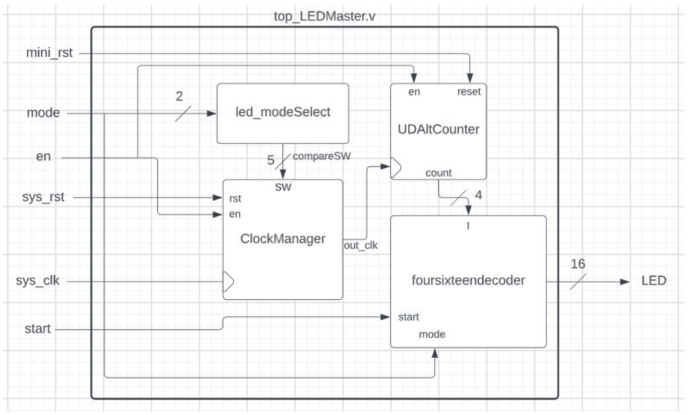

Fig. 3: Master design of the LED modules.

The top module features four different modules: the speed-select, the clock manager, the up/down alternating counter, and the four-to- sixteen decoder. In the speed-selecting module, using our 2-bit mode array input, we choose one of three 5-bit binary combinations: 10101_2, 10100_2, and 10011_2. In decimal, they are 21, 20, and 19, respectively. One of these three numbers would then act as a 5-bit select for the multiplexing portion of the clock manager module.

The clock manager features the 32-bit counter and the 32-to-1 multiplexer. If our system clock frequency is set to 100 Megahertz (MHz), then the frequency of the 19th bit alternating between 0 and 1, for instance, would be 100 / 2^{19} MHz. The frequency at the 20th bit would be 100 / 2^{20} MHz, half the frequency of that of the 19th bit. For that reason, the frequency of the 21st bit would be half the frequency of the 20th bit and also a quarter the frequency of the 19th. By having our multiplexer select as either 19, 20, or 21, we are choosing one of the frequencies at the 19th, 20th, and 21st bits of the 32-bit count.

In a casual up/down counter, we have an enable, reset, and up/down switching input. If we set our up/down switch to 0, the counter would count up. If the switch rises to 1, the count will go down to 0. However, it would usually be a switch that we control. In the up/down "alternating" counter, we don't control the up/down switch. Since the counter is 4-bits, the range goes from 0 to 15 in decimal. If we initialize our up/down switch to 0, the count will rise to 15. However, when 15 is reached, the up/down switch will automatically set itself to 1 and countdown to 0. When the count falls back to 0, the up/down switch switches to 0, getting the count to rise back up. In this sense, we can call this an up/down alternating counter.

Figure 4 demonstrates the concept of the up/down alternating counter, as the simulation displays a staircase pattern. As soon as the last element of the LED array is signaled on, the pattern begins to invert and fall back to the first element. The endless rising and falling edges on the top of Figure 4 are that of the clock generated by selecting a certain mode.

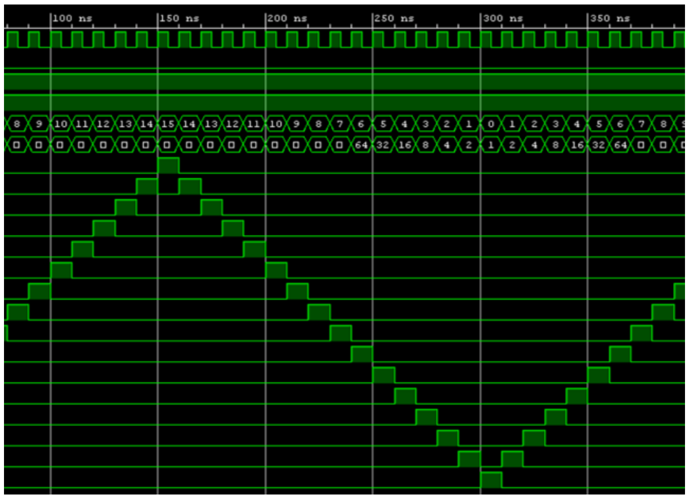

Fig. 4: Simulation of the up/down alternating counter.

The four-to-sixteen decoder, using 16 case statements, will take in the counter's output and determine which of the LED array indices to light up on the FPGA. If we view this on our Nexys A7 FPGA, we can see the one diode sliding from the right end to the left and back to the right, in alternating manner.

C. Switches

We utilized a total of 12 switches for our project. Six are designated to control the minutes, and the other six control the seconds.

The switches represent six bits in binary, whereas a switch being on represents a '1', and a switch being off represents a '0'. These values are then put into a load counter, which is used to set the desired time the user wants to start the microwave before it begins counting down.

The load counter is then connected to a countdown timer that decrements from the starting value given in binary by the switches. The countdown will count and pause at the whim of a button and utilizes a 1 MHz clock outputted by separate clock divider. The countdown decrements the 'seconds' every

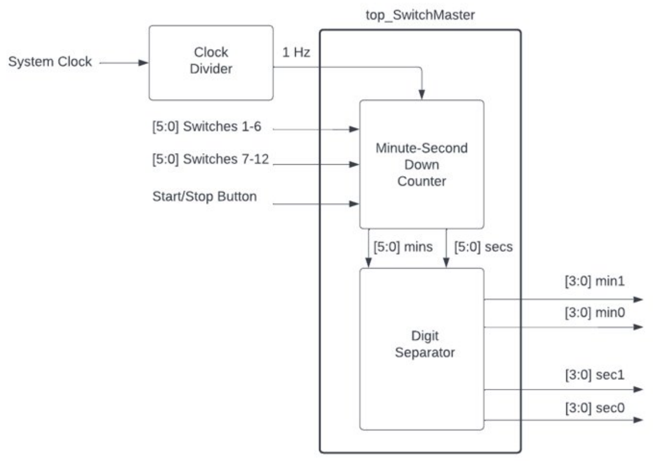

Fig. 5: Master design of the switch modules.

second and 'minutes' every minute; it stops decrementing once they reach 0. However, 'seconds' loops back around if there is at least one minute left. This countdown timer then connects to a digit separator which separates the ones and tens place of both the 'seconds' and the 'minutes'; this is achieved by dividing the seconds and minutes by 10 to get the tens place, and doing modulus 10 to get the ones place.

D. Seven-Segment Display

There are a total of eight seven segment displays on the FPGA. We utilized two to display the microwave's instantaneous mode, and four to display the timer that counts down.

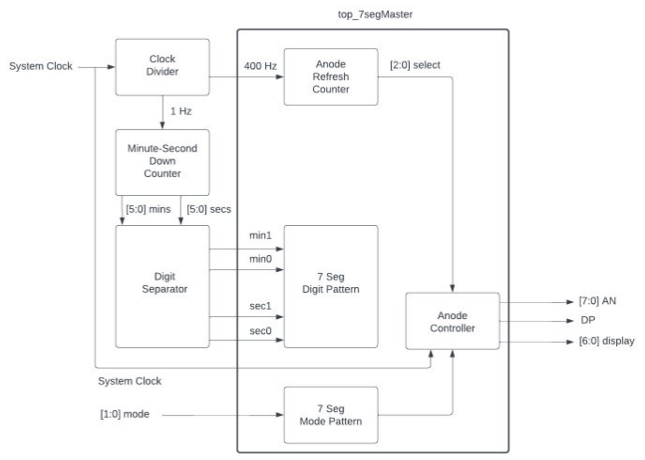

Fig. 6: Master design of the seven-segment display modules.

In order to achieve this, a clock divider was created that produces a 1 Hz and 400 Hz clock rate. The 1 Hz clock signal is used to control the speed at which the timer counts down, and the 400 Hz clock signal is used in the anode controller to control the refresh rate of the seven segment displays.

Figure 7 shows the names of each of the seven segments on the display. With this in mind, the group programs the modules to display certain characters. This is possible by creating a 7-bit array — one bit for each segment with Segment A holding the most significant. Additionally, turning a segment on requires setting the bit to zero. This is due to the segment displays being common cathode instead of common anode.

For instance, if the user wanted to display a '1', Segments B and C are set to 0, while the rest are set to 1. The 7-bit array would result in 1001111_2. Displaying an 'E' requires the array to be set to 0110000, and so on.

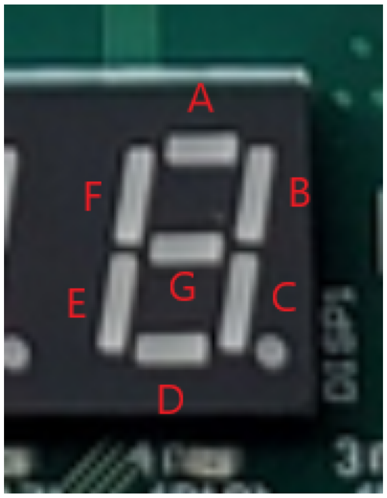

Fig. 7: The seven segments along with their allocated names.

There are a total of three modes to display. The modes will be indicated using the two most significant segment displays. The first mode is of a value 01_2, and it is here that the seven segment displays "L o"; the second mode has a value of 10_2 and displays "- -". Lastly, the third mode is the value of 11_2 and displays "H i".

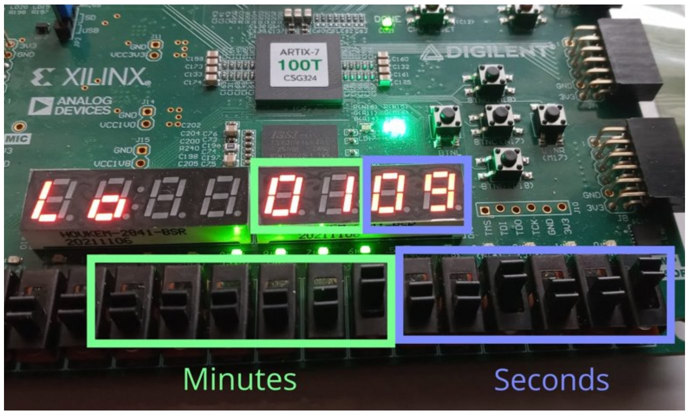

Fig. 8: Allocation of switches to their respective segment displays.

As Figure 8 shows, the four least significant displays are used for the timer representing minutes and seconds in binary-coded decimal (BCD). Of those four, the two on the right (in blue) represent minutes, and the two on the left (in green) represent seconds.

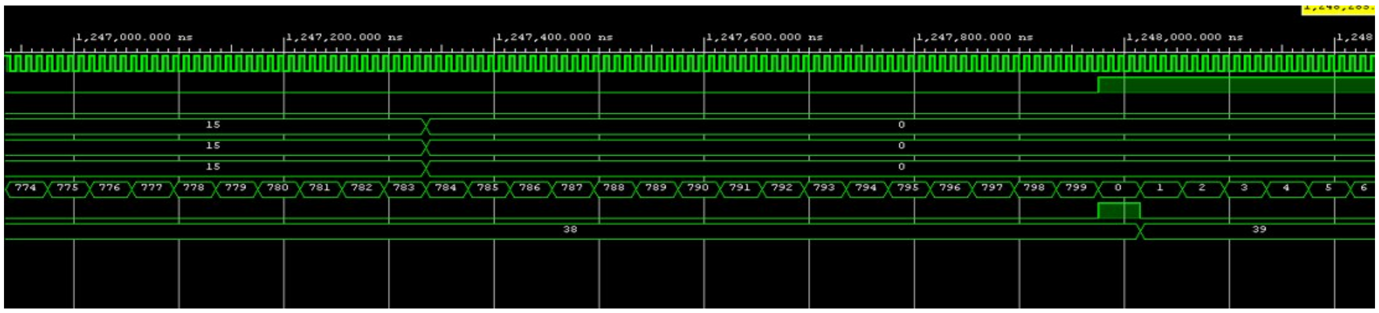

Fig. 9: Simulation of the video graphics array.

E. Video Graphics Array (VGA)

The VGA contains three modules: the horizontal counter, the vertical counter, and the top module. Both counters are simple up counters and depend on a 25.1 MHz clock. The horizontal counter counts up to 799 due to the number of pixels on screen, and after every cycle, it outputs an enable that allows the vertical counter to count up by a single increment. The vertical counter increments to 524 and has to wait for the horizontal counter to finish in order to increment up by 1, akin to a 2D array. This is how the screen is arranged and covered pixel by pixel.

Figure 9 demonstrates the concept of the the horizontal and vertical counters using Vivado. When the horizontal counter reaches 799, the simulation shows a one-cycle pulse to indicate an increment of the vertical counter. The vertical count in this instance incremented from 38 to 39.

The top module instantiates the horizontal and vertical counters. It also contains a 3 bit RGB register to control the desired color to be displayed via an always block. Another vital aspect of this module is a 25 MHz always block that consists mostly of if and else statements to control the VGA design output. These calculated values of the horizontal and vertical counters design the microwave image chunks at a time.

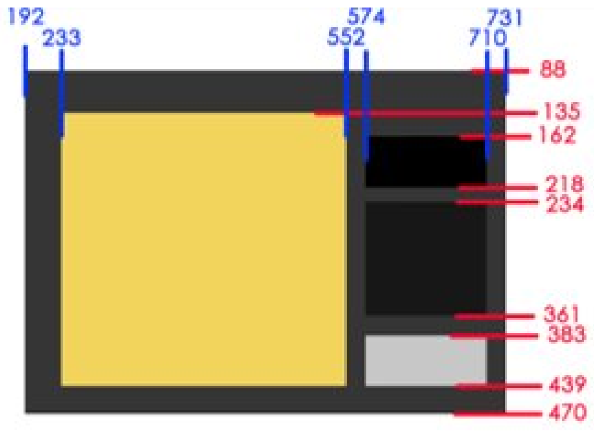

Fig. 10: Dimensions of displaying virtual microwave.

To sketch the simple, virtual microwave, the group was required to measure specific dimensions and boundaries, as shown in Figure 10. The microwave shown for example in Figure 10 is the state of the microwave when the 'start' condition is true. For the 'idle' condition, the microwave door would show gray instead of yellow.

F. Universal Asynchronous Receiver-Transmitter (UART)

The UART communication used in this project is relatively simple. Since we are only using the keyboard from the computer to communicate with our FPGA board, only the receiver module is required.

Upon any keyboard input, the receiver will output a pulse the length of one clock cycle. This will thus, act like its own button and start or stop our machine.

IV. Roles and Acknowledgements

Brian Lam had worked on programming the logic of the Nexys A7's buttons and video graphics array. Michael Tolmajian had worked on programming the logic of the light-emitting diodes as well as the UART. Lastly, Sarah Su and Michael Nava collaborated on functioning the switches and seven-segment displays.