Design Description

The aim of this lab was to build a stopwatch. To build this stopwatch, we created 5 different modules, each of which are described below. The first module is the clock\_divider, which takes in the hardware clock, and generates 4 other kinds of clocks. The second module is the counter module, which stores the value that needs to be displayed on the seven-segment display. The third one is a modify display module, which controls the blinking of the clock when we’re in adjust mode, and the fourth module is the display module that displays the digits on the seven-segment display. The final module is the overall module that instantiates each of these smaller modules and passes in the correct signals from one module to another.

Clock\_divider.v

|  |  |
| --- | --- |
| Input | Output |
| * 100 MHz clock (clk) * Reset signal (rst) | * 1 Hz clock (one\_hertz\_clk) * 2 Hz clock (two\_hertz\_clk) * 400 Hz clock (fast\_clk) * 4 Hz clock (blink\_clk) |

The aim of this module is to create clocks that run at different frequencies, using a clock that runs at an extremely high frequency. We did this by creating counters for each of the different kinds of clocks and then incrementing each of these at the positive edge of the fast\_clk. Each of the counters, counts up to a different value, corresponding to their respective frequencies, and outputs a clock signal of the required frequency. An example of how we figured out the counter value of the 4 Hz clock is shown below:

Given a 100 MHz clock, that is a signal that is high 10^8 t

counter.v

|  |  |
| --- | --- |
| Input | Output |
| * 1 Hz clock (one\_hertz\_clk) * 2 Hz clock (two\_hertz\_clk) * 400 Hz clock (fast\_clk) * Pause signal (pause) * Reset signal (reset) * Adjust signal (adj) * Select signal (sel) | * Minutes Ten’s digit (minutes\_tens) * Minutes One’s digit (minutes\_ones) * Seconds Ten’s digit (seconds\_tens) * Seconds One’s digit (seconds\_ones) |

The aim of this module is to maintain the value of the counter for the stopwatch. We maintain 4 different registers, each corresponding to a digit, to store the value of that digit. Once we hit 9 on a one’s digit or a 5 on the 10’s digit, we increment the digit to the left of that. In the case that the reset signal becomes high, we reset all the digits to 0, and start counting again.

The module uses the 1 Hz clock to increment the counter when the stopwatch is in normal mode. In the case that the adjust signal is high, we want to increment the counter at a frequency of 2 Hz. We do this by using combinational logic, based on the adjust signal to select between the two clocks, and then increment the counter accordingly. When the adjust signal is high, we either increment only the seconds counters or the minutes counters based on the select signal. For instance, if the adjust and select signals are high, we want to only increment the seconds values at a 2 Hz rate. At the positive edge of the clock, we check if the seconds digits are 5 and 9, then we reset it to 0s. Otherwise if the one’s digit is at a 9, we reset it to 0 and increment the tens digit. If neither of these cases is true, then we just increment the one’s digit for the seconds. It works similarly for adjusting the minutes when the adjust signal is high, and the select signal is low.

Pause Signal talk about debouncing l

modify\_display.v

|  |  |
| --- | --- |
| Input | Output |
| * Minutes Ten’s digit (minutes\_tens) * Minutes One’s digit (minutes\_ones) * Seconds Ten’s digit (seconds\_tens) * Seconds One’s digit (seconds\_ones) * Blinking Clock (blink\_clk) * Adjust signal (adj) * Select signal (sel) | * Minutes Ten’s digit (o\_minutes\_tens) * Minutes One’s digit (o\_minutes\_ones) * Seconds Ten’s digit (o\_seconds\_tens) * Seconds One’s digit (o\_seconds\_ones) |

The aim of this module is to display the blinking of the clock correctly, which is the case when the adjust signal is high. The underlying algorithm for this display blinking works as follows.

If the adjust signal is low, we output the digits as we receive them, without modifying them. In the case that adjust is high, we set the digits to their actual value for one clock cycle (of frequency blink\_clk), and set it to a dummy value (10) to signify that they should be off. We do this using a state machine. The FSM changes state at the positive edge of the blink clock, switching to and fro between the states of display\_state and off\_state. In the case that select is high, we want to switch off the minutes digits and blink the seconds digits. If at the positive edge of the blink\_clk, the current state is display state, we output the actual seconds values to o\_seconds\_\* signals, and the dummy value 10 to the o\_minutes\_\* signals, and set the current state to off\_state. When we’re in the off\_state, we set all the outputs to 10, such that it switches off all the digits. This works similarly for the case when select is low, swapping the o\_minutes\_\* and o\_seconds\_\* signals. This creates the illusion of the digits blinking at the blink\_clk frequency.

display.v

|  |  |
| --- | --- |
| Input | Output |
| * Fast Clock (fast\_clk) * Minutes Ten’s digit (minutes\_tens) * Minutes One’s digit (minutes\_ones) * Seconds Ten’s digit (seconds\_tens) * Seconds One’s digit (seconds\_ones) | * 7 segment display value (segments) * Which digit to illuminate (enables) |

The aim of the display module is to take in 4 digits and display them on the seven-segment display decoder. The inputs it takes in are the 4 digits (2 for minutes, and 2 for seconds) and a fast clock. The fast clock is used to cycle through the states displaying different digits so that the illusion of the digits being displayed together is created. We do this using a state machine. The state machine has 4 different states, each one corresponding to a digit. At the positive edge of this fast clock, the state machine moves to the state to display the next digit, working from left to right.

To display each individual digit, we created a module called convert\_to\_segments that takes in the digit as a decimal, and outputs the values of each of the seven segments. Furthermore, to set the digit, we need to output the correct enables signal, which is a one-hot signal. For instance, if we’re trying to display the minutes\_tens\_digits, we set the enables signal to 0111, so that only the first digit is illuminated on the actual seven-segment display. The first digit in this case will be illuminated with the segments value that we output.

The one special case is that we encoded the digit value 10 to correspond to all offs. Any time a digit is to be displayed as a 10, it means that the entire seven-segment display for that digit should be off.

stopwatch.v

|  |  |
| --- | --- |
| Input | Output |
| * Hardware Clock (clk) * Reset (reset) * Adjust (adj) * Select (select) * Pause (pause) | * Seven segment display output (segments) * Enable signal for dispaly (enables) |

The stopwatch module is the overall module that takes in inputs from the hardware, and outputs the correct digits on the seven-segment display. The stopwatch instantiates all the previously described modules and pipes the outputs correctly. It first instantiates the clock divider module. The 1 Hz, 2 Hz and the fast clock are sent in to the counter module, along with the reset, adjust, select and pause signal. This module then generates the 4 decimal values for the seven-segment display. These digits are sent to the modify\_display module. The output from the modify\_display module remains the same, unless the adjust signal is high, in which case it makes the seconds or minutes blink as described above. The digits outputted by the modify\_display module are sent to the display module. The only task performed by the display module is to generate the correct segments and enables output.