**Swinburne University of Technology**

**Computer Systems**

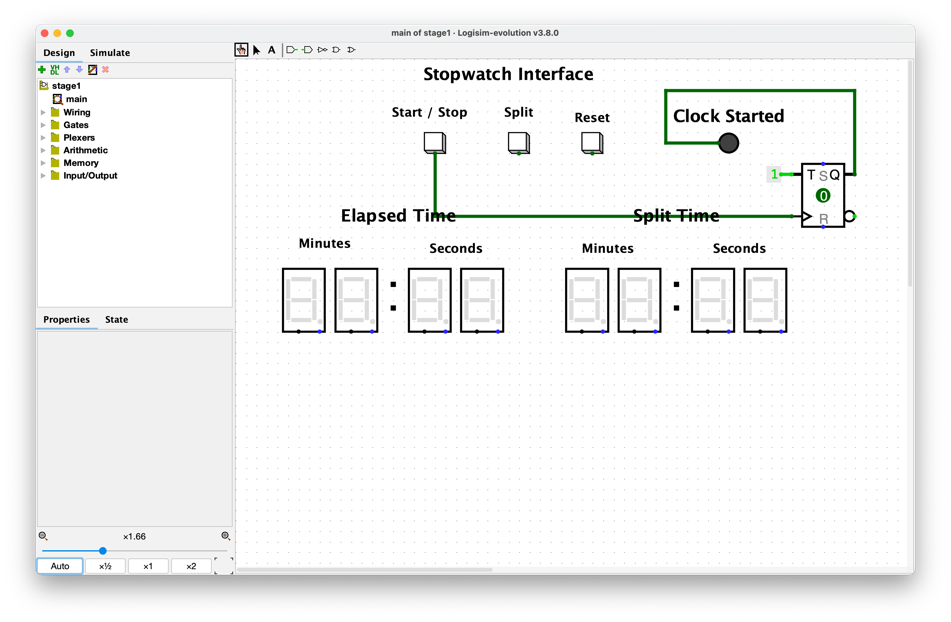
**COS10004**

**Assignment 1 – Digital Stopwatch**

**Marco Giacoppo**

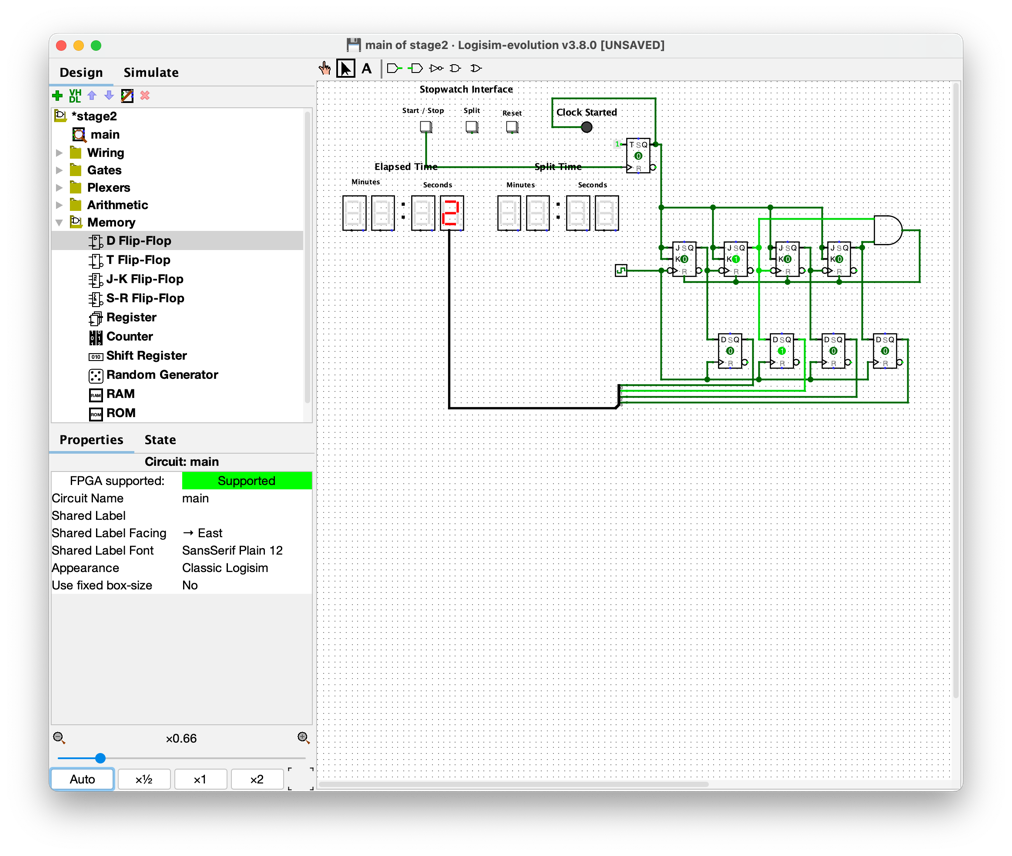
**104071453**

**Stage 1:**



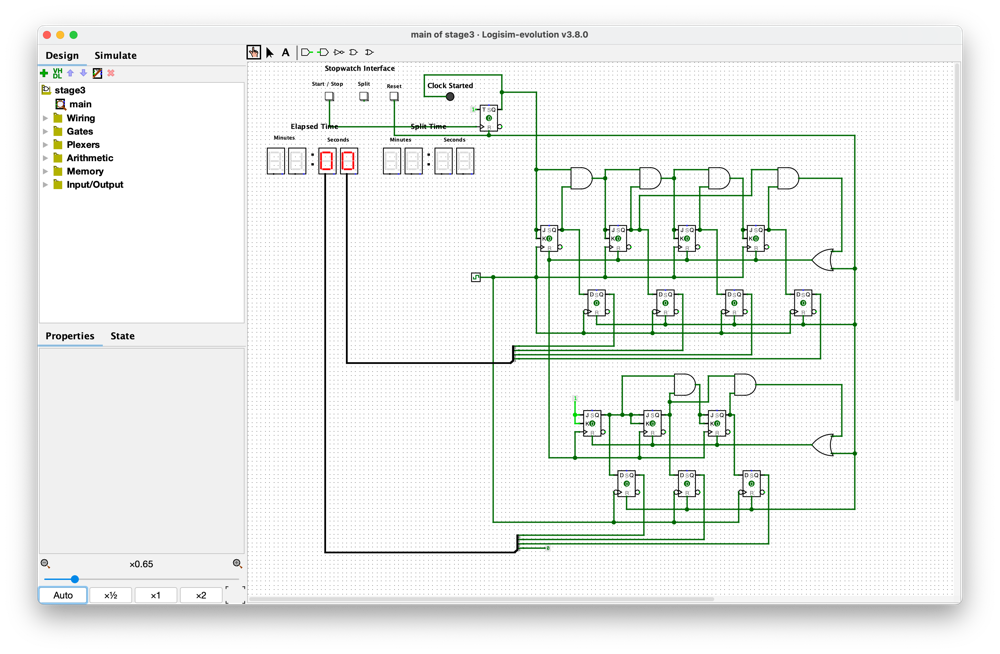
So here I used the T flip flop to store the signal using a constant. That way we can click the button once and the stopwatch is on it’s on state, and if we click it again, it will turn the stopwatch off.

**Stage 2:**

****

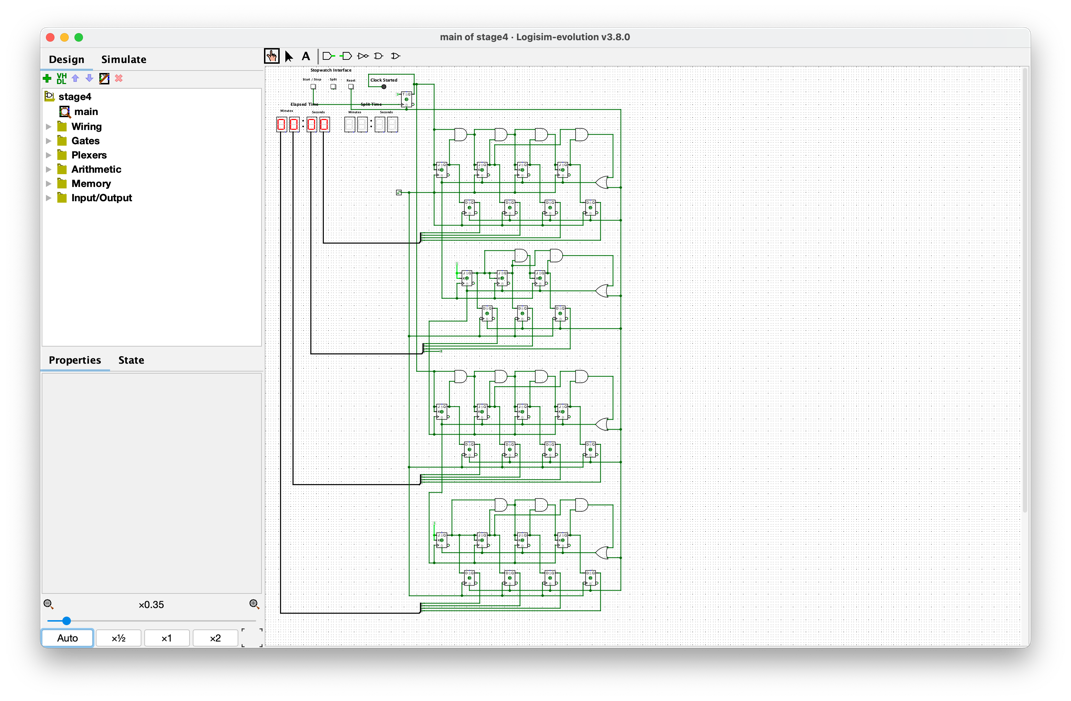
Here in stage 2, I built a 4-bit ripple counter using JK Flip-flops. I used an AND gate to make sure we don’t enter an illegal state. By taking the output of the second and last flip flop and putting it in the AND gate, I used the signal input to send a signal to all the flip flops to reset them. That way all the flip flops will reset itself when it reaches the number “9”.

**Stage 3:**



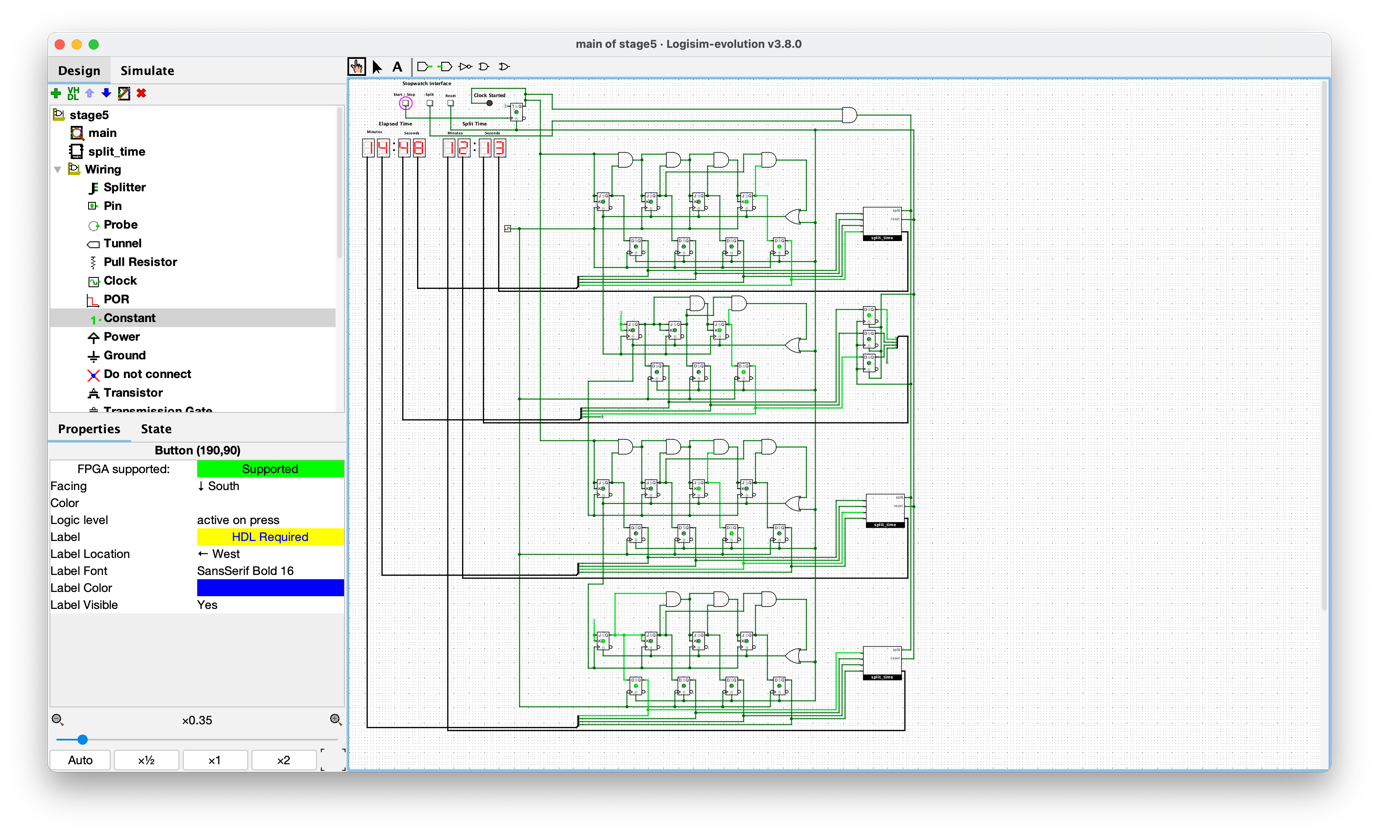
In stage 3, I used the same method as in stage 2, but now as we are implementing the reset button, I used an OR gate to reset the flip flops. The way the minute part work is that it receives the signal whenever the first set of flip flops (the second) are reset. By using a constant, it allows the flip flop to have an input and only turns on when it receives signal from the upper part of the flip flops.

**Stage 4:**



In stage 4, I used the same techniques as in stage 3 so there’s no difference in the circuits except here the minutes go from 00 – 99 then resets back to 00 as asked in the assignment.

**Stage 5:**



Here in stage 5, I managed to implement the split button using a series of D flip flops which is stored in a subcircuit called split\_time. The flip flop takes the signal from the output signal of the set of circuits on its left (the minutes and seconds). The split time’s flip flop input is connected to the output of an AND gate which is connected to the “Clock start” signal which allows the stopwatch to only split the time when the clock is running and not when it’s in its stop state.

**Stage 6A:**

Diagram

Description automatically generatedText

Description automatically generated

Here in stage 6A, I used a subcircuit in order to make the main interface neater. So, the way it works is I made a series of D flip flops to memorize the previous split times. By taking the first set of split time, it memorizes the binary bits and converts it into number in the hex digit display. By connecting the subcircuit to the previous outputs, it memorizes it first and when there’s a “Split” signal from the button, it shows the previous recorded split time and shows it in “Split Time” display. This continues 5 times until there is no more room, making it forget the first time recorded and making room for the newest split time.

**Stage 6B:**

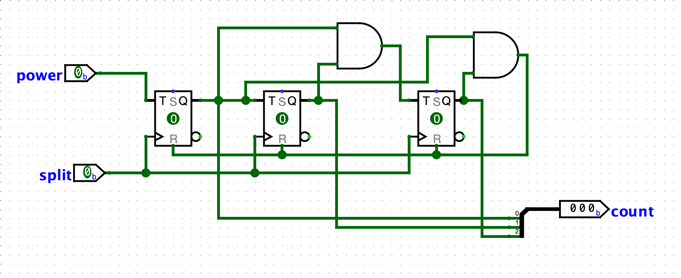


Figure 2: Counter

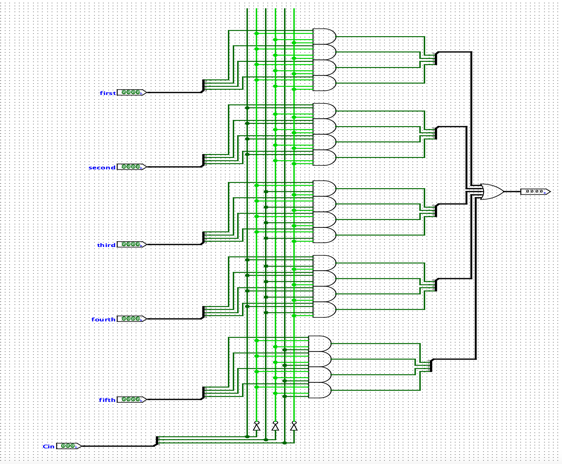


Figure 1: Multiplexer

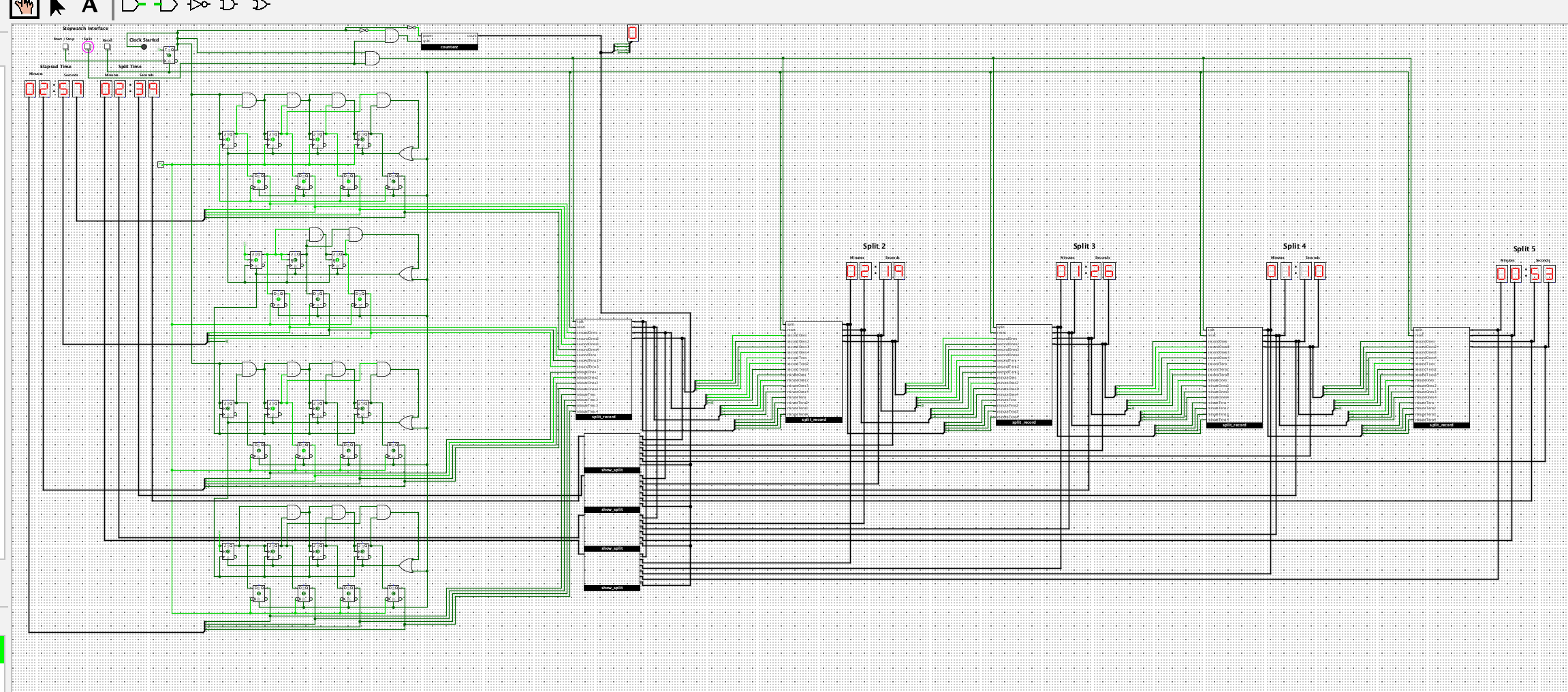


Figure 3: Main Circuit

Here in stage 6B, I think I managed to get the circuit working but there are some confusions I have. The circuit will work just fine if the counter stays in 0, which will store the “Split Time” in the display it’s supposed to be. The circuit will run normally if the counter stays in “0”, whenever the stopwatch is in the “Stop State”, it will loop the recorded Split from the most recent to the earliest and back again.

On figure 1, I made a multiplexer for each of the inputs which are secondOnes, secondTens, minuteOnes and minuteTens. Watching some tutorials on youtube helped me create this but I’m not sure if it’s 100% correct. Because I have 5 stored splits, I created 5 input and each input is 4 bits because that’s how many bits there are in the main counter. Each input then is combined into an OR gate which tells if one of the bit is on, it sends the signal.

On figure 2, I created a 3 bit counter to determine which line is chosen to be displayed into the Split Time display.

On figure 3, I took the outputs of the current split and 4 other stored split to put into each of the subcircuit that I created. This is done to combine all of the Stored Split and send the signal into the Split Time display.

I assumed this didn’t work as it intended to be because I did some mistakes either in the mux or the counter. Maybe even because I didn’t connect the right wires to the subcircuit input.

I didn’t understand the logic in stage 7, and because of it, I didn’t finish stage 7.