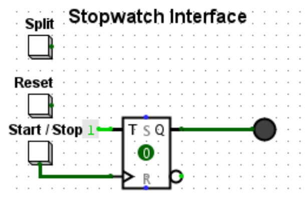
ASSIGNMENT 1 REPORT

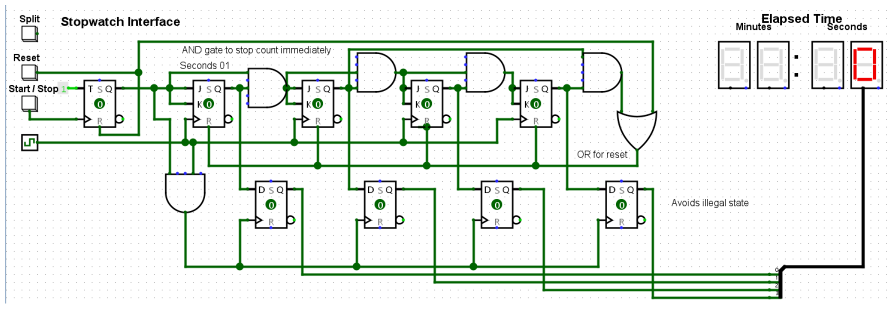
Anthony Tang, 103965555, COS10004, Tuesday 4:30PM

Stage 1). Implement the Start/Stop button



We need a T flip-flop circuit that can be toggled on and off by pressing a "Start/Stop" button. The CLK input of the flip-flop is connected to the "Start/Stop" button, so that a HIGH input on the CLK line will change the state of the flip-flop and output a 1 on Q. To ensure it stays In toggle mode, a constant 1 input is applied to the T input.

Stage 2). Implement a single digit seconds display



In order to achieve a clock with a decimal of 9 we need a 4 bit counter using J-K Flip Flop connected to a common clock.

To generate a clock with a decimal of 9, we need a 4-bit counter using J-K flip-flops connected to a common clock. The first step is to set up four J-K flip-flops, each representing one bit. We connect the common clock to all four flip-flops to ensure that they all operate on the same cycle.

Next, we input the T flip-flop output into the first J-K flip-flop. When the T flip-flop is toggled, the J and K inputs of the J-K flip-flop will receive a 1 input on the next CLK rising edge. The output of this J-K flip-flop is then connected to the 0 input of the splitter, representing the first binary digit; the next CLK cycle will then cause the Flip Flop to output a 0.

Additionally, We connect the output of the T Flip Flop and the first J-K Flip Flop to an AND gate. This ensures that when the stop button is pressed, no further CLK rising edges will cause the J-K flip-flop to propagate signals further to the next J-K Flip Flop when the "Stop" button is pressed.

The output of the previous J-K flip-flop is connected to the J and K inputs of the next flip-flop. When the clock input triggers a rising edge and the previous J-K input is 1, it outputs a 1. By using two J-K flip-flops, we can count from 01 to 11 or 1 to 3.

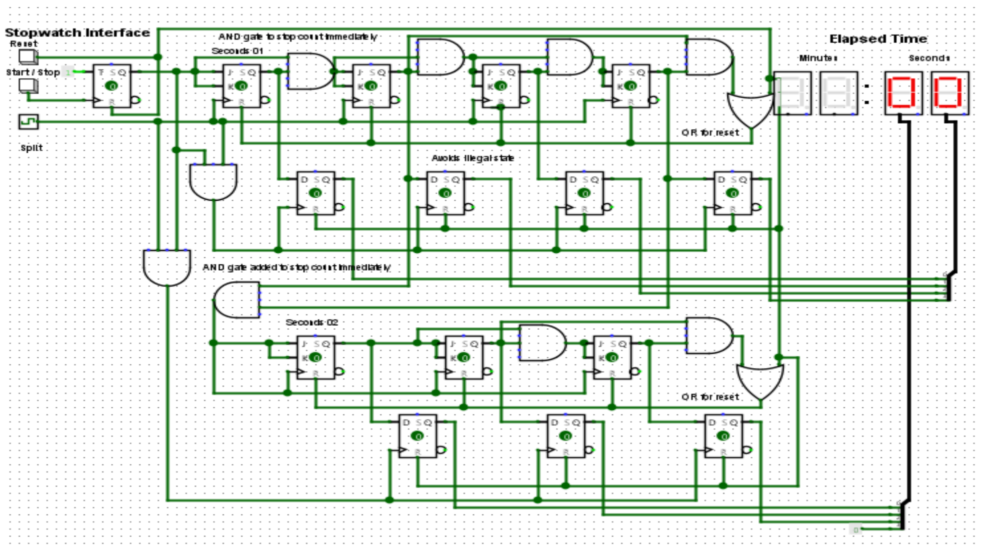
To generate the counting sequence, additional logic gates, such as AND gates, are used. The output of the second and first J-K flip-flops are then connected to an AND gate. This AND gate only outputs a 1 to the third J-K flip-flop when the first and second J-K are outputting a 1, The Third Flip Flop acts as the third binary bit. These additional gates allow the counting from 0001 to 0111 or 1 to 7. The output of the third J-K flip-flop is then connected to the 2 input of the splitter.

The output of the third and second J-K flip-flops is then connected to an AND gate, which is then connected to the fourth J-K flip-flop, representing the fourth bit. This is to prevent the signal from the third J-K flip-flop from causing the counter to skip 5, 6, and 7 (0101, 0110, and 0111) when it goes from 100 to 1000. Upon the eighth rising edge of the clock, the first, second, and third flip-flops will output a 0, and the fourth will output a 1, outputting an 8 in decimal when connected to the 4 input of the splitter.

The ninth rising edge will toggle the first J-K flip-flop, giving us the number 9 (1001). To stop at 9 and reset the counter back to 0, the output of the second J-K flip-flop and the fourth output are connected to an AND gate, which routes to all four flip-flop clear inputs. Upon the ninth rising edge, all flip-flops will reset to 0, giving us a 0 on the HEX digit display and counting from 0-9.

A buffer of D Flip Flops connected to the common CLK and outputs of the J-K Flip Flops is added to prevent the momentary illegal state of 10. An AND taking the outputs of the toggle and the CLK is connected to the reset of the buffer to stop the count immediately rather than 1 CLK cycle later.

Stage 3). Implement the full two-digit seconds display



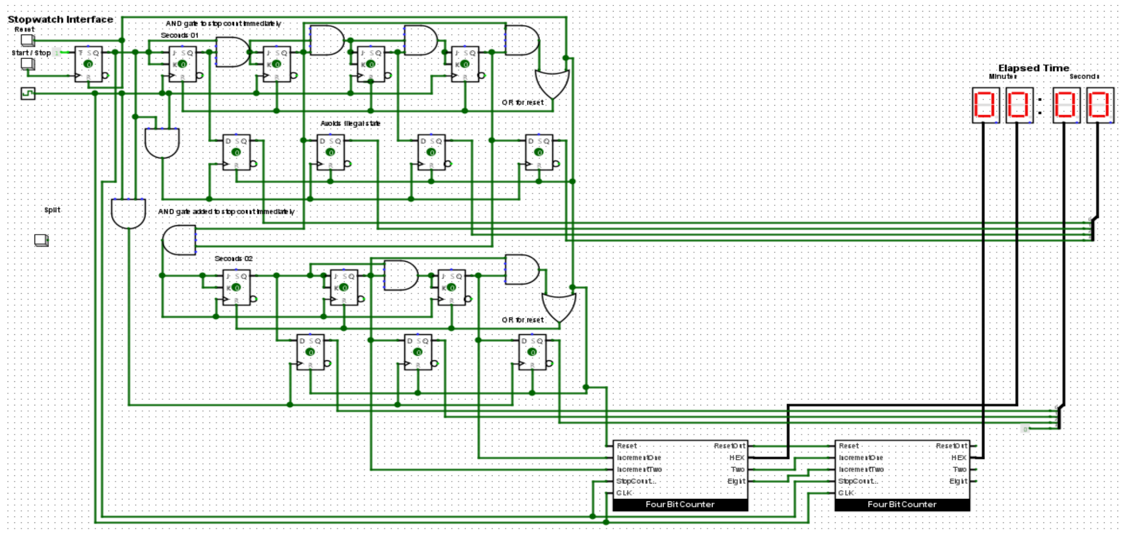
To expand the functionality of our timer, we need to add a second "seconds" HEX digit display. This can be achieved by incorporating the same logic as stage 2, with a few tweaks.

Since we only want to count up to 6 (or 5, in this case), we use only 3 J-K Flip Flops in this stage. Additionally, we want the 1st J-K Flip Flop to output a 1 only when the previous HEX Digit counts to 9. To accomplish this, we connect the outputs of the 2nd and 4th J-K Flip Flops from the previous synchronous counter to the inputs of an AND gate, and then route the output of the AND gate to the CLK and J/K inputs of the 1st J-K Flip Flop in the 2nd synchronous counter. Additionally, we connect the output of the 2nd and 3rd J-K Flip Flops to clear the Flip Flops after 5 of the 60th cycle.

This setup ensures that the 1st J-K Flip Flop of the 2nd synchronous counter outputs a 1 on the next CLK cycle, but only every 10th cycle, since the CLK inputs receives a 1 only every 10 cycles. Therefore, we are effectively propagating the next count every 10 cycles, allowing the counter to count from 1 to 5 and reset on the 6th cycle. This allows the timer to count from 0-60, which is exactly what we need for our "seconds" HEX digit display.

A constant 0 is applied to the 4th input as there are incompatible widths when the fan out option is 3. D Flip Flops connecting to the HEX display are used as buffers to prevent momentary 7. Similar to stage 2 an AND taking the outputs of the toggle and the CLK is connected to the reset of the buffer to stop the count immediately rather than 1 CLK cycle later.

Stage 4). Implement the Minutes display

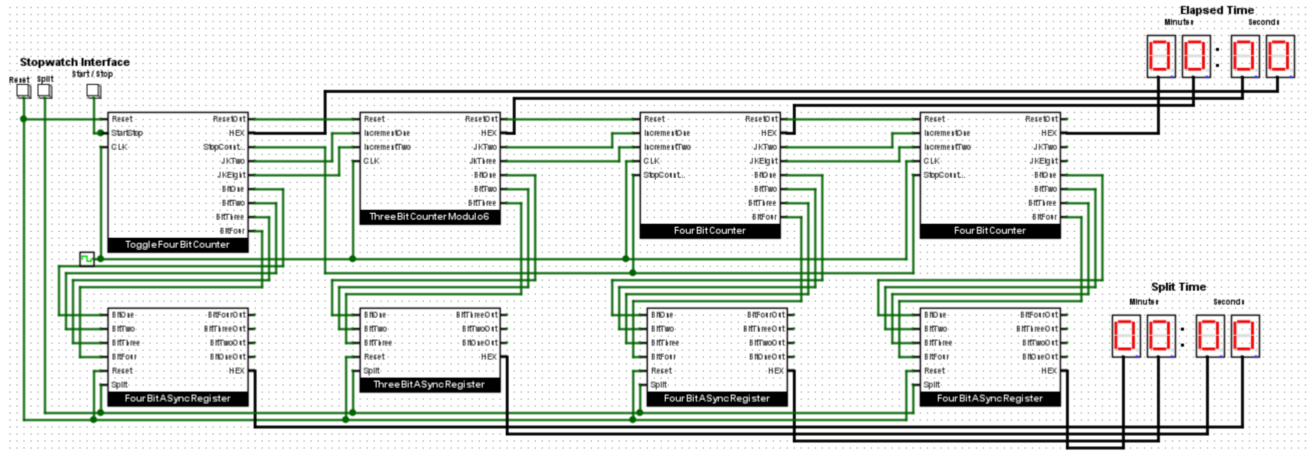


To extend the counter to track minutes, we use similar logic to what we employed in stage 2, we also implement subcircuits.

We create a synchronous counter subcircuit similar to stage 2 with 4 J-K Flip Flops. To ensure that this counter only counts when the previous synchronous counter counts up to 6, we use the outputs of the 2nd and 3rd J-K Flip Flops as inputs to an AND gate (IncrementInput on the subcircuit), which connects to the first J-K Flip Flop and the CLK of all 4 Flip Flops. This ensures that the Flip Flops only count or propagate their signals every 60th cycle. To limit the counting of this HEX digit to 9, we incorporate the counter reset mechanism we used in stage 2 after the 10th cycle. This resets the Flip Flops and outputs a 0.

For the second minute HEX digit display, we use the same “FourBitCounter” Subcircuit but we want it to count only every 600 cycles, or when the previous minutes synchronous counter cycles 10 times. Similarly to the first minute HEX, we input the 2nd and 4th Flip Flops from the previous synchronous counter to the “IncrementInputs” of the subcircuit , which routes to the input of the first J-K Flip Flop and all the CLKs of the other Flip Flops. This ensures that the counter propagates its signal only every 600th cycle, effectively counting from 0 to 9 for this HEX digit.

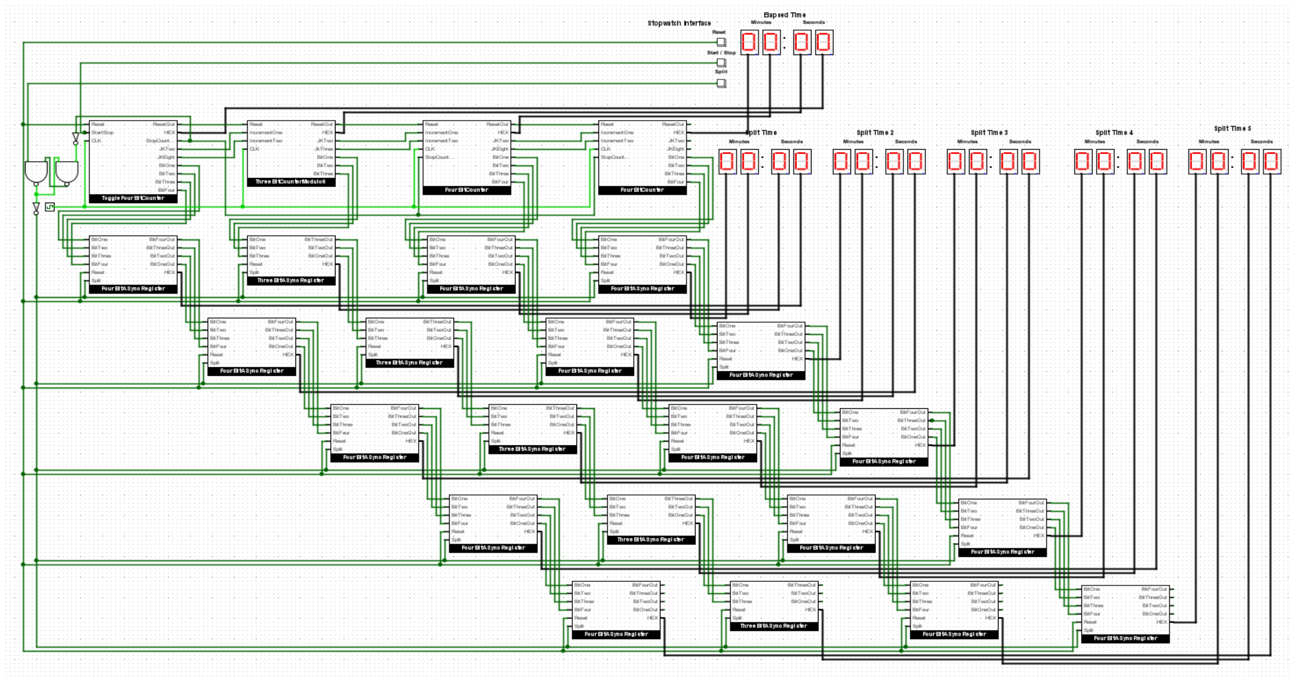
Stage 5). Implement the “split” button



In order to implement a Split function, we utilise D Flip Flops as asynchronous 4/3 bit registers. Each counter and each individual D Flip Flop output buffer are connected to a dedicated Register (We use subcircuit registers). The Split button is then connected to the CLK input of each D Flip Flop. Additionally, we have further implemented subcircuits to cut down on clutter of the wires. “BitOne” to “BitFour” or “BitThree” represent the D Flip Flop register outputs.

When the D Flip Flop of the counters outputs a 1 or a 0 and the Split button is pressed, the D Flip Flops will save the state of the counter at the time it was pressed. Each J-K Flip Flop bit has its own dedicated D Flip Flop that stores its state, and these D Flip Flop outputs are then connected to their respective HEX digit displays based on which bit they had saved. The top D Flip Flop is bit one, the bottom bit 3 or 4.To ensure that the Split time is reset when the reset button is pressed, They are connected to the Reset input of the D Flip Flops.

Stage 6a) Implement a “split” time recording

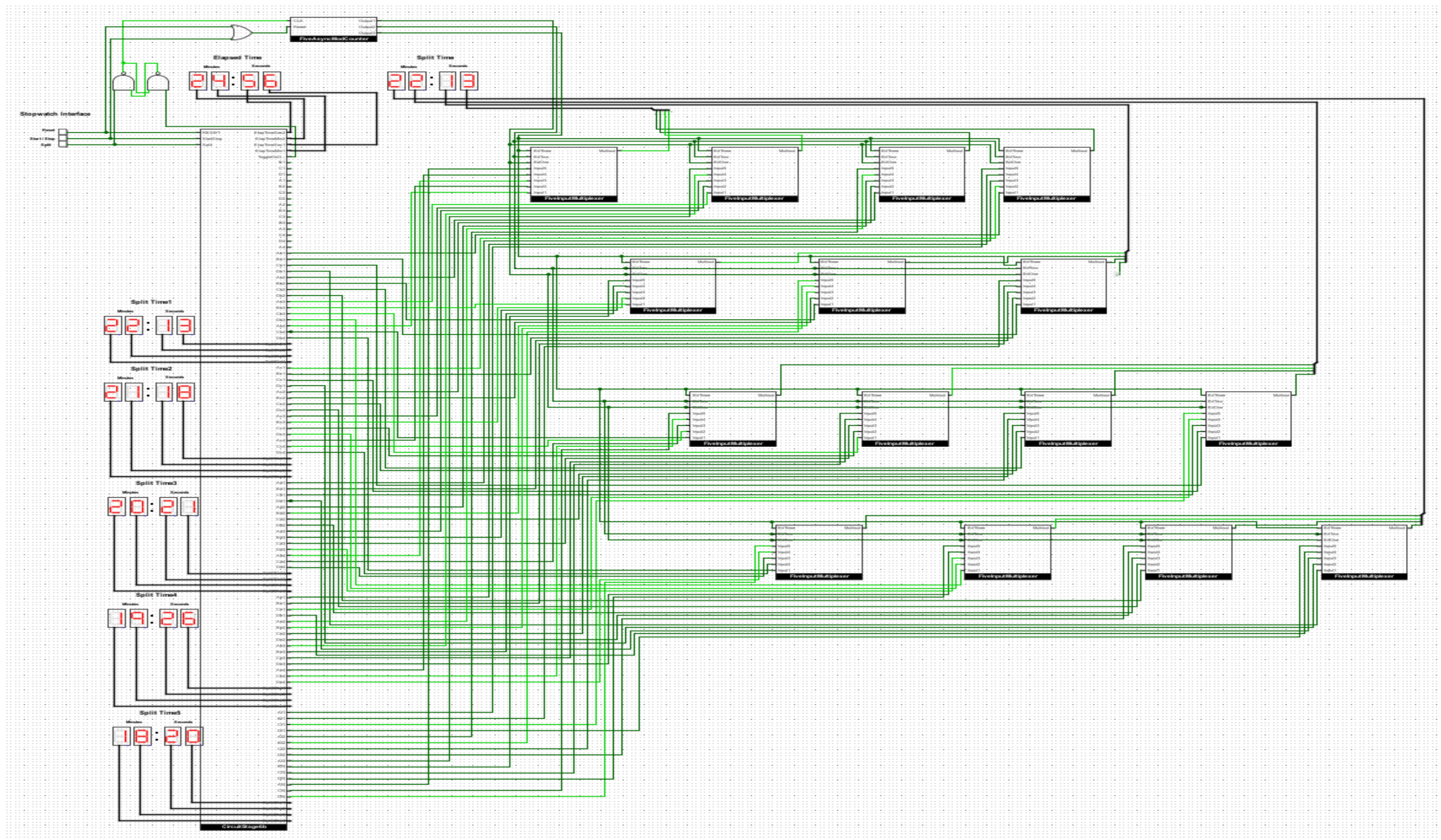


To efficiently store the required 5 splits or 4 additional splits, we can utilise a similar logic as the one used for storing the first 5 splits. By implementing D Flip Flops as registers to store each bit, we can output them to their respective HEX digit displays.

To achieve this, we can connect the first set of D Flip Flop outputs (“BitOneOut” to BitThreeOut or FourOut”) to the Inputs of the next set of D Flip Flop asynchronous registers. Their outputs are then connected to another set of asynchronous registers and so on. We are therefore passing on each of their saved states until we reach the end where the saved state is overridden and therefore deleted.
To ensure that the splits are not registered when the timer is paused, we can implement a NAND gate latch. This latch will only allow signals from the split to reach the CLK inputs when the timer is running. Otherwise, the signal will be blocked by the NAND gate latch. Additionally, the latch is modified with 2 NOT gates which makes the input send 1 as soon as the Split button is

pressed and not when the button is released. Finally, to initiate the movement of the values of the registers down to the next group of registers, we can connect the CLK input of all the D Flip Flops to the Split button. This way, each time the Split button is pressed, the values of the registers will move down to the next set of D Flip Flops. Each group of D Flip Flops will store its respective bit and will be connected to its corresponding HEX digit display.

Stage 6b). Multi split display



To achieve a multi split display for the Split HEX digits, we utilise 5 input multiplexers or 5 x 1 multiplexers (Implemented with a subcircuit called "FiveInputMultiplexer). The main circuit and their outputs are also condensed into a subcircuit (CircuitStage6b) with output pins connected to all of the D Flip Flop register outputs. Connecting the selector inputs of the multiplexers to the output of an asynchronous counter (Implemented with a subcircuit called "SixInputMultiplexer") we switch between the different sets of data, and therefore split times.

Since we have 5 splits, or 15 D Flip Flops registers in total, we need 15 sets of multiplexers. Each input of the multiplexer takes the output of each D Flip Flop from each group based on their respective bit. We connect the output of the D Flip Flops of each group to the inputs of their respective Multiplexer input. For example, the input of the first multiplexer takes the first binary digit of the first 9 second counter, the second multiplexer takes the second binary digit of the 9 second counter, the third digit with the third multiplexer and the fourth binary digit with the fourth multiplexer. We apply this with the second split outputs (second D Flip Flop registers) with the second inputs of the multiplexer and so on until we have all the 9 second counter inputs connected, and able to with the selector. We implement this for all the counters (the modulo 6 counter and 2 minutes counters) and we have a fully functional timer which can display all splits.

To cycle through the multiplexer inputs we connect the selector inputs of the multiplexers to an 5 modulo asynchronous counter that counts to 5, the input of the asynchronous counter to the split button. This allows us to display multiple saved splits when the Split button is pressed on pause state as the multiplexer's output depends on which bit it receives.

However, we need to ensure that the counter does not change data inputs when the timer is in its start state. To address this issue, we add an NAND latch that only allows the counter to receive its CLK input from the Split button when the timer is off.With these implementations, we have created a reliable and efficient way to display multiple saved splits while preventing disruptions to the Split display.

Stage 7). One display for everything

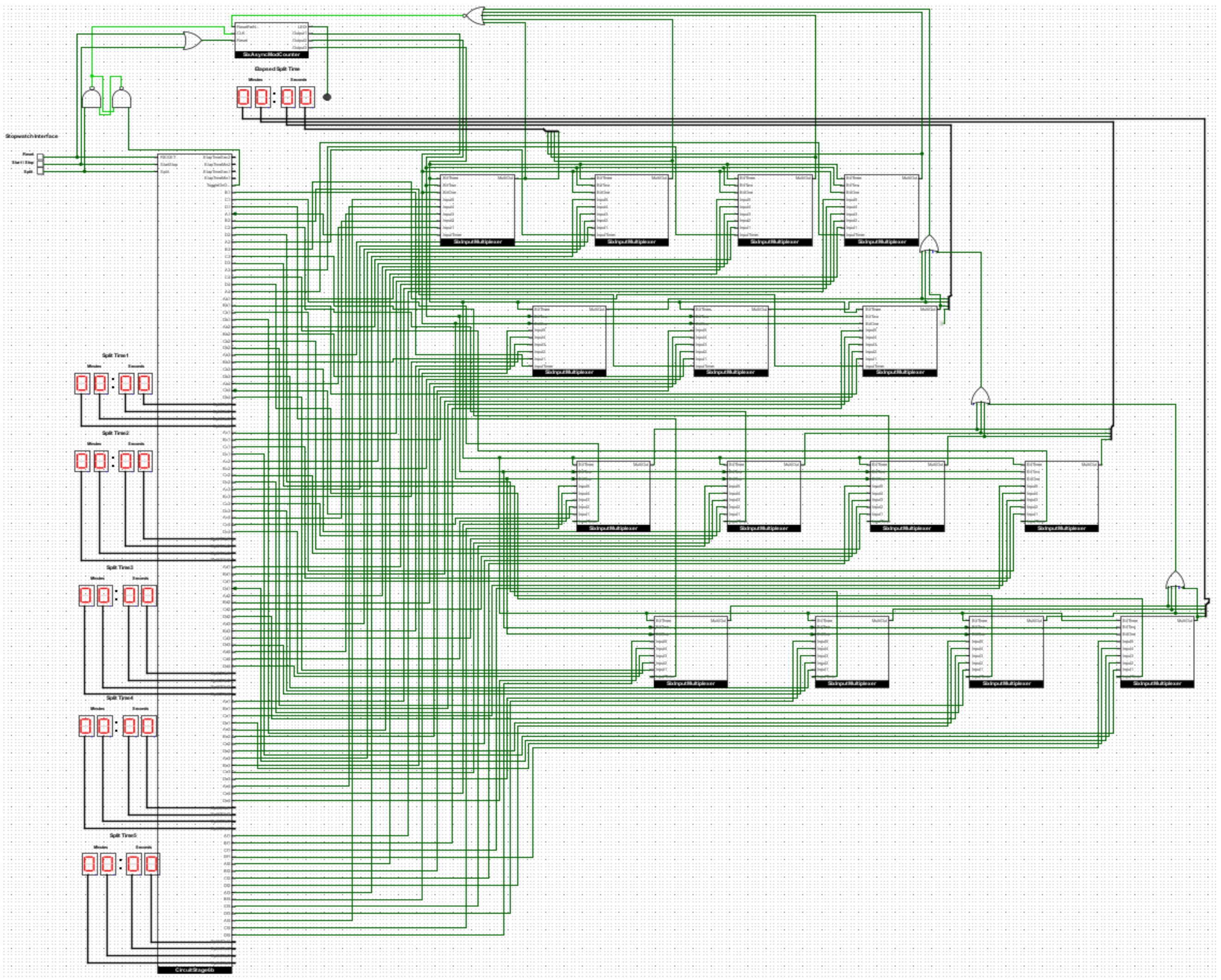
In order to implement a display for both the splits and elapsed time, the design had to be revised to accommodate an additional input for the elapsed timer, we utilise 6 input multiplexers which replaced the 5 input multiplexers and a modulo 6 counter with certain tweaks.

To display the outputs of the timer with the use of input multiplexers, the outputs of the D Flip Flop registers for the counter were connected to the inputs of the multiplexers based on their respective bits. This was achieved by connecting the outputs of the synchronous counters to the multiplexer input that only enables values through when the selector input is 000. Additionally, by having the start/stop button reset the counter to 000, we were able to switch back to the counter input on the multiplexers when the button is pressed when previously having displayed splits when the timer was on was on stop state.

The 6 modulo synchronous counter with the CLK input connected to the Split button was utilised to switch between each of the inputs of the multiplexers. An LED was added to turn on only when a split is displayed by connecting the 3 outputs of the J-K Flip Flops in the modulo 6 counter to an OR gate, and that output to the LED. When the bits on the counter indicate something other than 000, this suggests a split is displayed, therefore the LED will turn on. Furthermore we removed the split time as we had removed the 5 input multiplexers, instead we have 5 split times adjacent to the main circuit to show which times were saved.

To have the HEX digits only display saved times, the counter had to reset back to 000 when inputs were signalling 00:00 and therefore no inputs being received by the multiplexers. This was achieved by connecting the outputs of the seconds and minutes multiplexers to a NOR gate which connects to the reset input of the 6 modulo counter. This is because once the timer begins the seconds and minutes can never output a 0. Therefore, at any point when the Split button with the timer running, a seconds and minutes output must be saved, stopping the NOR gate from sending a 1 signal to reset the counter when the split button is pressed.

This seems simple enough but because of gate propagation delay which causes an illegal state (the NAND gate not receiving any inputs temporarily) which resets the modulo counter early when there are additional split times to display. I managed to fix this by negating the 1 input from the temporary illegal state the NAND gate outputs connects to two AND gates in a way that only allows the output to reset the 6 modulo counter when the previous AND gate receives a 1. This is shown below in the “SixAsyncModCounter” subcircuit in main.circ and stage7.circ.


SixAsyncModCounter
Elapsed Split Time
Minutes
Seconds
Stopwatch Interface
Reset
Start / Stop
Split
Split Time1
Split Time2
Split Time3
Split Time4
Split Time5
SixInputMultiplexer
CircuitStage6b