bilge batsukh

[illegible]

Using a Timer System

Download and open the Logisim file called “timer systems.” Begin with the circuit called “counter intro”.

1. The main component in this circuit is an 8-bit counter. It has two control inputs that are aptly labeled “Control input 1” and “Control input 0”. There is also an 8-bit constant input between the two control inputs and it is set to A5 (A and 5 are useful test inputs because they alternate 0 and 1). Click the “**Simulate**” option at the top of the pane to ensure that the “**Simulation Enabled**” option is indicated. Then click “**Ticks Enabled**” to start the clock ticking. You can change the clock frequency if you want. While the clock is ticking, use the “poke” tool (looks like a hand) to play with the control inputs. **Record your observations of the behavior of the counter under the 4 possible combinations of control inputs below.**

Control 1	Control 0	Counter Behavior
0	0	hold
0	1	Starts counting, increments per clock cycle
1	0	Set counter to A5
1	1	Causes counter to count in reverse

2. Now look at the circuit called “16 bit counter” which shows the counter configured in the “count up” mode. Notice the “splitter” on the Q output which splits the 16-bit output into four 4-bit outputs so that we can connect them to the hex digit displays. The purpose of this circuit is to show you what happens when the counter “rolls over” from its highest value (FFFF) to zero. You can use the “poke” tool to set the value of the counter to FFF0, slow down the clock and observe the LED output when the counter goes from FFFF to 0000 and then to 0001. You can also set the counter value to some small value and watch it count down. Use the mode table that you created for question 1 to tell you what values to put on the control inputs to make it count down. **Record your observations of the behavior of the carry output (to which an LED is affixed) here.**

   The carryout LED activates when it reaches FFFF, indicating the impending rollover. If it is in reverse count mode, it activates at 0000.

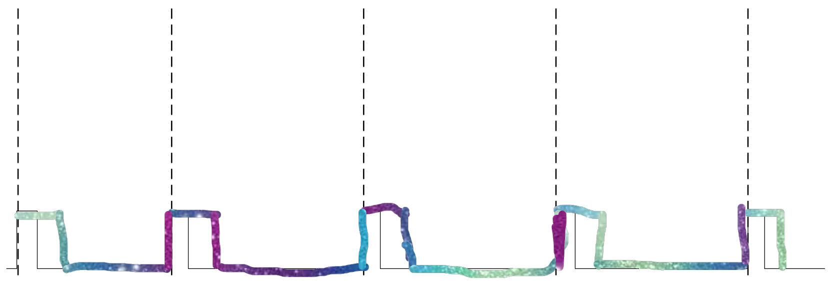
3. Now look at the circuit called “compare register.” This one uses an 8 bit timer in the same up-counting mode as before with a few additional components: n 8-bit “Compare Register”, a comparator and a T flip flop. We changed from a 16-bit counter to an 8-bit counter so that you can observe several iterations of the count sequence in a reasonable length of time. Notice that we are now calling the Clock signal “MCLK” (master clock).
   a. First set the counter to run freely by setting the clock to some high frequency (64 Hz for example) and setting “ticks enabled” (use the Simulate menu from the toolbar at the top). You can use “Ctrl-K” to start and stop the clock.
   b. There are 3 LED indicators, one connected to the carry out from the counter, one connected to the LSB of the counter and one showing the state of the T flip flop. The carry-out from the counter is connected to the Preset input to the T flip-flop. **What effect does that have on the state of the T flip-flop and hence on the output labeled “P output”?**

      The P output will be activated along with the T flip-flop whenever the carryout from the counter goes on

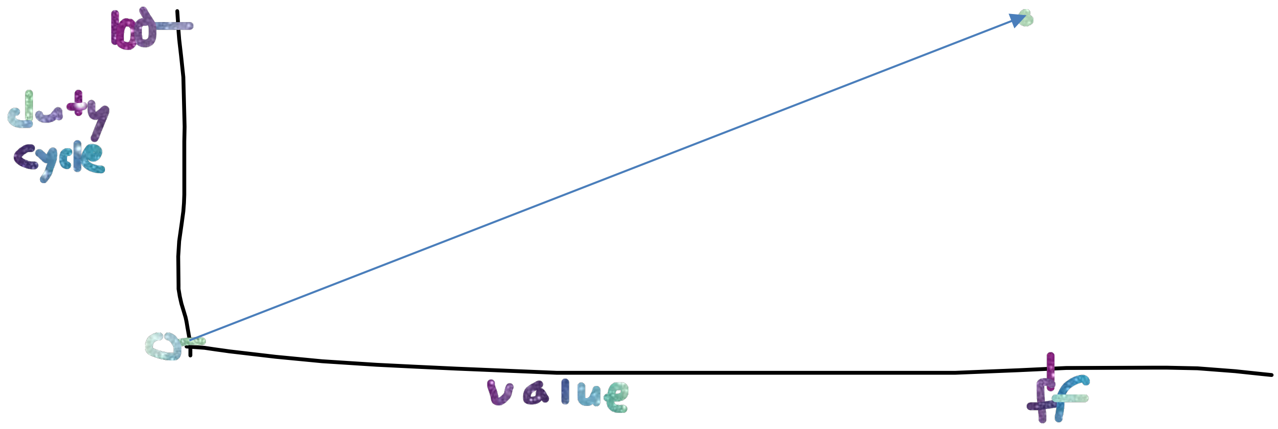
   c. Using the “poke” tool, put a value in the compare register which is equal to one-quarter of FF (in hex). *(Hint: convert FF to binary and shift right 2 places.)* Then start the clock ticking (using Ctrl-K) at a fast frequency and observe the behavior of the LED labeled ”P output”. **Record your observations here**.

      The P output will activate when the P output was off and the counter reaches 3f. It will then remain on until the rollover occurs and the counter reaches 3f again, at which point it turns off. This creates alternating behavior with a duty cycle determined by the value in the compare register.

   d. Consider the time between “ticks” on the carry out as a single period of a clock signal that is MCLK*256 (*why? Because FF in hex is 255 in decimal*). On the timing diagram below, where the “ticks” are the carry-out signal (not to scale), **sketch the signal that appears on the P output.** (*Hint: it has the same period as the “tick” signal but stays high longer.)*



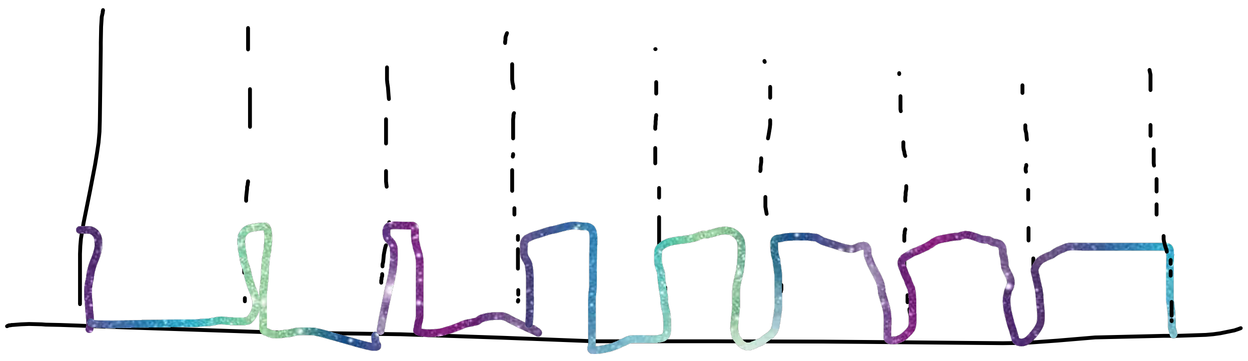
e. Change the value in the Compare Register to observe its effect on the duty cycle of P, that is, the portion of time that the P output stays high in each period. **Draw a sketch of the function that represents the relationship between the duty cycle of P and the value in the compare register.**



f. **What value in the compare register results in a 50% duty cycle? 75% duty cycle?**

4. Finally, consider the circuit called PWM. In this circuit we've added an adder to automatically change the value in the compare register on every "tick". **What effect does this have on the P output?**

This leaves the P output high for an increasing amount of time after every "tick" of our carryout-cycle-clock, so that it will do something like this:



One purpose of this assignment was to introduce a methodology for achieving Pulse Width Modulation (PWM). PWM is a way to simulate an analog voltage using digital signals and is commonly used in embedded applications. To learn more about PWM see Circuit Skills: PWM (Pulse Width Modulation) or Microcontrollers - Introduction to PWM (Pulse Width Modulation)