Wesley Vo
ESE 356 Project 0 Brief Discussions

Part 1

- From the simulated waveforms we can see the signal t_data_out going high when the sequence for bits [3:0] = "1011" at t = 30ns
- At t = 40ns, the reset signal is asserted (active-low) and all the internal registers are cleared
- At t = 50ns, the sequence starts to load the values again (same values as the beginning) and we expect to see t_data_out going high again, however t_clear is asserted (active-high) at the time t_data_out is about to go high at t = 80ns and instead t_data_out stays low
- While the waveform does not explicitly show, I believe the '**reset**' signal is designed to be an **asynchronous** signal because it is within the sensitivity list of the method "prc_seqdet" found in the "seqdet.h" header file
    - Similarly, the '**clear**' signal is not within the sensitivity list but is used within the "prc_output" method found in the "seqdet.cpp" file and this method is used implicitly by the clock edge, making it a **synchronous** signal

Part 2

Note: I know that the project description had the output ports count1 and count2 as sc_bv but I instead made them as sc_uint because I could not find a function online that could convert the data type sc_uint into sc_bv (not sure how to convert from one to the other).

First test:

- Count1 is loaded with 14, count2 is loaded with 5
- Dec1 is asserted at t = 5ns and count1 begins to decrement at this time. Dec2 is left alone so count2 stays at 5
- Count1 continues decreasing until at t = 50ns when it is detected that the value of count1==count2 which is 5.
    - The output signal t_ended stays at high and the contents of each counter remain the same until either counter is loaded which occurs at t = 60ns

Second test:

- Begins at t = 60ns
    - Counter1 = 7
    - Counter2 = 8 and in3 = 3
- Dec1 has already been asserted since the beginning, dec2 starts at t = 65ns
- At t = 75ns, counter2 attempts to subtract 3 from 2 but this will cause an overflow (overflow2 becomes high)
- At t = 75ns, the signal ended goes high and counter1 stops decreasing

Part 3

- Up to t = 60ns, inData is loaded with the hex values "1F1, 0E0, 171, 0E0, 1F0, 170" in that sequence at each clock cycle
- The data 0x0E0 is "ignored" because it is not a type1 data, so the signal t_count does not increase when it detects this data

Wesley Vo

ESE 356 Project 0 Brief Discussions

- The value 0x171 is considered an “error” because the middle four bits is ‘7’ or “0111” and it is an odd parity but the parity bit indicates that it expected it to be even - this causes the signal t_error to increment
    - Same situation applies for the value 0x1F0 - ‘F’ is “1111” which is even parity but the parity bit indicates that it expected it to be odd
- At t = 60ns, t_reset is asserted (active-low) and the output signals t_count, t_error, and t_payload is reset
- At t = 70ns, t_reset is not asserted and values are loaded again as normal
- At t = 130ns, the value 0x171 is loaded and the signals t_count, t_error, and t_payload are gathering the correct data, at t = 150ns t_clear is asserted (active high) for 10ns and the output signals are cleared. After the 10ns, the output signals begin to gather the correct data for the value 0x171 again
- The ‘**reset**’ signal is **asynchronous** since it is within the sensitivity list of the “prc_output” method found in the “communications.h” header file
    - The ‘**clear**’ signal is **synchronous** since it is within the same method but it is not on the sensitivity list and the signal will be checked for if there is a rising clock edge