Danny Dutton
03/23/15
ECE 3544

Project 2: Parity Bits on a Communication Channel

Objective:
The purpose of this project was to design a communications channel that would include a parity generator and checker. In addition, the modules used would include timing delays to simulate real-world hardware.

Design:
This was accomplished using a 9-bit counter for the initial data, a 74HC280 parity generator, two 10-bit registers to transfer the data, another 74HC280 parity generator and an XNOR gate to act as a comparator between the first generated parity bit and the second one. In addition a clock pulse source was provided for use in the test benches.

The 9-bit counter is positive-edge clock triggered and includes an enable and clear input. The enable input will stop the counter from ascending on the clock pulses when asserted low. The clear input will restart the counter at zero when asserted high. The all of the outputs of the counter will connect to the first nine inputs of the first 10-bit register.

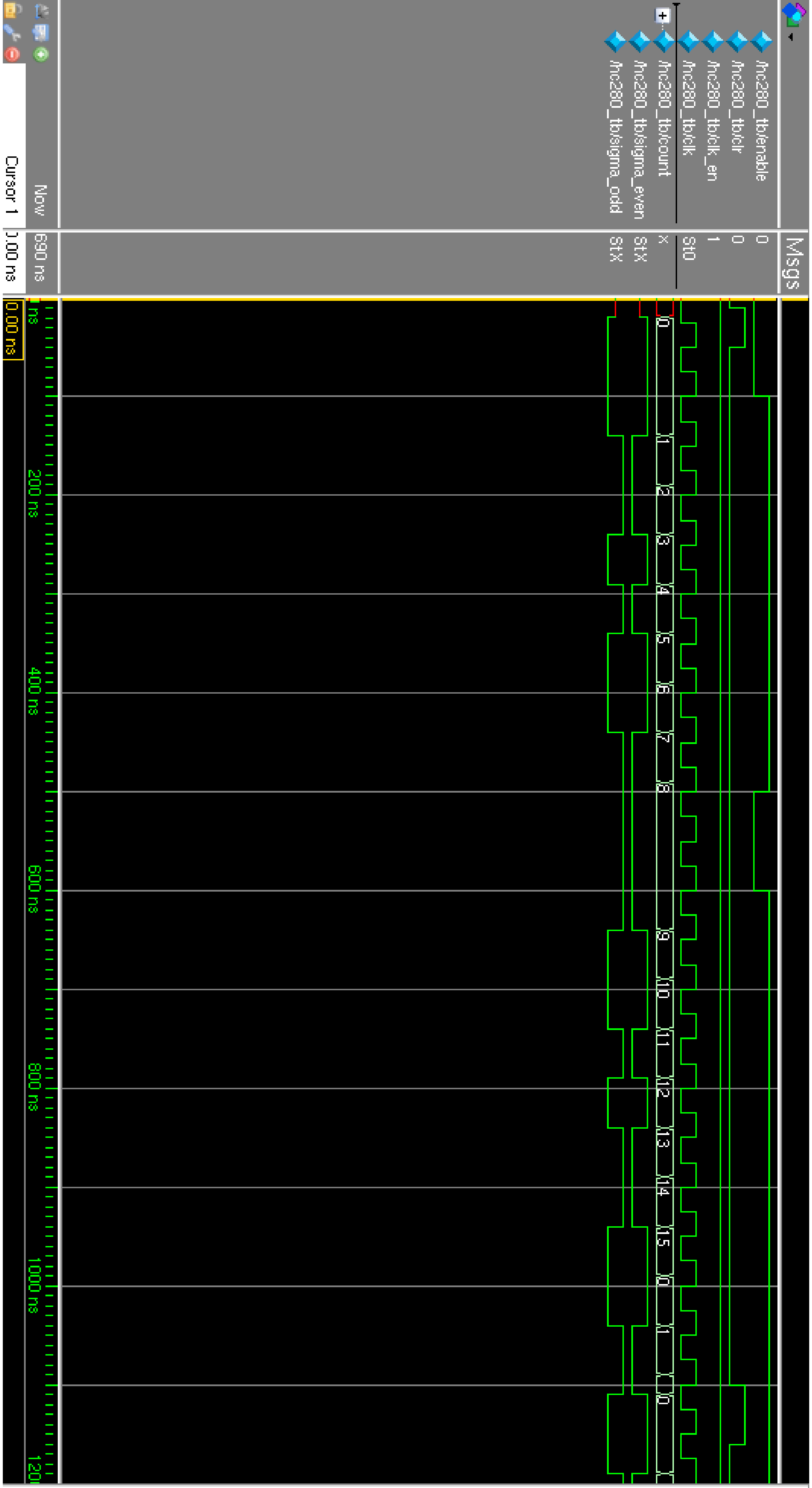
The 74HC280 is a 9-bit parity generator that has a 9-bit input with two outputs, even and odd. When the 9-bit input has an even number of ones, the even output will assert high and the odd will assert low. The output behavior is reversed when the 9-bit input has an odd number of ones. The even output is generated using XNOR on all the inputs, odd output is generated using XOR on all the inputs. The even output is ignored in the project while the odd output is connected to the 10th input on the first 10-bit register. This module uses timing delays to mimic the propagation delay of the actual hardware: 20ns on the switch from even high and odd low to even low and odd high, 23ns for the opposite switch. A specify block is used to implement this delay.

The 10-bit register is positive-edge clock triggered and simply moves the bits from the 10-bit input to the 10-bit output. There are two of these registers connected in series so as to mimic a transmission channel of some distance between them.

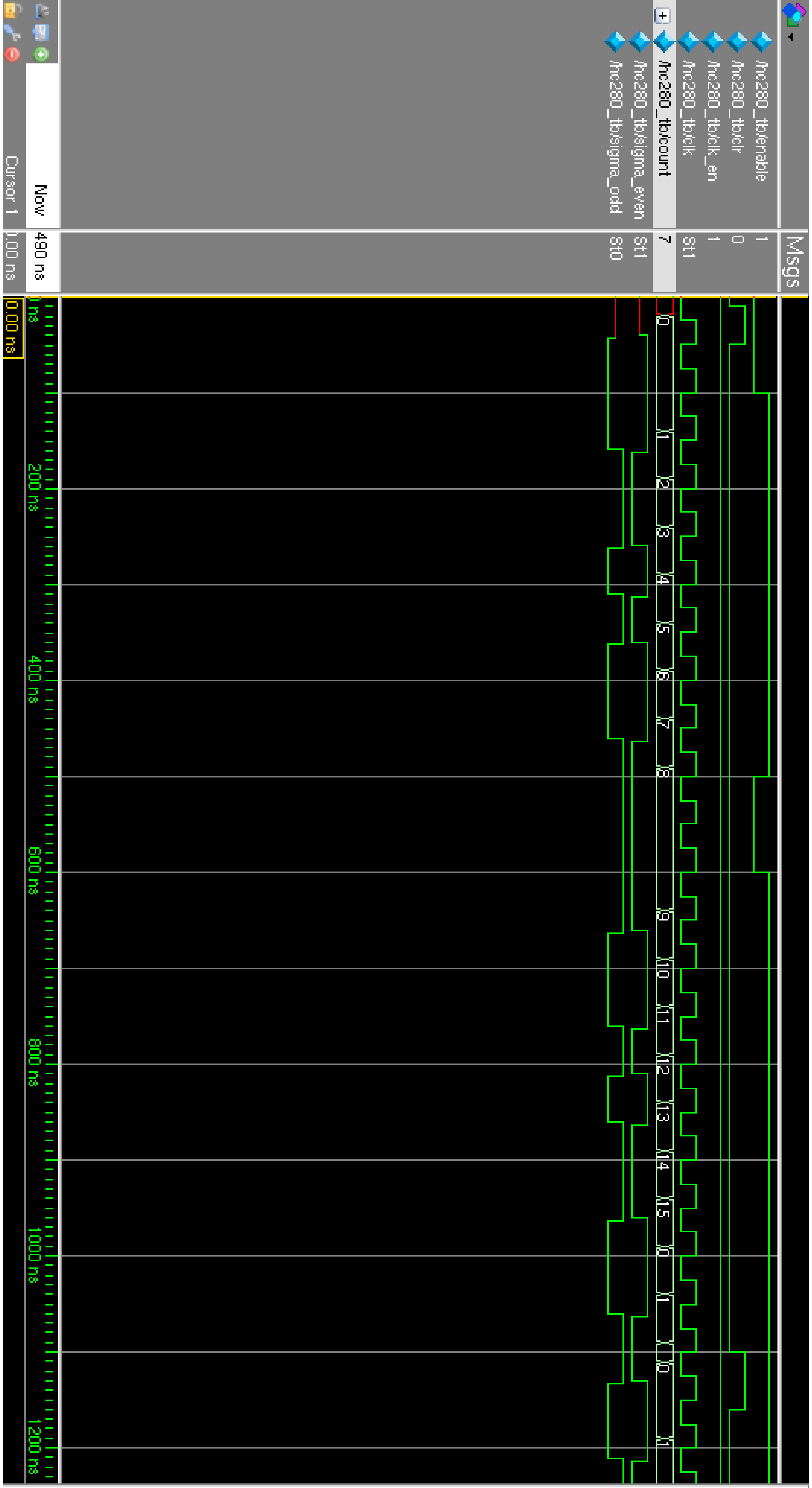
The second 74HC280's input connects to the first nine bits of the second register's output. This generates another parity bit that should be the same as the parity bit on the second register's 10th bit output. A comparator consisting simply of an XNOR gate compares the two outputs. Despite using the odd output on the 74HC280, an even parity bit is being generated since if the number of high bits on the 9-bit data bus is odd, a 10th high bit is being generated to make the count even and vice versa.

Results:
The first test bench created is for the 74HC280 without a propagation delay and the 9-bit counter. Looking at the waveform generated, the counter does not begin to increment until the first positive-edge after the enable input is set high. The slight delay between the positive clock edge and when the counter changes is due to the delays built into the counter (10ns on the clear input, 15ns on the clock input). At 500ns, the enable is set low and the counter pauses until set positive again. The clear is set high 1100ns and the output is reset from two to zero. This test bench acts exactly as expected. The other test bench for hc280 uses propagation delays. This is seen by the slight delay between the count change and the sigma_even and odd change.

hc280_tb, no delays

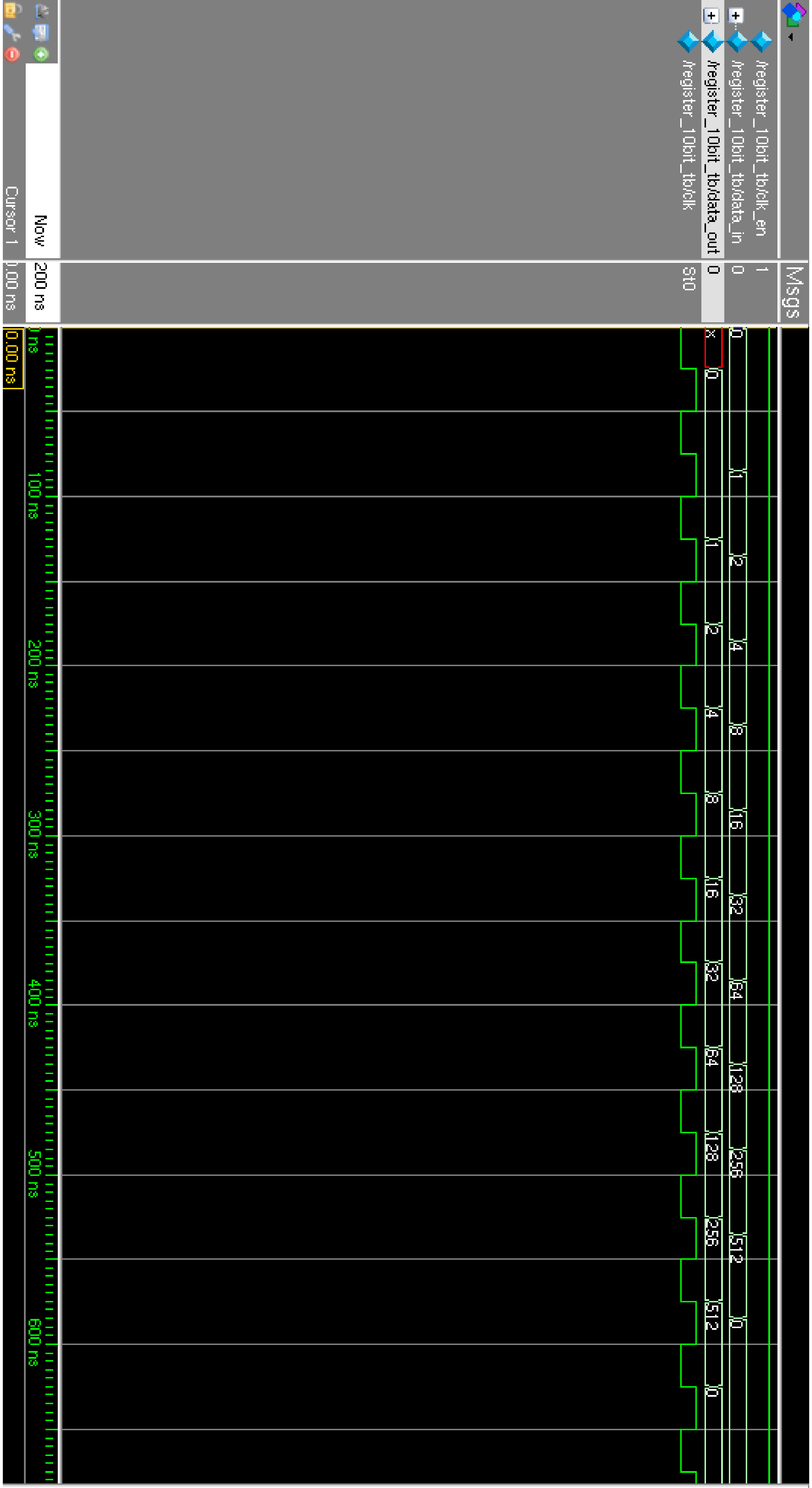


hc280_tb, with delays



The second test bench is for the 10-bit register. The inputs are just 1 bit shifted left for each clock cycle. The output is positive-edge clock triggered. This test bench works exactly as expected.

register_10bit_tb



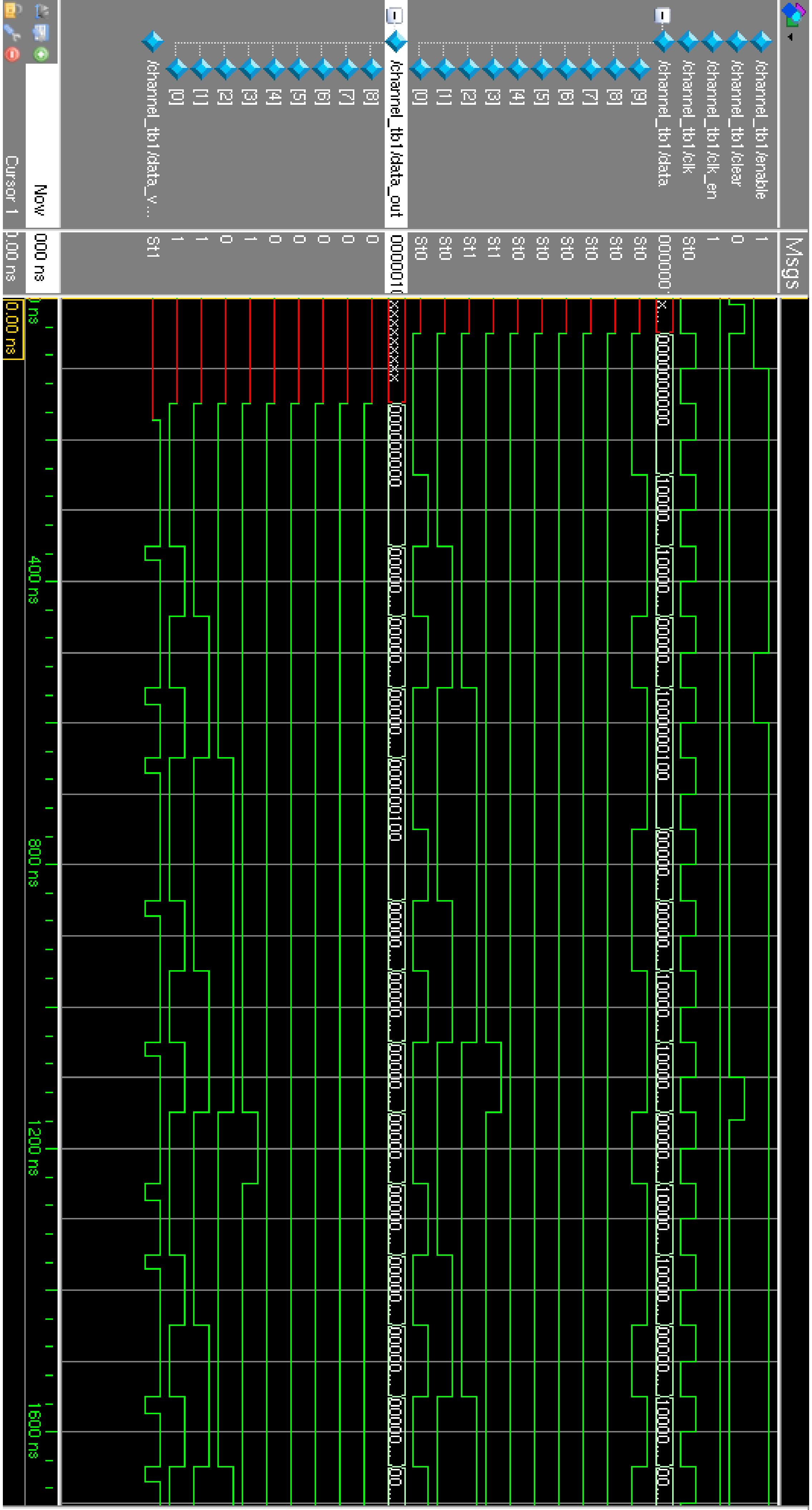
	Msgs
/register_10bit_tb/clk_en	1
/register_10bit_tb/data_in	0
/register_10bit_tb/data_out	0
/register_10bit_tb/clk	St0
Now	200 ns
Cursor 1	0.00 ns

The last set of test benches are the combination of all of the modules into a channel_tb. This test bench consists of two modules, transmit and receive. Transmit contains the counter, first 74HC280, and first register. The receive module contains the second register, second hc280, and the comparator. In addition, the clock is instantiated in this test bench for the counter and registers.

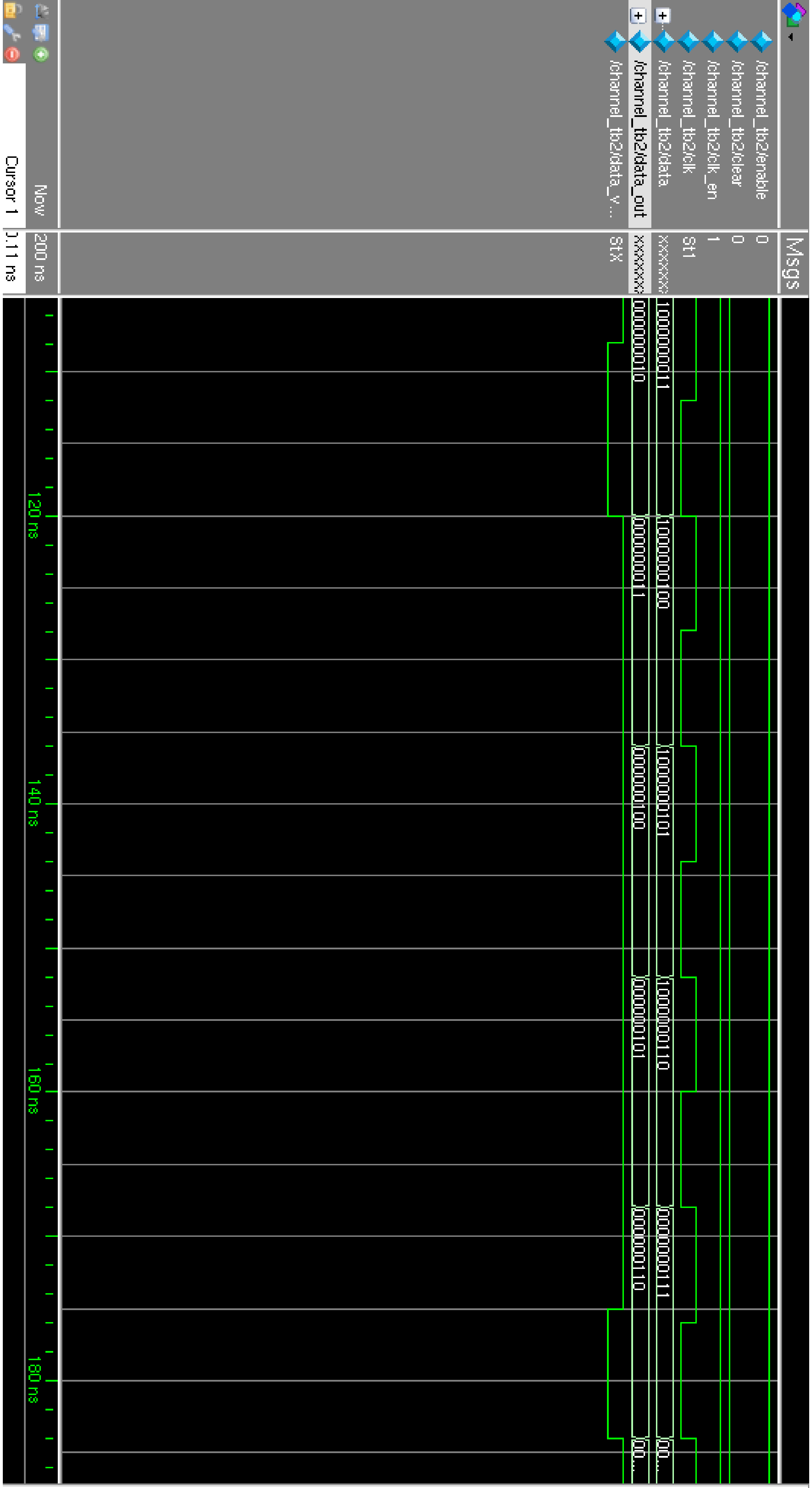
The first test bench for channel_tb, channel_tb1, is intended to use a clock period that would allow for the propagation delays in the first 74HC280 to keep up with the time difference from when the counter is incremented to when the first register passes on its inputs to the next register. This means that the clock must be at least 23ns (as per the delays for the 74HC280 to be sure that the correct parity bit is at the 10th input on the first register when the clock's signal rises again. In channel_tb1, a clock period of 100ns was used to ensure the circuit functioned correctly. Looking at the waveform, there is a 20-23ns dip on the data_valid output. This is due to the 74HC280 having the propagation delay. Otherwise, the outputs of circuit match what they should expect to be.

The second test bench for channel_tb, channel_tb2, is intended to show what happens when the clock period is too short for the circuit to work correctly. The clock period was set to 17ns to ensure that the counter was able to work correctly since it has a clock delay of 15ns. Since the 74HC280 delay is too high, the parity bit generated does not match what it should. On the waveform, at around 140ns, the data value is 1000000101 and this is moved to data_out at around 160ns. The 10th bit should be a zero since there are an even number of ones in the count. The second 74HC280 is unable to find the correct parity bit because it too is too slow for the clock cycle.

channel_tb1


Msgs
/channel_tb1/enable 1
/channel_tb1/clear 0
/channel_tb1/clk_en 1
/channel_tb1/clk St0
/channel_tb1/data 000000
[9] St0
[8] St0
[7] St0
[6] St0
[5] St0
[4] St0
[3] St1
[2] St1
[1] St0
[0] St0
/channel_tb1/data_out 000001
[8] 0
[7] 0
[6] 0
[5] 0
[4] 0
[3] 1
[2] 0
[1] 1
[0] 1
/channel_tb1/data_v... St1
Now 000 ns
Cursor 1 0.00 ns
0.00 ns
400 ns
800 ns
1200 ns
1600 ns

channel_tb2



```
                         Msgs
/channel_tb2/enable      0
/channel_tb2/clear       0
/channel_tb2/clk_en      1
/channel_tb2/clk         St1
/channel_tb2/data        xxxxxxxx   1000000011   1000000100   1000000101   1000000110   0000000111   00...
/channel_tb2/data_out    xxxxxxxx   0000000010   0000000011   0000000100   0000000101   0000000110   00...
/channel_tb2/data_v...   StX

Now                      200 ns
Cursor 1                 ).11 ns

                         120 ns   140 ns   160 ns   180 ns
```