Fpgautil -b our_bit.bit

Demo.sh: calls tb.sh first with preprogrammed address ranges(0xb0028000 to 0xb0028010) as $arg1 and $arg2, tb.sh then kicks start the internal counters and registers to record the different values on the bus and store them into registers. Demo.sh then calls 4KB_write. 4KB_write writes to 0xb0028000 (within the range of interested addresses) from the OCM in the background, then waits 1 second so it makes a difference on the graph, and in the same time, tb.sh writes [$current_clk_count 1] to a file named data.txt if there is transfer within the range at this clock cycle or [$current_clk_count 0] if there is no transactions within this address at this clock. Demo.sh then calls gnuplot on gnuplot config file demo.plt, which sets up the auto scale plotting on x axis and reading from data.txt as it's being updated to show the real time data by refreshing the display window with newly added values.

Notes:

1. After you run demo.sh, ctr-c the parent process (current way to stop recording and graphing) won't kill all the child processes I created so we will have to kill them with a ps to stop data transfer and data.txt being updated.
2. Currently my graphing will always miss the first transaction because I'm indexing the transaction address with their corresponding wlast signals from the time the previous transaction has finished. And since there is no previous transaction for the first transaction, my implementation will always start with the second transaction and start graphing from there. This also means tb.sh must always start first compared to the axi bus transfer scripts.
3. Currently the array registers don't get cleared out after a global enable signal is de-asserted to preserve the value from the last run after the global enable signal is asserted by my tb.sh file.
4. The diff, compared to 4KB_write, writes to starting location 0xb0028080 which is outside of the range so it won't write a 1 in the data.txt so the data.txt will be all 0s and the graph will be all 0s.