EPK Input:

The input to EPK is periodic, where the design sets *rdy_data_in* high, one clock cycle later the testbench sets *rd_data_in* high for the next 32 clock cycles. During this time the input is being decoded in *bitslice_vector_decode*, its computation is one clock cycle behind *data_in* so, it finishes computing it's output 33 clock cycles after *rd_data_in* goes high, it outputs a flag to *p_mat_storage_controller* indicating it's data is ready to be consumed by *epk_slice_storage* on the 34 clock cycle. On the following clock cycle *epk_slice_storage* is triggered, this takes 128 clock cycles to store the 64 bits of the decoded EPK, 2 cycles per bit. After which on the next clock it triggers a flag for the controller indicating it's done. The controller than on the clock cycle after *epk_slice_storage*'s flag goes high sets *rdy_data_in* back to high restarting the EPK input cadence. The delay between the rising edges of *rdy_data_in* for 32 bytes of EPK and the next rising edge of *rdy_data_in* can be computed in terms of clock cycles as,

Operation	Delay (Clocks)
rd_data_in	1
data_in	32
bitslice_vector_decode	1
bitslice_vector_decode done flag	1
epk_slice_storage start flag	1
epk_slice_storage	128
epk_slice_storage done flag	1
rdy_data_in	1
Total	166

Table 1. EPK Expected Execution Time

Since the delay of 166 clock cycles is for one period of reading 32 bytes of EPK, and the EPK in total is 70752 bytes this cycle happens 2211 times. **Therefore, the expected execution time is 2211*166 = 367026 clock cycles.**

Figure 1 shows the simulation of one EPK input cycle. With a 20 ns clock the time between two rising edges of *rdy_data_in* is 3.32 us. 3.32us/20ns = 166. Verifying our computed execution time. This execution time can be seen by running, *TB_mayo_verify.vhd, TB_p_mat_storage.vhd,* or *TB_p_mat_storage_datapath.vhd*.

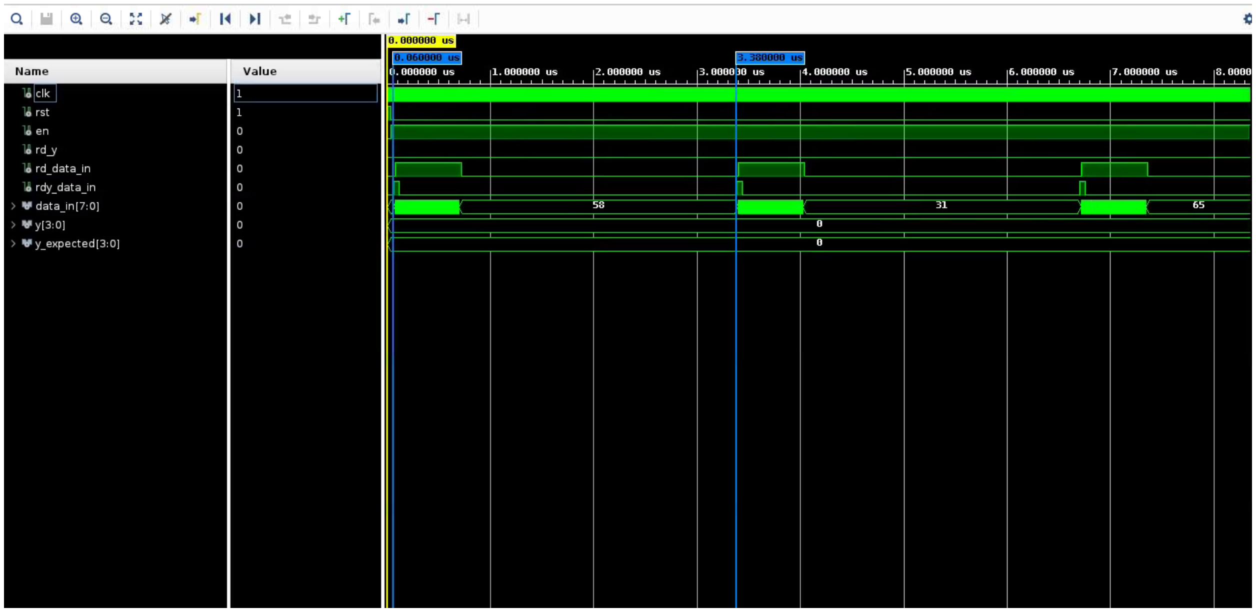


Figure 1. EPK input execution time for one period

SIG Input:

After EPK storage is done and *rdy_data_in* goes high again *data_in* changes to expecting the SIG. unlike the EPK, SIG can be decoded and stored in a single clock cycle, one delayed from the input. Meaning to read and store all 297 SIG bytes takes 298 clock cycles, the done flag is then raised on the following clock cycle after all bytes have been stored. The expected execution time can be expressed as,

Operation	Delay (Clocks)
rd_data_in	1
data_in	297
s_vec_storage done flag	1
Total	299

Table 2. SIG Expected Execution Time

The expected value can be confirmed by running *TB_s_vec_storage.vhd* and measuring the time between the rising edges of the input load and output done. This was measured to be 5.98 us, for a 20 ns clock. **5.98us/20ns = 299 clock cycles**.

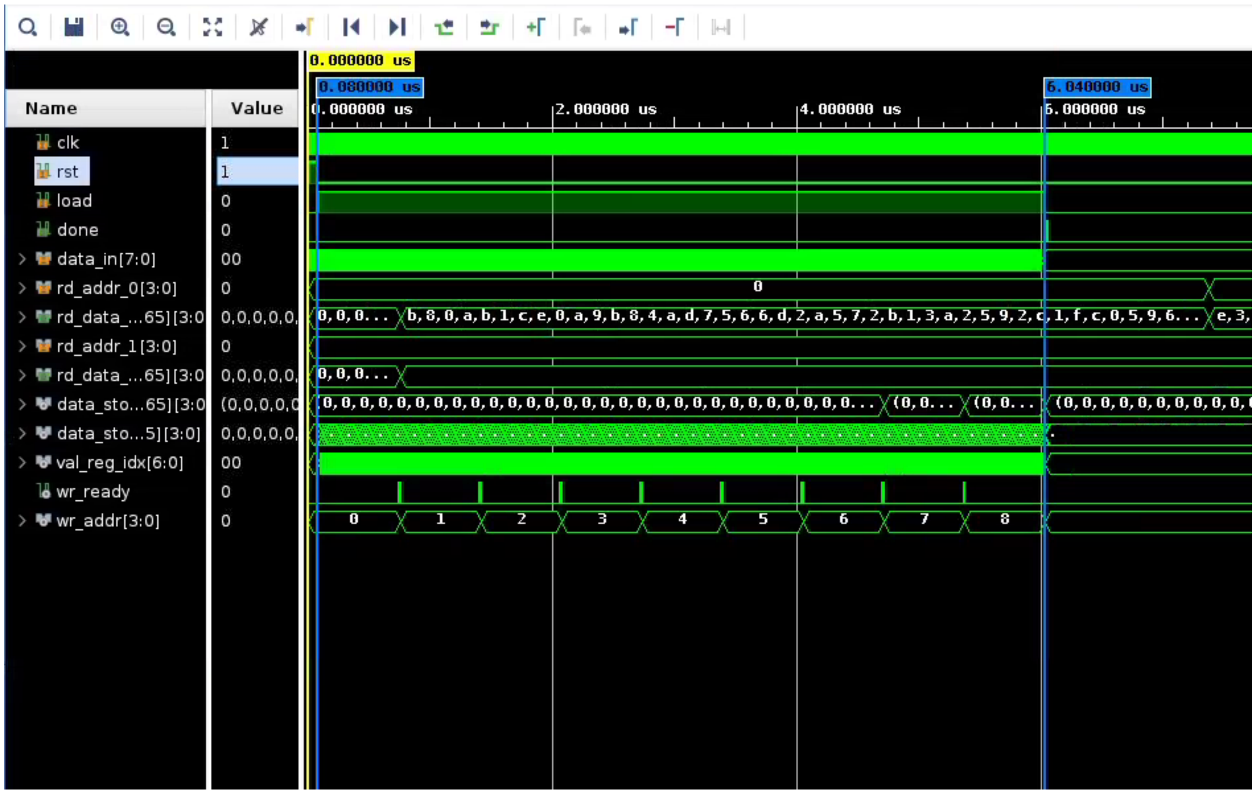


Figure 2. SIG input execution time

Compute y long:

Compute y long follows the nested loop structure shown below, where *k* and *m* are MAYO1 constants and equal 9 and 64 respectively. The primary computation of *u* happens inside the aa loop, since *u* is 64 elements, u(aa) is computed inside the inner most loop. The outer two loops cause the inner most loop to trigger 45 times, that means the expected number of u(aa) computations is 64*45 = 2880 cycles.

```
for ii = [0:k-1]
    for jj = [k-1:-1:ii]
        for aa = 1:m
        end
    end
end
```

However, my design uses an addition clock cycle per loop after the inner aa loop finishes executing to reset the counters, or in the case of the final loop to output the done flag. This means in total the expected execution time is actually 64*45 = 2925 cycles.

This can be verified by running *TB_compute_y_long.vhd* and measuring the time between the first rising edge of the clock when enable is high and the rising edge of done. This was measured to be 5.85 us for a 20 ns clock. **58.5us/20ns = 2925 clock cycles.**

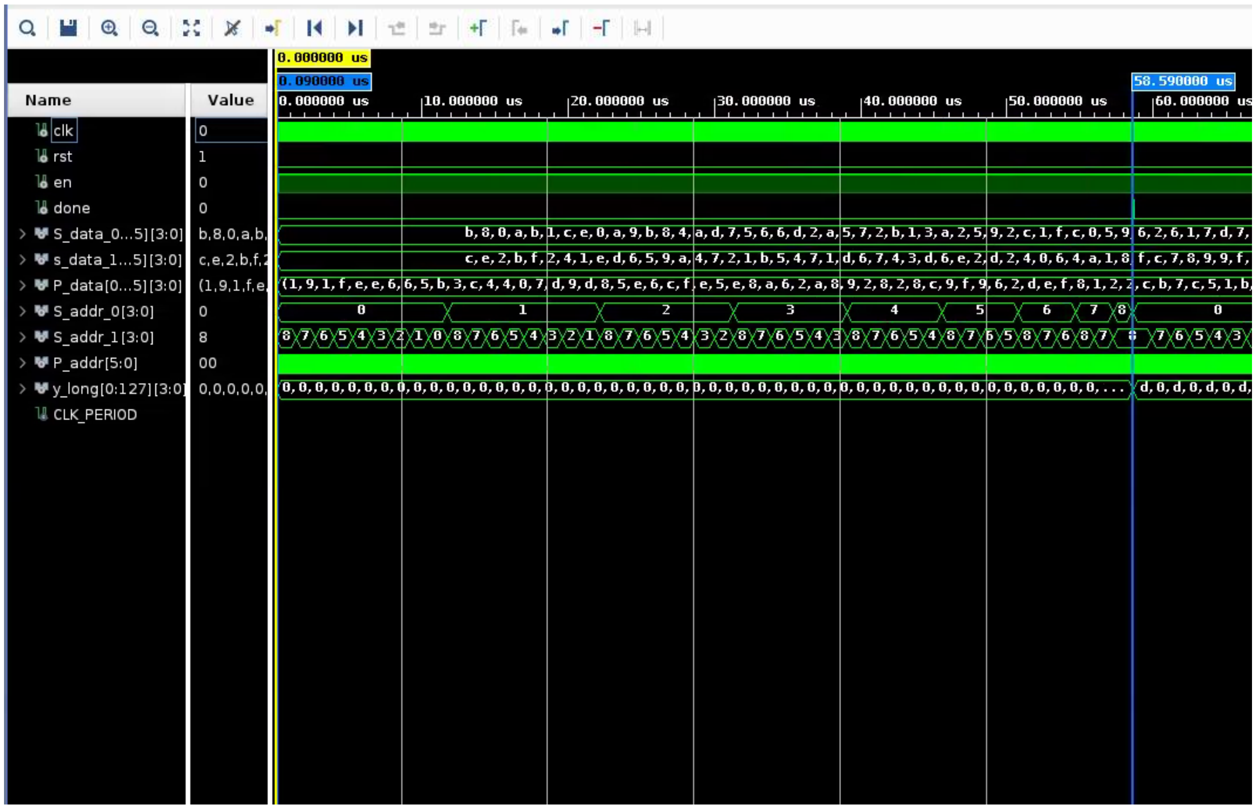


Figure 3. compute y long execution time

Collapse y long:

The final major operation of the design is to collapse *y_long* into the final output *y*. This operation follows the loop structure shown below. Where *k* and *m* are the save MAYO1 constants from compute y long, and *F_TAIL_LEN* is another constant with a value of 5. In total the inner loop, where the main computation happens executes 220 times to compute *y*.

```
for idx_o = [m + k * (k + 1) / 2 - 2 : -1 : m]
    for idx_i = [1:F_TAIL_LEN]
    end
end
```

In addition to the 220 cycles needed to compute *y* this module also uses the first clock cycle at the start to latch *y_long*. It uses one cycle after *y* is computed to indicate the output should start being written. Writing the output takes 64 clock cycles since *y* has 64 elements. Expected execution time then is 1 + 220 + 1 + 64 = 286 clock cycles.

The execution time of collapse y long can be verified by running *TB_collapse_y_long.vhd* and measuring the time between the first rising edge of the clock when enable is high and the falling edge of rd_y

(When the output is no longer valid). This was measured to be 5.72 us for a 20 ns clock. **5.72us/20ns = 286 clock cycles.** Agreeing with our expected execution time.

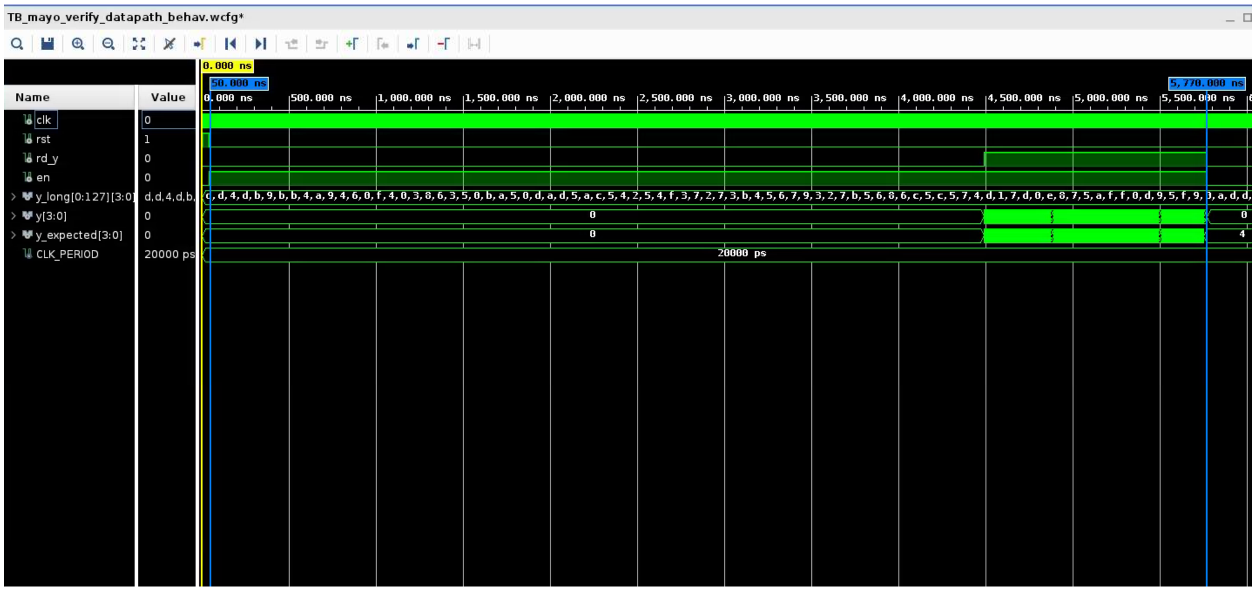
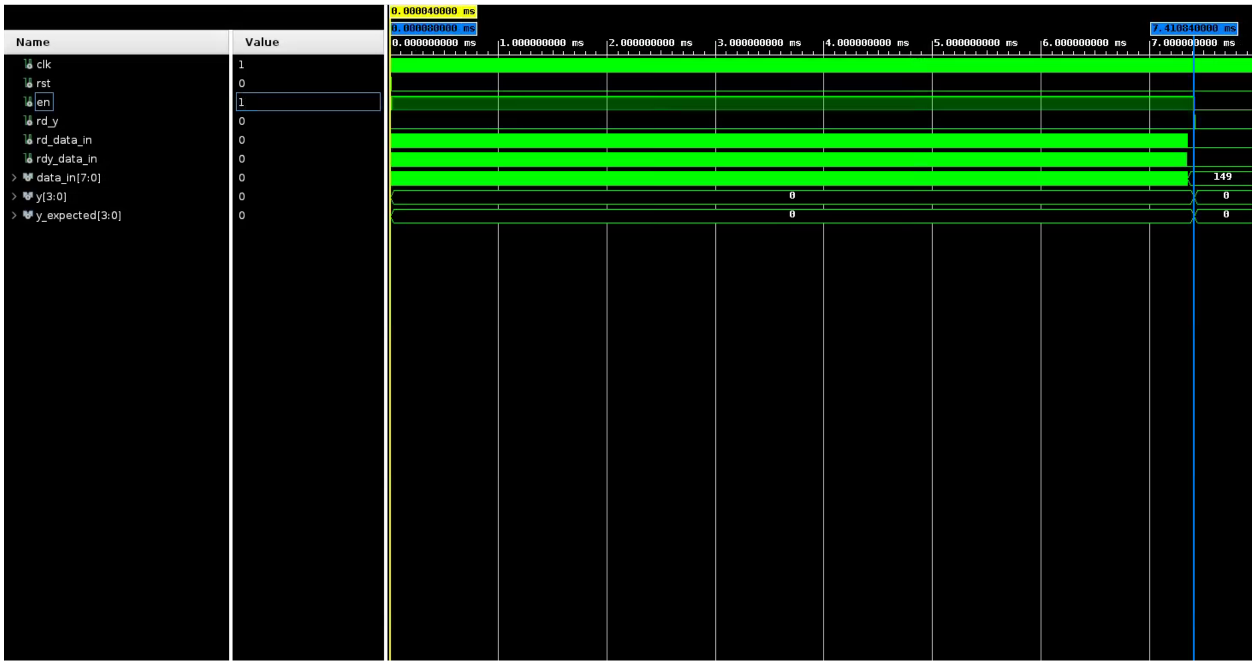


Figure 4. collapse y long execution time

Full Design (Mayo Verify):

The expressed execution times for *compute_y_long* and *collapse_y_long* where for the stand-alone modules. When integrated into the full design both entities have an addition clock delay imparted by the *mayo_verify_controller* changing start to support the next stage of operation. Thus the total expected execution time of the design is the sum of all the modules + 2. That is, 367026+299+2925+286+2 = 370538 clock cycles.

When simulating the full design using *TB_mayo_verify.vhd* the measured execution time was 7.41076 ms when us for a 20 ns clock. **7.41076ms/20ns = 370538 clock cycles.** This agrees with our expected execution time.



Name	Value
clk	1
rst	0
en	1
rd_y	0
rd_data_in	0
rdy_data_in	0
data_in[7:0]	0
y[3:0]	0
y_expected[3:0]	0

Figure 5. Mayo Verify execution time