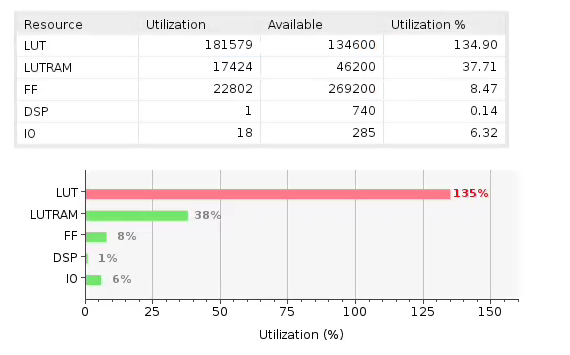
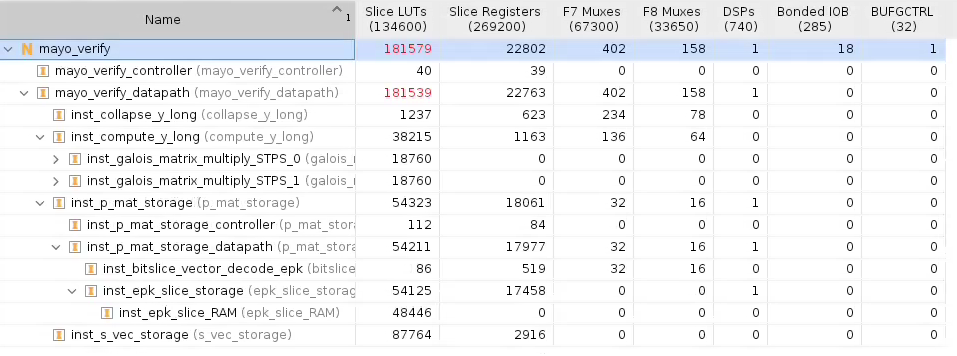
**Utilization:**

Despite the circuit being logically correct. There is a glaring error when considering the synthesized utilization shown in Figure 1. The LUT usage exceeds the maximum available for the largest Artix-7. This prevents Vivado from being able to place the design on the chip. Synthesis was as far as the design could go.



**Figure 1.** Mayo Verification utilization results

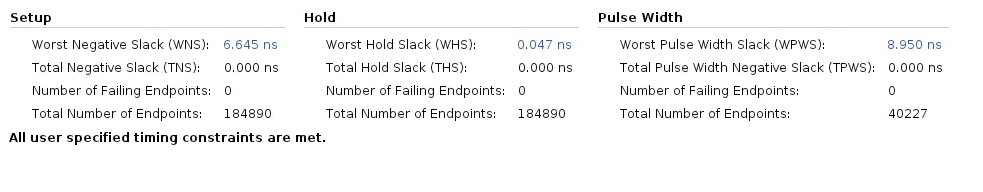


**Figure 2.** Mayo Verification hierarchical utilization results

Because of assumption 2, I knew the design was going to be quite large. The parallel matrix computation of (1x66)\*(66x66)\*(66x1) has 66\*66 + 66 = 4422 4-bit Galois multiply operations. Each Galois multiply contains three LUTs to performs the operations. And that’s excluding the Galois adders (xor) needed to compute the computation. All that happens in *inst\_compute\_y\_long,* I expected the utilization on it to be large. Same for the *inst\_epk\_slice\_storage* since it’s storing the entirety of the EPK, which is ~70 Mbytes. What is surprising about these utilization results is *insta\_s\_vec\_storage*. The SIG input is only 297 bytes, simply stores the input into a fabric “pseudo RAM”. And provides two addresses to simultaneously read out two different addresses. I am unable to understand why this entity takes up nearly half the total available LUTs to implement. This would be something to further investigate in a revision 2.0.

**Timing:**

The result of running timing analysis on the synthesized design with a 20 ns clock constraint are shown below in Figure 3. From these results our worst negative slack is 6.645, which means our clock period could be theoretically lowered to **13.355 ns** for a minimum clock period which corresponds to **74.9MHz** maximum clock period.



**Figure 3.** Synthesized timing results for a 20 ns clock

Given assumption 2 of trying to design for a minimum number of clock cycles rather than a shortest possible critical path I believe the estimated maximum frequency is about expected. There are ample places in the *compute\_y\_long* entity to add pipeline registers to shorten the critical path, and increase the maximum clock frequency. The challenge to this being *compute\_y\_long* is designed in such a way as that makes reading in the EPK slower than it takes to store it. In order to pipeline the design, a lot of it would need to be rebuilt from the ground up to handle the issue of an input data rate that’s faster than the processing speed.