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.

Resource	Utilization	Available	Utilization %
LUT	181579	134600	134.90
LUTRAM	17424	46200	37.71
FF	22802	269200	8.47
DSP	1	740	0.14
IO	18	285	6.32

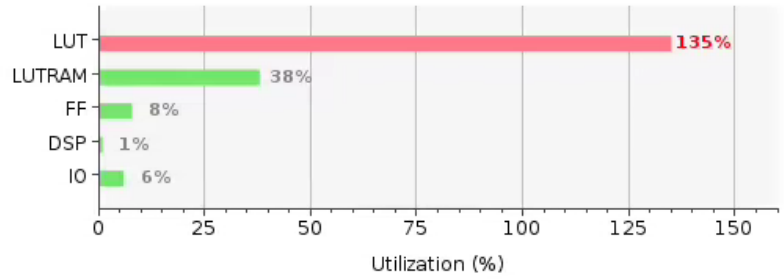


Figure 1. Mayo Verification utilization results

Name	Slice LUTs (134600)	Slice Registers (269200)	F7 Muxes (67300)	F8 Muxes (33650)	DSPs (740)	Bonded IOB (285)	BUFGCTRL (32)
mayo_verify	181579	22802	402	158	1	18	1
mayo_verify_controller (mayo_verify_controller)	40	39	0	0	0	0	0
mayo_verify_datapath (mayo_verify_datapath)	181539	22763	402	158	1	0	0
inst_collapse_y_long (collapse_y_long)	1237	623	234	78	0	0	0
inst_compute_y_long (compute_y_long)	38215	1163	136	64	0	0	0
inst_galois_matrix_multiply_STPS_0 (galois_	18760	0	0	0	0	0	0
inst_galois_matrix_multiply_STPS_1 (galois_	18760	0	0	0	0	0	0
inst_p_mat_storage (p_mat_storage)	54323	18061	32	16	1	0	0
inst_p_mat_storage_controller (p_mat_stor	112	84	0	0	0	0	0
inst_p_mat_storage_datapath (p_mat_stora	54211	17977	32	16	1	0	0
inst_bitslice_vector_decode_epk (bitslice	86	519	32	16	0	0	0
inst_epk_slice_storage (epk_slice_storag	54125	17458	0	0	1	0	0
inst_epk_slice_RAM (epk_slice_RAM)	48446	0	0	0	0	0	0
inst_s_vec_storage (s_vec_storage)	87764	2916	0	0	0	0	0

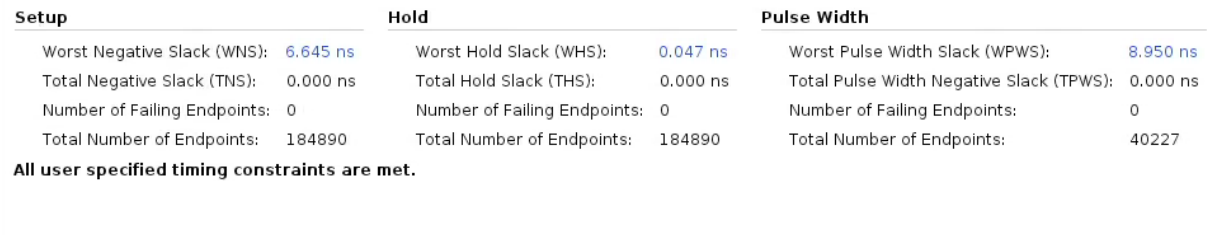
Figure 2. Mayo Verification hierarchical utilization results

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.



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.645 ns	Worst Hold Slack (WHS):	0.047 ns	Worst Pulse Width Slack (WPWS):	8.950 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	184890	Total Number of Endpoints:	184890	Total Number of Endpoints:	40227

All user specified timing constraints are met.

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.