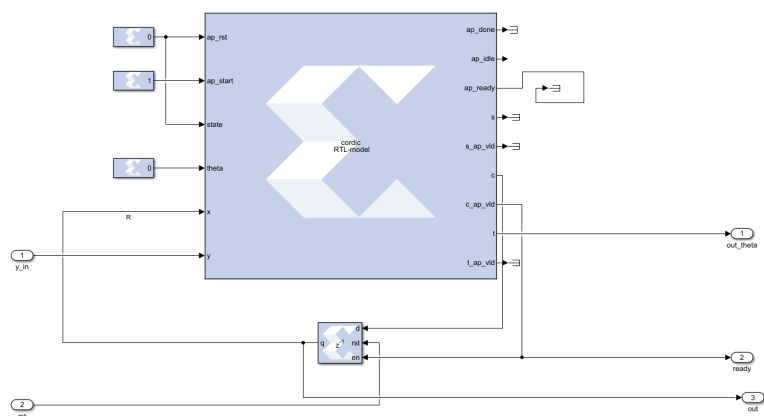


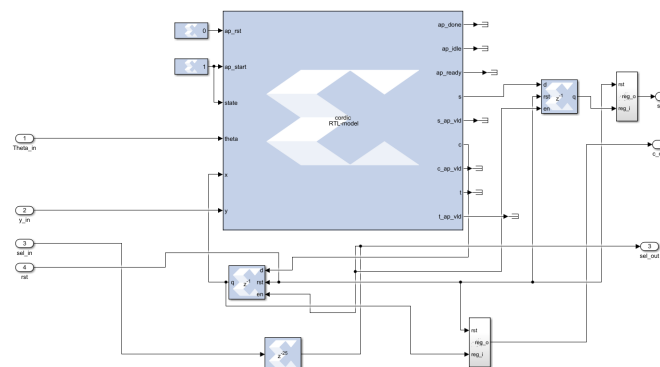
## Calculations of elements

My design allows the variables to be loaded in a non-systolic way (the inputs can be placed into design all at once) and yet load the elements into the cordic blocks in a pipelined systolic manner. This Allows it to be easily used in vitis because it's harder to control inputs that are very cycle and time dependent.

### Arctan block



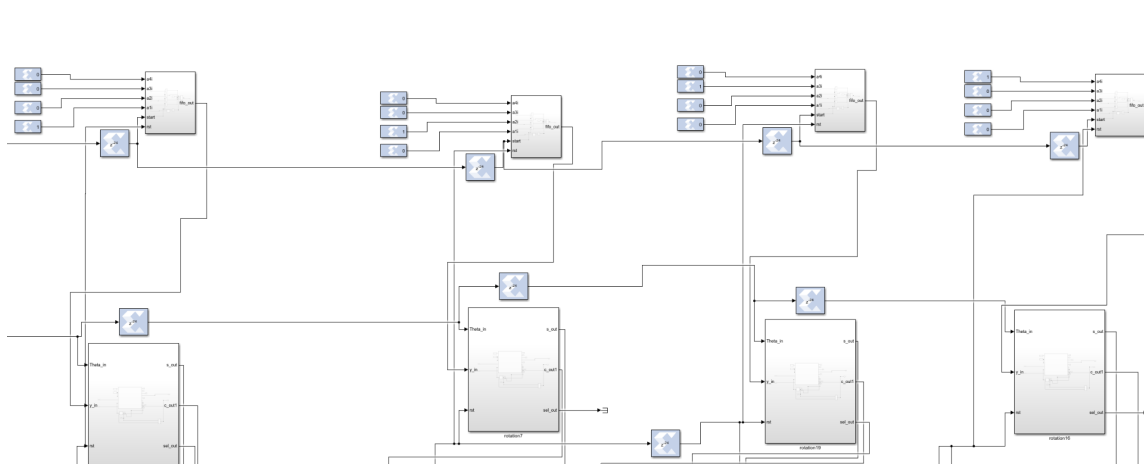
### Rotation Block



Shown above is the Arctan block which feeds back the Cosine for the next theta and a  $a_{i_i}$  value, the register saves that value for when it's done calculating that value and when it's time for a new matrix value, the register can be reset.

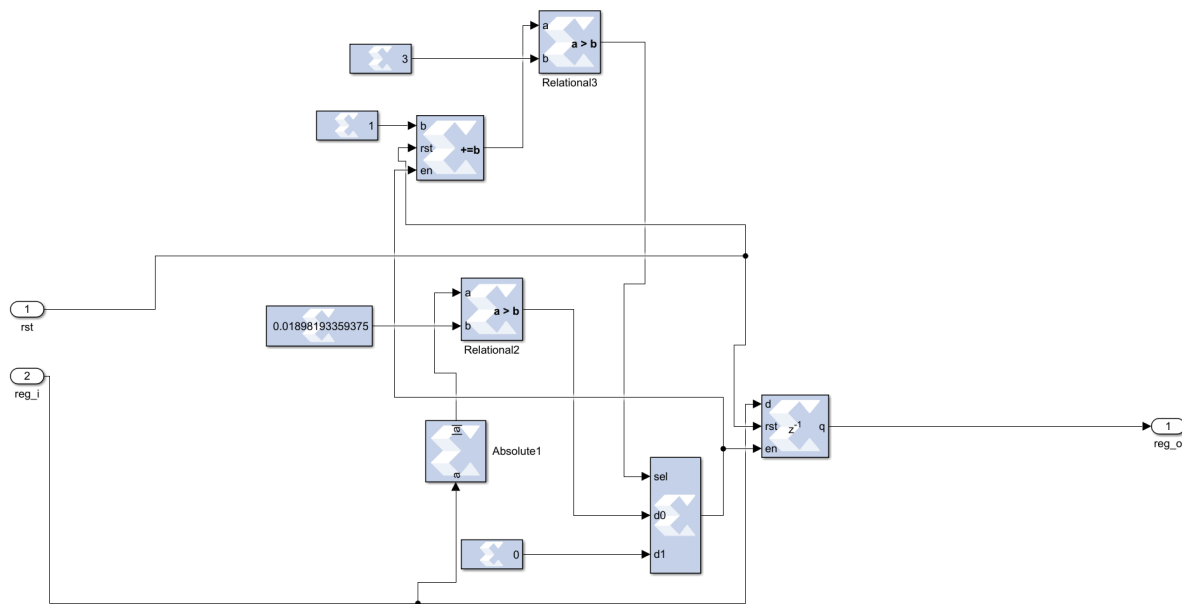
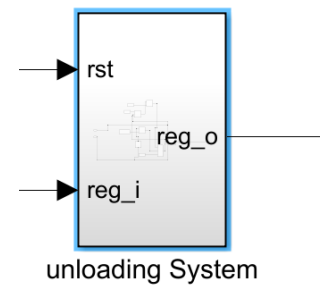
Also shown above is the rotation block which feeds back the Cosine for the next theta and a  $a_{i_j}$  value, the register saves that value for when it's done calculating that value and when it's time for a new matrix value, the register can be reset. The registers feed into a latch such that the value is saved and reading the output at a certain cycle would not be necessary.

For the Q values, an identity matrix will be fed but since these values are constant as input registers, we don't have to have the identity matrix as a separate matrix input and instead make it built in

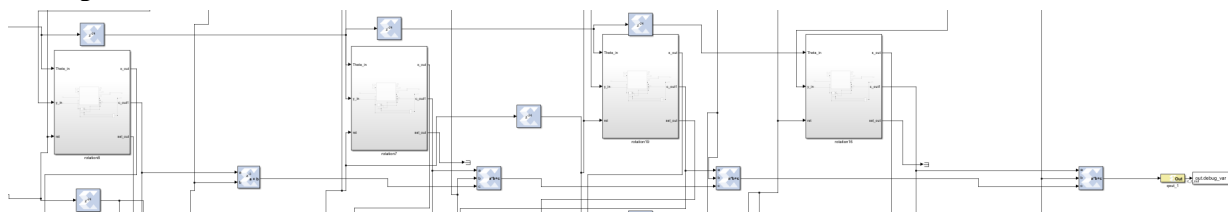


## Unloading Variables:

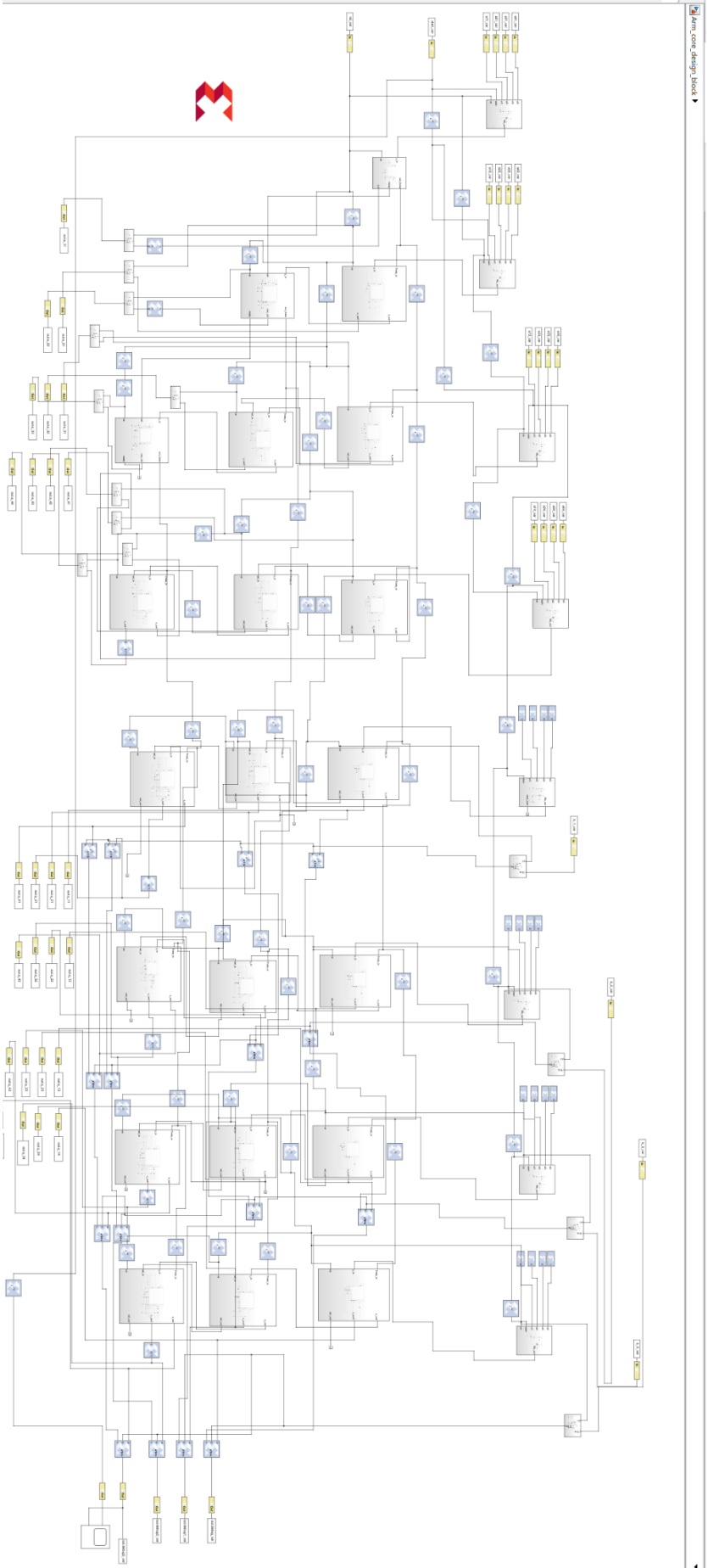
Values coming out of the cordic blocks will be latched into the unloading subsystem which takes the input  $reg\_i$ , which is the output of cordic. After a certain amount of changes of this value (for the case of values of R matrix is 4) the value will latch and only be reset after the REST signal is given. This ensures that the design can be read without having to be time / cycle dependent.



## Multiplication on FPGA



Values coming out of the cordic Q blocks will be latched and thus their outputs are fed into multipliers and mult and add blocks, meaning that as Q values are calculated, they are multiplied by the corresponding B values and then added, meaning that after the last Q value is calculated ( $q_{44}$ ) the delay to find the final resultant  $q^T b_4$  is only the latency of the last mult-add block. This makes it faster than putting the entire system through a matrix multiplier and uses less resources and saves time.



## Testing in Model Composer

**A1 Precalculation(matlab),FPGA Results (Hardware), Software results:**

```

A1 =
    0.1000    0.1000    0.4000    0.3000
    0.2000    0.8000    0.6000    0.5000
    0.9000    0.1000    0.3000    0.2000
    0.3000    0.1000    0.4000    0.6000

Q1 =
   -0.1026   -0.0911    0.7655   -0.6286
   -0.2052   -0.9691   -0.1370    0.0071
   -0.9234    0.2291   -0.2473   -0.1836
   -0.3078   -0.0110    0.5780    0.7557

r1 =
   -0.9747   -0.2975   -0.5643   -0.5027
         0   -0.7625   -0.5535   -0.4727
         0         0    0.3810    0.4585
         0         0         0    0.2317

b_l_disp =
    0.2000    0.1000    0.8000    0.4000

qTb_l =
   -0.9029    0.0638    0.1728    0.0304

x_ldisp =
    0.8034   -0.3796    0.2957    0.1311

```

---

```

-----Ai-----
{0.0999756, 0.0999756, 0.400024, 0.300049},

{0.199951, 0.800049, 0.599976, 0.5},

{0.900024, 0.0999756, 0.300049, 0.199951},

{0.300049, 0.0999756, 0.400024, 0.599976}}

-----Ri-----
{-0.972656, -0.295044, -0.560547, -0.499268},

{0.00000, -0.756836, -0.545532, -0.467285},

{0.00000, 0.00000, 0.383179, 0.45459},

{0.00000, 0.00000, 0.00000, 0.230957}}

-----Qi-----
{-0.0986328, -0.0891113, 0.761963, -0.620972},

{-0.201294, -0.959595, -0.137451, 0.00427246},

{-0.919189, 0.226929, -0.245728, -0.187744},

{-0.304565, -0.00695801, 0.567017, 0.752441}}

-----*Next Matrix*-----

b {0.199951, 0.0999756, 0.800049, 0.400024}

qTb {-0.897095, 0.0649414, 0.168823, 0.0269775}

X_v = {0.802246, -0.37561, 0.302124, 0.116699}

```

---

```

-----Next Matrix-----

R:
0.9747 0.2975 0.5643 0.5027
0.0000 0.7625 0.5535 0.4727
0.0000 -0.0000 0.3810 0.4585
0.0000 0.0000 -0.0000 -0.2317

Q:
0.1026 0.0911 0.7655 0.6286
0.2052 0.9691 -0.1370 -0.0071
0.9234 -0.2291 -0.2473 0.1836
0.3078 0.0110 0.5780 -0.7557

Qtb:
0.9029
-0.0638
0.1728
-0.0304

X_in:
0.8034
-0.3796
0.2957
0.1311

A?:
0.1000 0.1000 0.4000 0.3000
0.2000 0.8000 0.6000 0.5000
0.9000 0.1000 0.3000 0.2000
0.3000 0.1000 0.4000 0.6000

```

## A2 Precalculation(matlab),FPGA Results (Hardware), Software results:

A2 =

```
0.6900    0.5054    0.5914    0.5547
0.3401    0.8434    0.0687    0.4099
0.3784    0.2577    0.2073    0.6262
0.8799    0.3194    0.9805    0.0850
```

Q2 =

```
-0.5617   -0.0620   -0.1173   -0.8167
-0.2768   -0.9128    0.1901    0.2324
-0.3080   -0.0051   -0.8886    0.3398
-0.7162    0.4036    0.4006    0.4045
```

r2 =

```
-1.2285   -0.8255   -1.1173   -0.6788
0         -0.6736    0.2953   -0.3774
0          0         0.1523   -0.5095
0          0          0        -0.1106
```

b\_2\_disp =

```
0.4014    0.1995    0.1061    0.5954
```

qTb\_2 =

```
-0.7398    0.0328    0.1351   -0.0046
```

x\_2\_disp =

```
-0.6066    0.3776    1.0250    0.0413
```

-----Ai-----

```
{0.689941, 0.505371, 0.591431, 0.554688},
```

```
{0.340088, 0.84375, 0.0687256, 0.409912},
```

```
{0.378418, 0.25769, 0.207275, 0.626221},
```

```
{0.879883, 0.319458, 0.980469, 0.0849609}}
```

-----Ri-----

```
{-1.22571, -0.818359, -1.11157, -0.672974},
```

```
{0.00000, -0.671997, 0.297729, -0.37146},
```

```
{0.00000, 0.00000, 0.157227, -0.506714},
```

```
{0.00000, 0.00000, 0.00000, -0.121094}}
```

-----Qi-----

```
{-0.553223, -0.0579834, -0.107422, -0.812866},
```

```
{-0.273071, -0.905762, 0.184204, 0.227539},
```

```
{-0.30603, -0.00427246, -0.887939, 0.32019},
```

```
{-0.710205, 0.401611, 0.391968, 0.404541}}
```

-----"Next Matrix"-----

```
b {0.400024, 0.199951, 0.0999756, 0.599976}
```

```
qTb {-0.732666, 0.0361328, 0.140259, -0.00500488}
```

```
X_v = {-0.606567, 0.377563, 1.02502, 0.0412598}
```

-----Next Matrix-----

R:

```
1.2285 0.8256 1.1173 0.6788
```

```
0.0000 0.6740 -0.2953 0.3773
```

```
0.0000 -0.0000 0.1524 -0.5096
-0.0000 0.0000 -0.0000 0.1105
```

Q:

```
0.5617 0.0618 -0.1175 0.8166
```

```
0.2768 0.9129 0.1900 -0.2322
```

```
0.3080 0.0050 -0.8885 -0.3401
```

```
0.7162 -0.4035 0.4008 -0.4044
```

Qtb:

```
0.7406
```

```
-0.0343
```

```
0.1426
```

```
0.0036
```

X\_in:

```
-0.6259
```

```
0.3886
```

```
1.0442
```

```
0.0323
```

A?:

```
0.6900 0.5054 0.5914 0.5547
```

```
0.3401 0.8438 0.0687 0.4099
```

```
0.3784 0.2577 0.2073 0.6262
```

```
0.8799 0.3194 0.9805 0.0850
```

### A3 Precalculation(matlab),FPGA Results (Hardware), Software results:

```

A3 =
    0.1845    0.7026    0.7124    0.6811
    0.6496    0.9529    0.7813    0.9637
    0.1345    0.5616    0.1008    0.9517
    0.9629    0.2118    0.8349    0.4713

b_3_disp =
    0.3384    0.8989    0.7795    0.7040

Q3 =
   -0.1559   -0.5651    0.7623   -0.2745
   -0.5488   -0.4745   -0.2304    0.6486
   -0.1136   -0.4612   -0.5977   -0.6459
   -0.8134    0.4928    0.0929   -0.2947

r =
   -1.1838   -0.8685   -1.2304   -1.1265
         0   -1.0038   -0.4083   -1.0488
         0         0    0.3803   -0.2279
         0         0         0   -0.3155

qTb_3 =
   -1.2072   -0.6304   -0.3497   -0.2209

x_3_disp =
    0.8000    0.1000   -0.5000    0.7000

```

```

-----A-----
(0.184448, 0.702637, 0.712402, 0.681152),
(0.649658, 0.952881, 0.78125, 0.963745),
(0.134521, 0.561646, 0.10083, 0.95166),
(0.962891, 0.211792, 0.834961, 0.471313))
-----R-----
(-1.18018, -0.86084, -1.22253, -1.11877),
(0.00000, -0.997314, -0.401611, -1.04224),
(0.00000, 0.00000, 0.38208, -0.230469),
(0.00000, 0.00000, 0.00000, 0.305176))
-----Q-----
(-0.151733, -0.558716, 0.750244, 0.275391),
(-0.542725, -0.468994, -0.22168, -0.649048),
(-0.112183, -0.460083, -0.600098, 0.628662),
(-0.806519, 0.490112, 0.0882568, 0.29126))
-----"Next Matrix"-----
b (0.338379, 0.898926, 0.779541, 0.703979)
qTb (-1.19446, -0.62439, -0.351196, 0.204834)
X_v = (0.8125, 0.131836, -0.514282, 0.771143)

```

```

-----Next Matrix-----
R:
1.1838 0.8685 1.2304 1.1265
-0.0000 1.0038 0.4083 1.0488
-0.0000 0.0000 0.3803 -0.2279
-0.0000 0.0000 0.0000 0.3155

Q:
0.1559 0.5651 0.7623 0.2745
0.5488 0.4745 -0.2304 -0.6486
0.1136 0.4612 -0.5977 0.6459
0.8134 -0.4928 0.0929 0.2947

Qtb:
1.2072
0.6303
-0.3497
0.2209

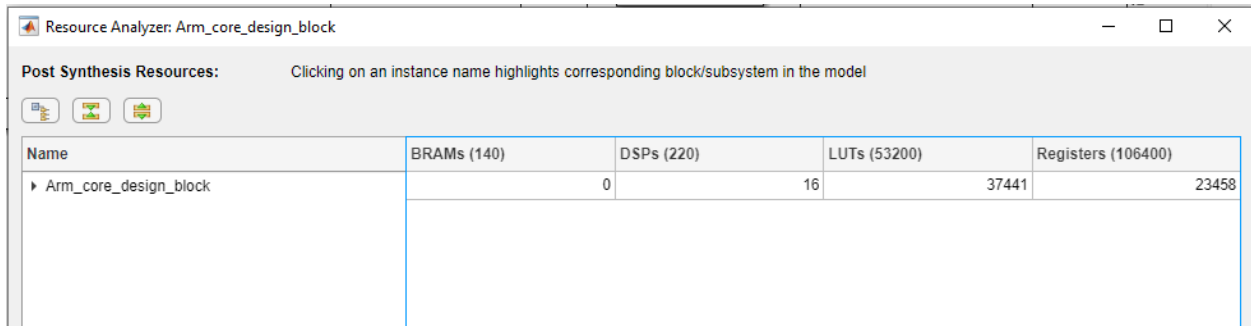
X_in:
0.8001
0.1000
-0.5000
0.6999

A?:
0.1845 0.7026 0.7124 0.6811
0.6496 0.9529 0.7813 0.9637
0.1345 0.5616 0.1008 0.9517
0.9629 0.2118 0.8349 0.4713

```

Now for model composer, I tested the matrices separately after the handling of the inputs and outputs. To test A1 you must run Final\_design\_inputs,m and A2 will be Final\_design\_inputsa2.m, to read a matrix after simulation is Final\_design\_disp,m

### Resource Usage:



Name	BRAMs (140)	DSPs (220)	LUTs (53200)	Registers (106400)
Arm_core_design_block	0	16	37441	23458

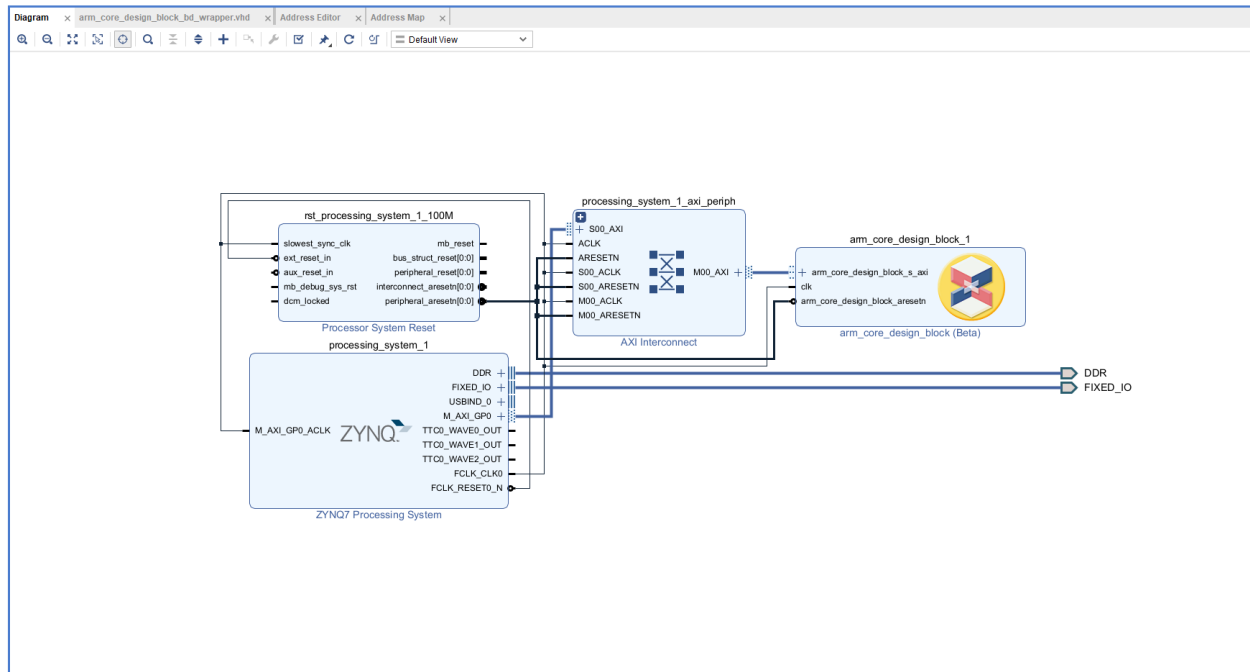
LUT: 37441

Reg: 23458

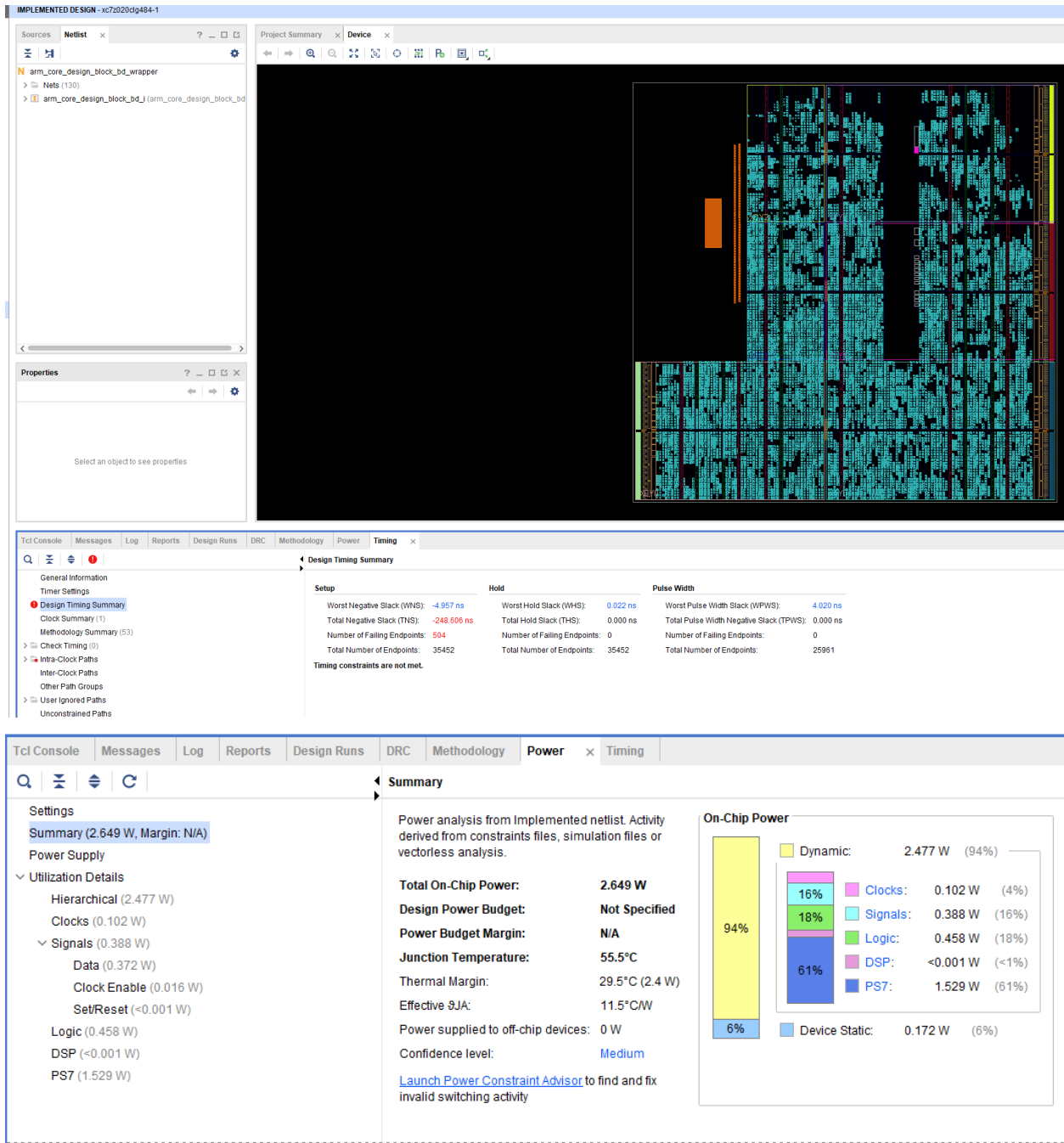


**Vivado:**

The workspace used for Vivado was Netlist\_proj6\_test7 which should have all that was included. The exported hardware is in vitis\_test6\_agan folder which has the vitis testbench



## Vivado Report



Report	Type	Options	Modified	Size
Design Initialization (init_design)				
Timing Summary - Design Initialization	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Opt Design (opt_design)				
DRC - Opt Design	report_drc		11/28/22, 4:47 PM	1.5 KB
Timing Summary - Opt Design	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Power Opt Design (power_opt_design)				
Timing Summary - Power Opt Design	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Place Design (place_design)				
IO - Place Design	report_io		11/28/22, 4:48 PM	146.6 KB
Utilization - Place Design	report_utilization		11/28/22, 4:48 PM	10.9 KB
Control Sets - Place Design	report_control_sets	verbose = true;	11/28/22, 4:48 PM	142.7 KB
Incremental Reuse - Place Design	report_incremental_reuse			
Incremental Reuse - Place Design	report_incremental_reuse			
Timing Summary - Place Design	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Post-Place Power Opt Design (post_place_power_opt_design)				
Timing Summary - Post-Place Power Opt Design	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Post-Place Phys Opt Design (phys_opt_design)				
Timing Summary - Post-Place Phys Opt Design	report_timing_summary	max_paths = 10; report_unconstrained = true;		
Route Design (route_design)				
DRC - Route Design	report_drc		11/28/22, 4:50 PM	1.5 KB
Methodology - Route Design	report_methodology		11/28/22, 4:50 PM	1.6 KB
Power - Route Design	report_power		11/28/22, 4:50 PM	10.1 KB
Route Status - Route Design	report_route_status		11/28/22, 4:50 PM	0.6 KB
Timing Summary - Route Design	report_timing_summary	max_paths = 10; report_unconstrained = true;	11/28/22, 4:50 PM	184.5 KB
Incremental Reuse - Route Design	report_incremental_reuse			
Clock Utilization - Route Design	report_clock_utilization		11/28/22, 4:50 PM	20.3 KB
Bus Skew - Route Design	report_bus_skew	warn_on_violation = true;	11/28/22, 4:50 PM	1.1 KB
Implementation Log			11/28/22, 4:51 PM	44.7 KB
Post-Route Phys Opt Design (post_route_phys_opt_design)				
Timing Summary - Post-Route Phys Opt Design	report_timing_summary	max_paths = 10; report_unconstrained = true; warn_on_violation = true;		
Bus Skew - Post-Route Phys Opt Design	report_bus_skew	warn_on_violation = true;		
Write Bitstream (write_bitstream)				
report_webtalk				
Implementation Log			11/28/22, 4:51 PM	44.7 KB

**Methodology**

impl\_1 (53 violations) (saved)

Tcl Console Messages Log x Reports Design Runs DRC Power Timing

synth\_1

Start Writing Synthesis Report

Report BlackBoxes:

BlackBox name	Instances
arm_core_design_block_bd_auto_pc_0	1
arm_core_design_block_bd_arm_core_design_block_1_0	1
arm_core_design_block_bd_processing_system_1_0	1
arm_core_design_block_bd_rst_processing_system_1_100M_0	1

Report Cell Usage:

Cell	Count
arm_core_design_block_bd_arm_core_design_block_1_0_bbox	1
arm_core_design_block_bd_auto_pc_0_bbox	1
arm_core_design_block_bd_processing_system_1_0_bbox	1
arm_core_design_block_bd_rst_processing_system_1_100M_0_bbox	1

Finished Writing Synthesis Report : Time (s): cpu = 00:00:23 ; elapsed = 00:00:23 . Memory (MB): peak = 1312.441 ; gain = 29.312

Synthesis finished with 0 errors, 0 critical warnings and 5 warnings.

Synthesis Optimization Runtime: Time (s): cpu = 00:00:15 ; elapsed = 00:00:21 . Memory (MB): peak = 1312.441 ; gain = 29.312

Synthesis Optimization Complete: Time (s): cpu = 00:00:23 ; elapsed = 00:00:23 . Memory (MB): peak = 1312.441 ; gain = 29.312

INFO: [Project 1-571] Translating synthesized netlist

Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.010 . Memory (MB): peak = 1312.441 ; gain = 0.000

INFO: [Project 1-570] Preparing netlist for logic optimisation

INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 1330.141 ; gain = 0.000

INFO: [Project 1-111] Unisim Transformation Summary:

No Unisim elements were transformed.

Synth Design complete, checksum: ee501797

INFO: [Common 17-83] Releasing license: Synthesis

52 Infos, 13 Warnings, 0 Critical Warnings and 0 Errors encountered.

synth\_design completed successfully

synth\_design: Time (s): cpu = 00:00:26 ; elapsed = 00:00:27 . Memory (MB): peak = 1343.156 ; gain = 60.027

INFO: [Common 17-1381] The checkpoint 'C:/EIE/C52/netlist\_proj6\_test7/ip\_catalog/arm\_core\_design\_block\_runs/synth\_1/arm\_core\_design\_block\_bd\_wrapper.dcp' has been generated.

INFO: [runtcl-4] Executing: report\_utilization -file arm\_core\_design\_block\_bd\_wrapper\_utilization\_synth.rpt -pb arm\_core\_design\_block\_bd\_wrapper\_utilization\_synth.pb

INFO: [Common 17-206] Exiting Vivado at Tue Dec 13 03:11:47 2022...

Synthesis Implementation Simulation

**Arm Code Results:**

In the Arm code, my code has a design lock which will be fed a matrix, and then unload this matrix into each port of the Axi lite port and into my design, once all variables are fed into design calculation starts. Once a done variable is asserted my design unloads the variables and starts back substitution. Back substitution uses the following equation:

$$\begin{aligned}
 x_1 &= \frac{Q^T b_1 - r_{1,2} \left( \frac{Q^T b_2 - r_{2,3} \left( \frac{Q^T b_3 - r_{3,4} \frac{Q^T b}{r_{4,4}} \right) - r_{2,4} \frac{Q^T b_4}{r_{4,4}}}{r_{3,3}} \right) - r_{1,3} \left( \frac{Q^T b_3 - r_{3,4} \frac{Q^T b}{r_{4,4}}}{r_{3,3}} \right) - r_{1,4} \left( \frac{Q^T b_4}{r_{4,4}} \right)}{r_{1,1}} \\
 x_2 &= \frac{Q^T b_2 - r_{2,3} \left( \frac{Q^T b_3 - r_{3,4} \frac{Q^T b}{r_{4,4}}}{r_{3,3}} \right) - r_{2,4} \frac{Q^T b_4}{r_{4,4}}}{r_{2,2}} \\
 x_3 &= \frac{Q^T b_3 - r_{3,4} \frac{Q^T b}{r_{4,4}}}{r_{3,3}} \\
 x_4 &= \frac{Q^T b_4}{r_{4,4}}
 \end{aligned}$$

Note that for  $x_1, x_2, x_3$ , if any values are from the previous equation, they will be used as to compute less times

**Software Implementation:**

I used the QR Decomposition via Givens Rotations algorithm from John Russell Strauss (from Canvas of 522 provided from Prof) and implemented it in C++ for this project. I had to make several functions for this implementation to work such as matrix multiplication and vector multiplication.

**Main Functions Made:**

- hardware\_des(Array) - For calculating the values and displaying
- hardware\_des\_time(Array) - For calculating without print statements and for seeing time it takes to calculate values
- sw\_des(Array) - For calculating the values and displaying
- sw\_des\_time(Array) - For calculating without print statements and for seeing time it takes to calculate values for software

For One system of equation:

```
Done, Total time steps HARDWARE from start to finish = 1535189
Cycle counts per second for HARDWARE = 3.33333e+08
Time in seconds for PL hardware add = times steps / COUNTS_PER_SECOND = 0.00460557
Done, Total time steps SOFTWARE from start to finish = 27185
Cycle counts per second for SOFTWARE = 3.33333e+08
Time in seconds for PL hardware add = times steps / COUNTS_PER_SECOND = 8.1555e-05
```

For All Three system of equations:

```
Done, Total time steps HARDWARE from start to finish = 4520558
Cycle counts per second for HARDWARE = 3.33333e+08
Time in seconds for PL hardware add = times steps / COUNTS_PER_SECOND = 0.0135617
Done, Total time steps SOFTWARE from start to finish = 79827
Cycle counts per second for SOFTWARE = 3.33333e+08
Time in seconds for PL hardware add = times steps / COUNTS_PER_SECOND = 0.000239481
```

It seems that the John Straus QR decomposition works faster as it was optimized compared to my hardware implementation. I think this may have to do with how my variables are loaded and unloaded. If possible, I would optimize my QR rotation block to take less cycles (currently 25) and figure out a faster way to load and unload variables. My design does create accurate and real results meaning that I must focus on optimizing what I already have.

**Conclusion:**

My design for software and Hardware work however the hardware implementation could be more efficient as it is slower then the software implementation. The tradeoffs between the two versions are timing and resource use as hardware could be faster but has to have its dedicated components which use more power compared to just a software approach which may use less but take more cycles to complete. My hardware uses FPGA multiplication which saves time and makes design more efficient.