



Hardware Accelerator for QR Decomposition

Gayathri Vadhyan, Shreya Mahetaliya

Dept. of Electrical and Computer Engineering, Box 352500
University of Washington
Seattle, Washington 98195-2500 U.S.A

June 7, 2024

Introduction.....	3
--------------------------	----------

Choice of Gram Schmidt Algorithm.....	3
Usefulness Angle.....	4
Novelty Angle.....	4
Background and Related Works.....	4
Final Design Results.....	5
Final Specification.....	5
Final Deliverable.....	5
Working.....	6
Implementation results.....	7
Die photo.....	10
Key Metrics.....	11
Hardware Simulation Results.....	16
Verification Results.....	16
Realistic Factors.....	17
Public Health, Safety, and Welfare.....	17
Global factors.....	17
Cultural factors.....	18
Social factors.....	18
Environmental factors.....	18
Economic factors.....	18
Link to our repository.....	18
Quality Metrics.....	18
Agile Milestones.....	19
Limitations.....	20
Future Works.....	20

Introduction

QR decomposition is a fundamental numerical algorithm used for various applications, including solving simultaneous equations and least squares problem. It decomposes a matrix into the product of an orthogonal matrix (Q) and an upper triangular matrix (R). In simpler terms, it breaks down a given matrix into two matrices, one representing the rotation or reflection (Q) and the other representing scaling and shearing (R).

Choice of Gram Schmidt Algorithm:

Our project stands out by adopting the Modified Gram Schmidt algorithm for QR decomposition. The Gram Schmidt algorithm offers distinct advantages compared to traditional methods like the Householder algorithm which involves more complex arithmetic operations, leading to higher computational overheads and memory requirements. Known for its superior sequential efficiency, the Gram-Schmidt algorithm excels in performing QR decomposition for smaller matrices with fewer arithmetic operations. However, one of its inherent limitations lies in its lack of inherent parallelism, which can hinder performance on modern parallel computing architectures.

In the process of orthogonalizing matrix $A = [a \ b \ c \ d]$ using Gram-Schmidt Process, we follow a systematic procedure to transform its columns into orthonormal vectors. Initially, the first column of A is normalized to obtain q_1 , which is calculated as $q_1 = a / \text{norm}(a)$, ensuring that q_1 is a unit vector in the direction of the first column of A. Subsequently, we make the second column of A orthogonal to q_1 by subtracting its projection onto q_1 from itself, denoted as $b = b - (q_1^T b) q_1$, and then normalize this orthogonal component to obtain q_2 as $q_2 = b / \text{norm}(b)$. This process ensures that q_2 is orthogonal to q_1 . We extend this approach to make the third column of A orthogonal to both q_1 and q_2 . We subtract the projections of the third column onto q_1 and q_2 from itself, denoted as $c = c - (q_1^T c) q_1 - (q_2^T c) q_2$, and then normalize this orthogonal component to obtain q_3 as $q_3 = c / \text{norm}(c)$. This ensures that q_3 is orthogonal to both q_1 and q_2 . Extending this approach, q_4 can be determined from $d / \text{norm}(d)$ where $d = d - (q_1^T d) q_1 - (q_2^T d) q_2 - (q_3^T d) q_3$. This process can be iterated for additional columns if present in A, providing a set of orthonormal vectors that form the columns of matrix Q.

In our computational approach, we've adopted the Modified Gram-Schmidt algorithm due to its ability to produce more reliable values for Q. The initial steps in computing q_1 and q_2 remain consistent with those of the classical Gram-Schmidt process: $q_1 = a / \|a\|$ and $q_2 = B / \|B\|$ where $B = b - (q_1^T b) q_1$. However, for q_3 , instead of directly computing it as $q_3 = c - q_1^T c q_1 - q_2^T c q_2$, we've introduced intermediate steps. We define $Z(1) = c - q_1^T c q_1$ and $Z(2) = Z(1) - q_2^T Z(1) q_2$, and then compute $q_3 = Z(2) / \|Z(2)\|$. This modified process results in more robust values for Q, enhancing the overall accuracy of our computations.

Usefulness Angle

QR decomposition is a fundamental numerical algorithm extensively used in scientific and engineering applications, including solving simultaneous equations, least squares problems, and

eigenvalue computations. Our project aims to develop an open-source QR decomposition module that offers superior performance in terms of area and power efficiency. By addressing the challenges associated with large-scale datasets and real-time processing requirements, we seek to enhance computational efficiency and provide faster, more accurate solutions for various numerical problems. This effort will facilitate advancements in fields such as signal processing, data analysis, and scientific computing. A significant aspect of our project is the decision to target an Application-Specific Integrated Circuit (ASIC) implementation, which is tailored for specific scientific and engineering applications to ensure optimal performance and resource utilization.

Novelty Angle

Our project stands out due to its unique approach to developing an efficient and scalable QR decomposition module. Unlike existing references, such as the whitepaper “Advanced QRD Optimization with Intel® HLS Compiler” and the paper “FPGA Implementation of Gram-Schmidt QR Decomposition Using High Level Synthesis,” which use High-Level Synthesis (HLS) for FPGA implementations, our approach involves creating an ASIC. This choice is expected to deliver superior performance, power efficiency, and area optimization. Additionally, we aim to open-source the Register-Transfer Level (RTL) code, providing the community with access to a high-performance QR decomposition module that can be further developed and optimized. This novel approach ensures that our solution is not only highly efficient and scalable but also accessible for broader use and further innovation.

Background and Related Works

In developing our open-source QR decomposition module, we conducted thorough research to gather relevant reference materials and understand existing methodologies in QR decomposition and hardware acceleration. Key references include:

1. **Advanced QRD Optimization with Intel® HLS Compiler:** This whitepaper provided insights into implementing QR decomposition using the Modified Gram-Schmidt (MGS) algorithm with High-Level Synthesis (HLS). It offered valuable knowledge on optimizing QR decomposition for FPGA platforms using HLS tools.
2. **FPGA Implementation of Gram-Schmidt QR Decomposition Using High Level Synthesis:** This paper detailed the implementation of the Gram-Schmidt algorithm on FPGAs using HLS. It helped us grasp the practical aspects of translating high-level algorithmic descriptions into hardware implementations.
3. **Textbooks and Academic Journals:** We reviewed several textbooks and peer-reviewed journals on numerical linear algebra and digital design, providing us with a solid theoretical foundation in QR decomposition algorithms and hardware design principles.

We effectively applied the knowledge acquired from these resources to our project:

1. **Comparing Approaches:** By analyzing methodologies from the whitepaper and the paper, we compared the efficiency, scalability, and feasibility of different QR decomposition algorithms and their hardware implementations. This analysis guided our decision to develop an ASIC implementation for better performance and resource efficiency.

2. **Adopting Reliable Techniques:** We adopted the Modified Gram-Schmidt (MGS) algorithm, recognized for its numerical stability, from the reviewed literature. We adapted it for ASIC design to leverage the full potential of custom hardware.
3. **Discriminating by Reliability:** We evaluated the reliability of resources based on their publication sources, the credibility of authors, and the technical depth of their methodologies. This ensured our design choices were based on robust and validated information.
4. **Integrating Knowledge:** Using insights gained from the literature, we structured our project to include fundamental vector operations, ensuring correctness and efficiency before progressing to more complex QR factorization methods.

By integrating these diverse sources of knowledge into our project, we demonstrated the ability to acquire, apply, and discriminate among relevant external resources to create a highly efficient and scalable QR decomposition module.

Final Design Results

1. Final Specification

Final Deliverable

1. The final deliverable of our project is an open-source hardware accelerator for QR decomposition, implemented as an ASIC (Application-Specific Integrated Circuit). The key components and functionality include. Our entire design was divided into a datapath and a controller, which managed the flow of data and control signals to ensure the correct execution of the QR decomposition algorithm. The different modules designed were:
 1. **Dot Product module:** Designed to calculate the dot product of two vectors.
 2. **Normalization module:** Designed a module that would take a vector, square and add each element (accum), then take a square root of that sum ($\text{accum}^{0.5}$) and then divide one by that value ($1/\text{accum}^{0.5}$)
 3. **Multiplication module:** These were designed with internal counters to ensure that results are being directed to the right output since values are entering into the module only one at a time
 4. **Subtraction module:** These were designed with internal counters to ensure that results are being directed to the right output since values are entering into the module only one at a time
 5. **Buffers:** Three types of buffers have been designed for storing and sending data. Since we are storing and reading one vector at a time, there are internal counters that keep track and control how many values are being read and written in one read and write cycle. The types of buffer are:
 1. Input buffer: Stores the input matrix (A)
 2. Intermediate buffer: They store the intermediate vector values for the multipliers, dot product module and the subtractor module.

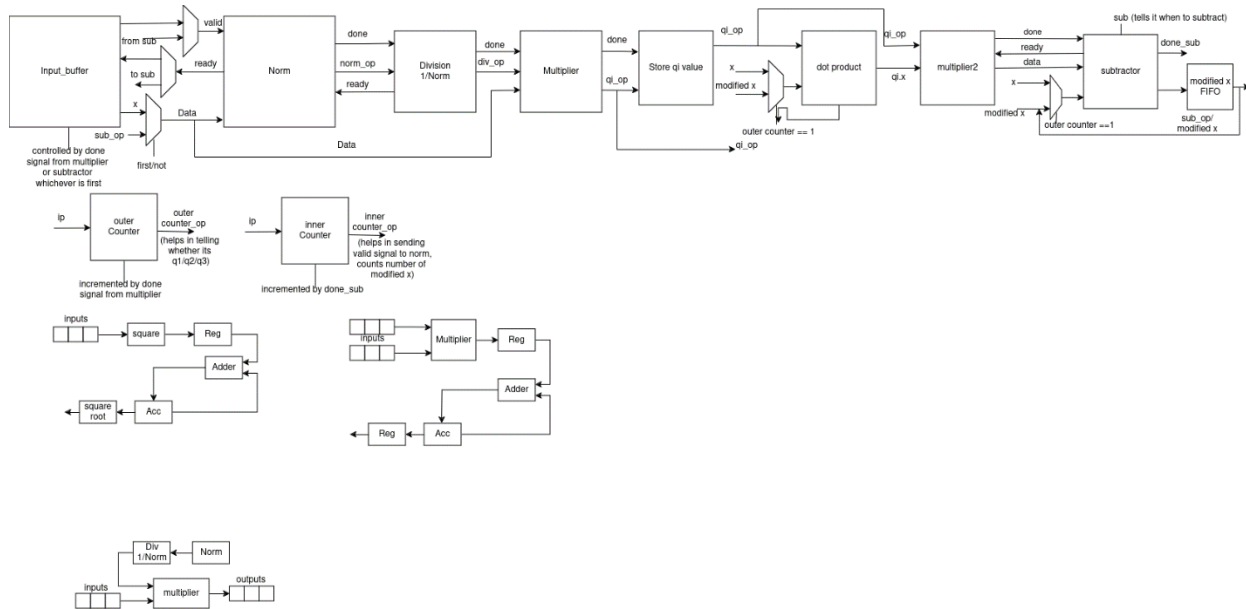


Fig 2: Architecture of the complete QR decomposition module, including design of lower modules like the norm unit and the dot product module

Working:

The inner counter plays a crucial role in managing modified inputs generated after subtraction and determines which of these inputs should be forwarded to the normalization unit for final orthogonal vector computation. Specifically, when the inner counter value is 1, the output of the subtractor is directed to the normalization module. Designed initially for a 3x3 matrix with 16-bit fixed-point elements (8 bits for integers and 8 bits for decimals), the architecture is easily customizable for different matrix sizes and data formats. The process begins with loading 9 input elements into the input buffer. Computation for the first orthogonal vector involves taking the first column (or the first 3 values in the input buffer), computing their norm value, and then obtaining its reciprocal using a divider. This reciprocal value is then multiplied with the first column to derive the first orthogonal vector. Subsequent orthogonal vector calculations (q2 and q3) depend on the values of the first orthogonal vector, stored in registers and accessed by the dot product module. The dot product module computes dot products between previous orthogonal vector values and either other input columns or modified inputs generated downstream in the design, with computation order determined by values provided by the two counters. Dot product values are further multiplied with previous orthogonal vectors and then subtracted from input or modified input vectors, with results stored in the modified buffer. Depending on the inner counter values, the normalization unit either processes values from the modified buffer to derive further orthogonal vectors or uses them as modified inputs for the dot product and subtraction units. This intricate process ensures efficient computation of QR decomposition while allowing for scalability and customization to meet various matrix requirements. Additionally, in our design, values from the modified buffer are directed to the normalization unit for final output vector calculation if the inner counter value is 1; otherwise, they are utilized for further computation. This approach aligns with our modified Gram-Schmidt algorithm, which relies on modified input values for orthogonal vector derivation. Furthermore, the outer counter increments once the inner counter reaches a fixed value, managing the progression of computation stages efficiently. The entire computation concludes when the outer counter equals the number of vectors in the input, providing a clear termination condition for the algorithm. This comprehensive design strategy facilitates the effective computation of QR

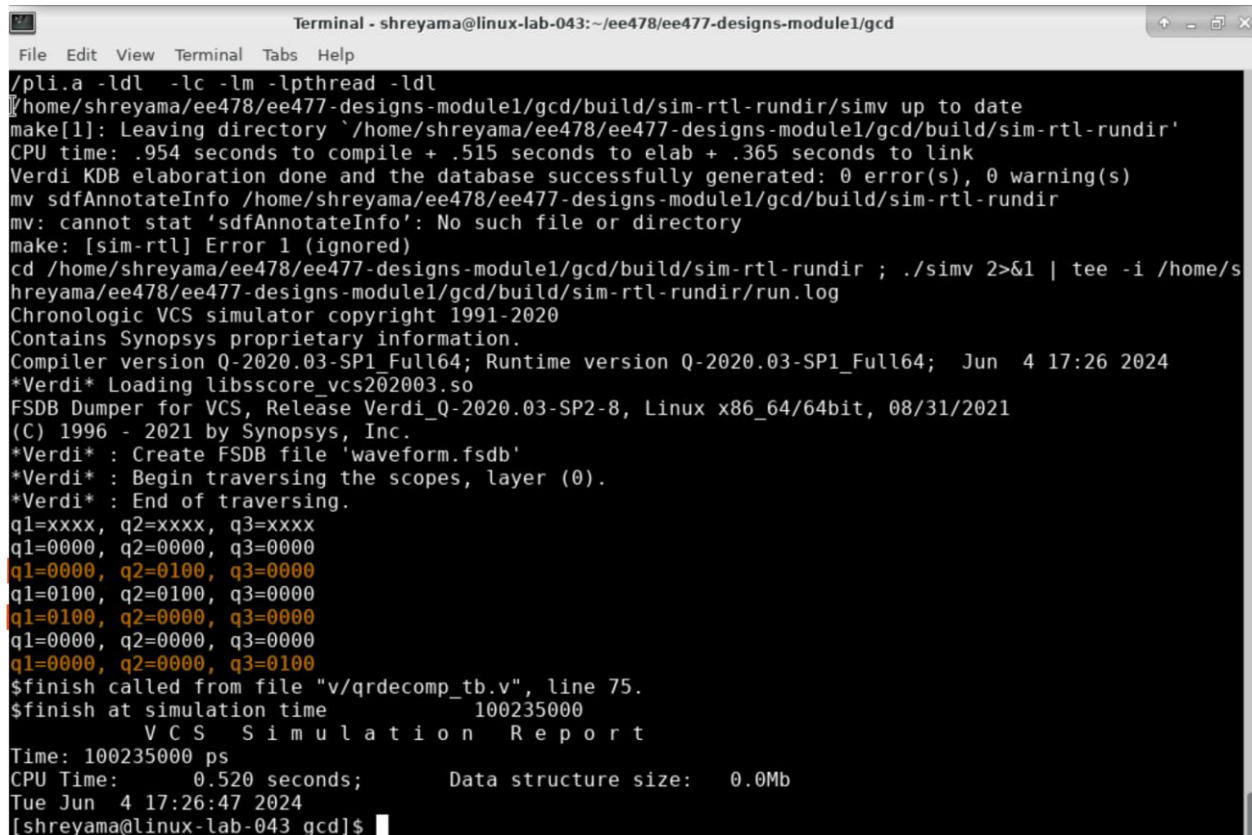
decomposition while ensuring scalability and adaptability to varying input matrix sizes and data formats.

2. Implementation results:

The design was synthesized and laid out from RTL (Register-Transfer Level) to GDSII (Graphic Data System II) using standard VLSI design tools, ensuring optimal performance in terms of area, power, and speed.

The matrix below was used as input. A simple matrix was used, since our module currently requires positive values for A and Q. Below, the screenshots of results, observations, and comparisons of synthesis and place-and-route (PAR) can be found

$$A = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$



```
Terminal - shreyama@linux-lab-043: ~/ee478/ee477-designs-module1/gcd
File Edit View Terminal Tabs Help
/pli.a -ldl -lc -lm -lpthread -ldl
/home/shreyama/ee478/ee477-designs-module1/gcd/build/sim-rtl-rundir/simv up to date
make[1]: Leaving directory `/home/shreyama/ee478/ee477-designs-module1/gcd/build/sim-rtl-rundir'
CPU time: .954 seconds to compile + .515 seconds to elab + .365 seconds to link
Verdi KDB elaboration done and the database successfully generated: 0 error(s), 0 warning(s)
mv sdfAnnotateInfo /home/shreyama/ee478/ee477-designs-module1/gcd/build/sim-rtl-rundir
mv: cannot stat 'sdfAnnotateInfo': No such file or directory
make: [sim-rtl] Error 1 (ignored)
cd /home/shreyama/ee478/ee477-designs-module1/gcd/build/sim-rtl-rundir ; ./simv 2>&1 | tee -i /home/s
hreyama/ee478/ee477-designs-module1/gcd/build/sim-rtl-rundir/run.log
Chronologic VCS simulator copyright 1991-2020
Contains Synopsys proprietary information.
Compiler version Q-2020.03-SP1_Full64; Runtime version Q-2020.03-SP1_Full64; Jun 4 17:26 2024
*Verdi* Loading libsscore_vcs202003.so
FSDB Dumper for VCS, Release Verdi_Q-2020.03-SP2-8, Linux x86_64/64bit, 08/31/2021
(C) 1996 - 2021 by Synopsys, Inc.
*Verdi* : Create FSDB file 'waveform.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
q1=xxxx, q2=xxxx, q3=xxxx
q1=0000, q2=0000, q3=0000
q1=0000, q2=0100, q3=0000
q1=0100, q2=0100, q3=0000
q1=0100, q2=0000, q3=0000
q1=0000, q2=0000, q3=0000
q1=0000, q2=0000, q3=0100
$finish called from file "v/qrdecomp_tb.v", line 75.
$finish at simulation time 100235000
V C S S i m u l a t i o n R e p o r t
Time: 100235000 ps
CPU Time: 0.520 seconds; Data structure size: 0.0Mb
Tue Jun 4 17:26:47 2024
[shreyama@linux-lab-043 gcd]$
```

Fig 3: RTL simulation showing the q output for the first A matrix

The Q that we got is the following:

$$Q = \begin{bmatrix} 0 & 0100 & 0 \\ 0100 & 0 & 0 \\ 0 & 0 & 0100 \end{bmatrix}$$

The output is in hex format where first 8 bits are integers, and the next 8 bits are decimal numbers.

When we use $A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$, $q1$ and $q2$ vectors come out to be accurate, but since $q3$ comes out to be negative, the values do not come out as expected

If $Q = [q1 \ q2 \ q3]$, the approximated values of $q1 = [0.25 \ 0.5 \ 0.75]$, $q2 = [1 \ 0.5 \ 0]$, $q3 = [-0.875 \ -0.625 \ -0.375]$

As one can see in the results below, the values of $q1$ and $q2$ are correct, but $q3$ gives weird results due to the presence of negative values.

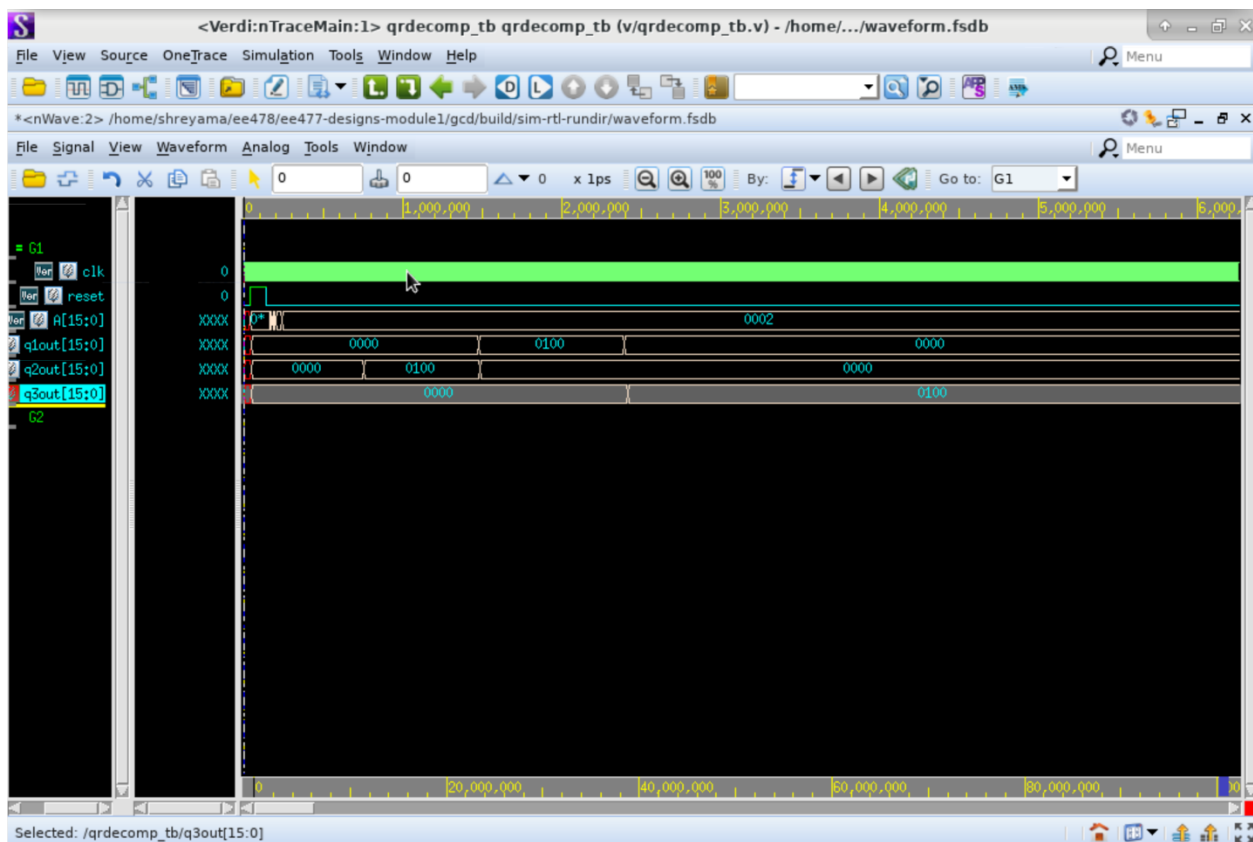


Fig 4: RTL simulation showing the q output on VCS for the first A vector

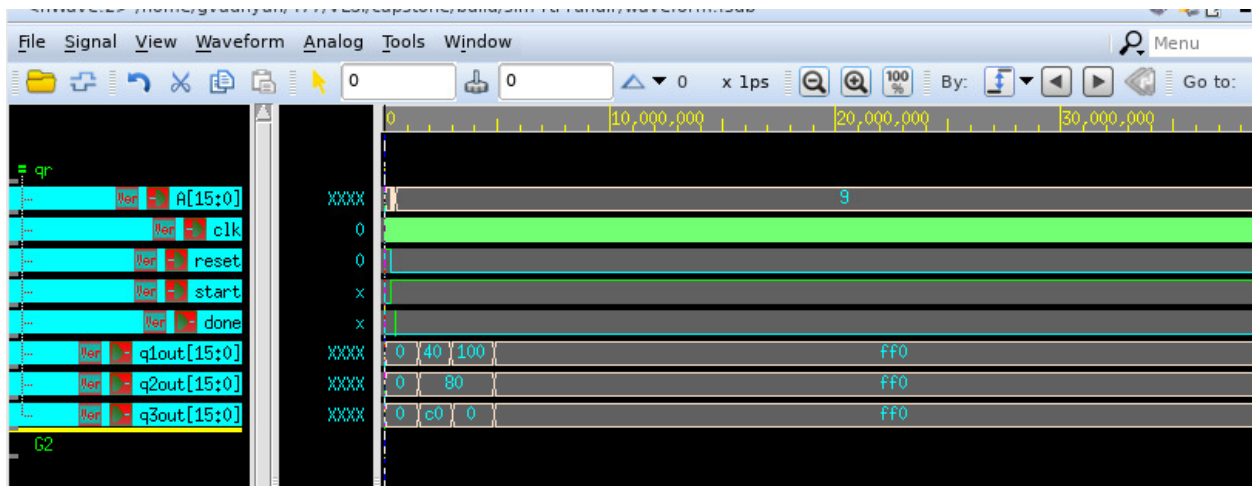


Fig 5: RTL simulation showing the q output on VCS for the second A vector

a. Die photo (screenshot):

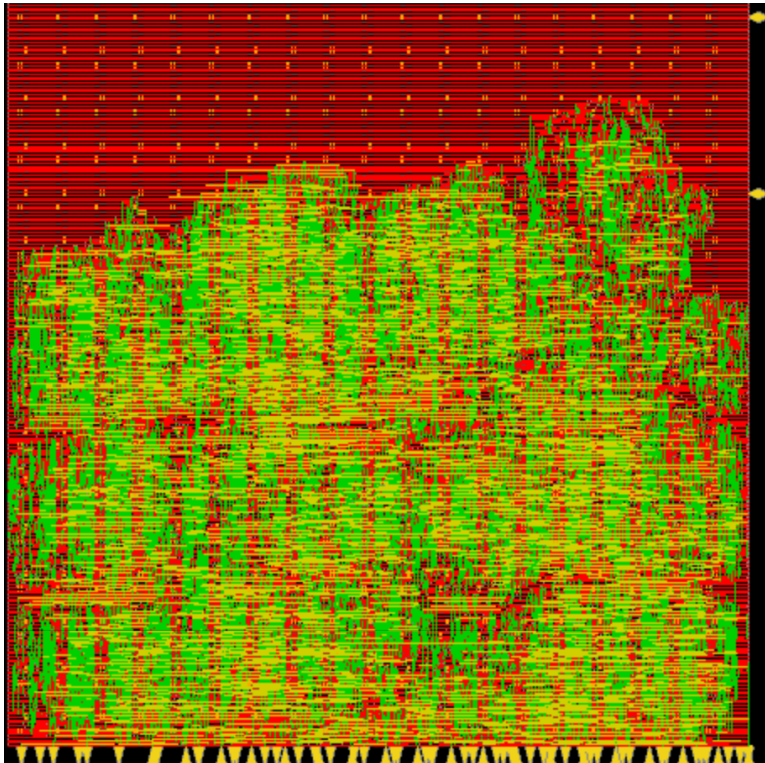


Fig 6: Die photo after place and route

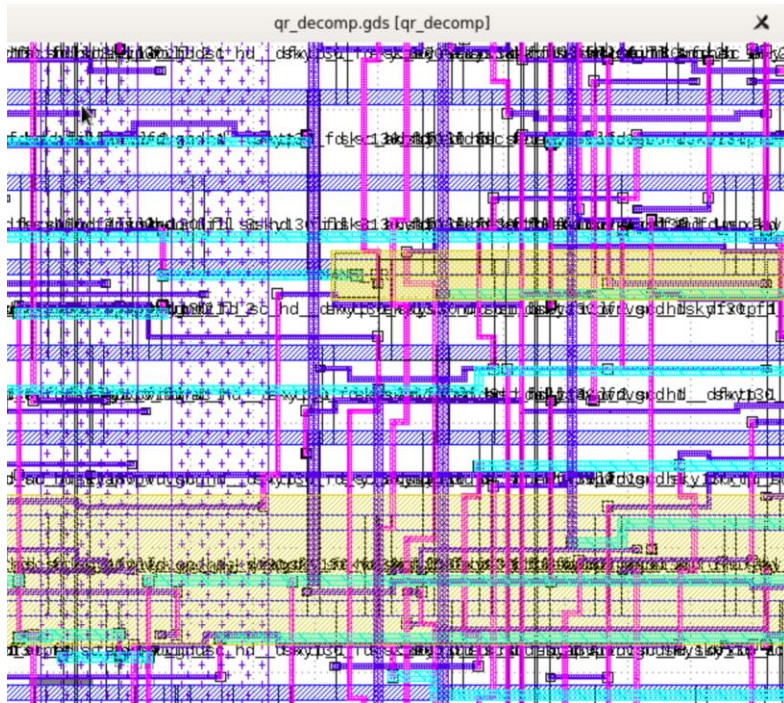


Fig 7: Top level GDS view

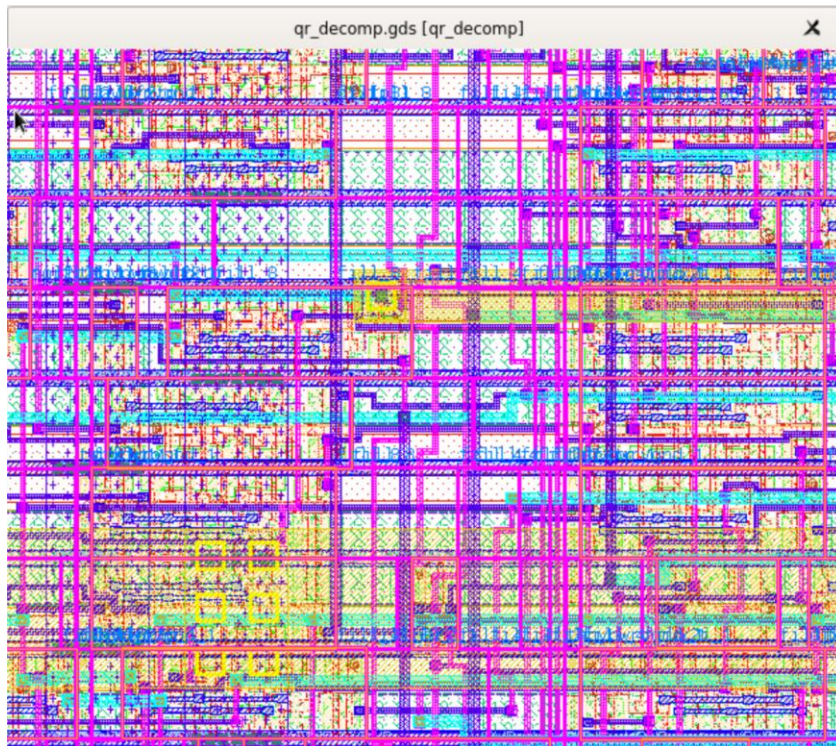


Fig 8: Full hierarchy GDS view

b. Key metrics:

```
## constraints.tcl
#
# This file is where design timing constraints are defined for Genus and Innovus.
# Many constraints can be written directly into the Hammer config files. However,
# you may manually define constraints here as well.
#
# TODO: add constraints here!
create_clock -name clk -period 9 [get_ports clk]
set_clock_uncertainty 0.50 [get_clocks clk]

set_input_delay 4.5 -max -clock [get_clocks clk] [all_inputs]
set_output_delay 4.5 -max -clock [get_clocks clk] [remove_from_collection [all_outputs] [get_ports clk_o]]

set_input_delay 0.0 -min -clock [get_clocks clk] [all_inputs]
set_output_delay 0.0 -min -clock [get_clocks clk] [remove_from_collection [all_outputs] [get_ports clk_o]]
```

Fig 9: Constraints used for synthesis

```
1 =====
2 Generated by:      Genus(TM) Synthesis Solution 21.12-s068_1
3 Generated on:      Jun 04 2024 08:34:59 pm
4 Module:            qr_decomp
5 Operating conditions: ss_100C_lv60
6 Interconnect mode: global
7 Area mode:         physical library
8 =====
9
10
11 Path 1: MET (0 ps) Setup Check with Pin d1/dot_p/A_r_reg[19]/CLK->D
12 View: ss_100C_lv60.setup_view
13 Group: clk
14 Startpoint: (R) d1/buffer_dot/in_buffer/c_unit/rd_ptr_reg[2]/CLK
15 Clock: (R) clk
16 Endpoint: (F) d1/dot_p/A_r_reg[19]/D
17 Clock: (R) clk
18
19      Capture      Launch
20      Clock Edge:+ 9000      0
21      Src Latency:+ 0      0
22      Net Latency:+ 0 (I)    0 (I)
23      Arrival:= 9000      0
24
25      Setup:- 300
26      Required Time:= 8700
27      Launch Clock:- 0
28      Data Path:- 8700
29      Slack:= 0
30
31 #-----
32 # Timing Point      Flags  Arc  Edge  Cell
33 # Fanout Load Trans Delay Arrival Instance
```

Fig 10: The setup corner timing report after synthesis

Open

final_area.rpt

~\ee478\ee477-designs-module1\gcd\build\syn-rundir\reports

Save

-

x

1=====

2Generated by:Genus(TM) Synthesis Solution 21.12-s068_1

3Generated on:Jun 04 2024 08:34:59 pm

4Module:qr_decomp

5Library domain:ss_100C_lv60.setup_cond

6Domain index:0

7Technology library:sky130_fd_sc_hd_ss_100C_lv60 1.0000000000

8Operating conditions:ss_100C_lv60

9Interconnect mode:global

10Area mode:physical library

11=====

12

13InstanceModuleCell CountCell AreaNet AreaTotal Area

14-----

15qr_decomp9225102603.40548869.811151473.216

16c1controller86546.774349.904896.678

17d1datapath_DATA_WIDTH16_ADDR_WIDTH49136102045.37047953.153149998.522

18buffer_dotbuffer_3_DATA_WIDTH16_ADDR_WIDTH4_11291910831.6384202.23715033.875

19in_bufferfifo_DATA_WIDTH16_ADDR_WIDTH4_11688210497.5684040.46814538.036

20c_unitfifo_ctrl_ADDR_WIDTH4_VECTOR_LEN1_12064646.870244.785891.655

21r_unitreg_file_DATA_WIDTH16_ADDR_WIDTH4_1258149831.9303670.72513502.654

22buffer_normbuffer_3_DATA_WIDTH16_ADDR_WIDTH47339671.7763435.42513107.201

23in_bufferfifo_DATA_WIDTH16_ADDR_WIDTH4_1186969373.9903281.92312655.914

24c_unitfifo_ctrl_ADDR_WIDTH4_VECTOR_LEN1_12261613.088213.110826.198

25r_unitreg_file_DATA_WIDTH16_ADDR_WIDTH4_1276318742.1342927.32111669.455

26buffer_subbuffer_3_DATA_WIDTH16_ADDR_WIDTH4_1116718528.1793228.32011756.499

27in_bufferfifo_DATA_WIDTH16_ADDR_WIDTH4_1156378247.9103084.99911332.910

28c_unitfifo_ctrl_ADDR_WIDTH4_VECTOR_LEN1_11962605.581219.810825.391

29r_unitreg_file_DATA_WIDTH16_ADDR_WIDTH4_1245717623.5622713.77610337.338

30dot_pdot_product8989776.8774615.40314392.280

31ibinput_buffer_DATA_WIDTH16_ADDR_WIDTH46869190.0643287.49312477.557

32in_bufferfifo_DATA_WIDTH16_ADDR_WIDTH46528823.4623135.90511959.368

33c_unitfifo_ctrl_ADDR_WIDTH4_VECTOR_LEN160606.832211.369818.201

34r_unitreg_file_DATA_WIDTH16_ADDR_WIDTH45888197.8622784.69710982.559

Plain Text

Tab Width: 8

Ln 1, Col 1

IN

Fig 11: The area report after synthesis

Observations after synthesis are as follows:

Maximum frequency : 111.11 MHz.

Total area: 151473.216 um²

Open ▾

qr_decomp_postRoute_all.tarpt

~\ee478\ee477-designs-module1\gcd\build\par-rundir\timingReports

Save

-

▢

×

1 #####

2 # Generated by: Cadence Innovus 21.13-s100_1

3 # OS: Linux x86_64(Host ID linux-lab-043.ece.uw.edu)

4 # Generated on: Tue Jun 4 21:40:32 2024

5 # Design: qr_decomp

6 # Command: opt design -post_route -setup -hold -expanded_views

7 #####

8 Path 1: MET (0.054 ns) Setup Check with Pin d1/muld/d2_mul_temp_reg[23]/CLK->D

9 View: ss_100C_lv60.setup_view

10 Group: reg2reg

11 Startpoint: (R) d1/lq/q2_l_r_reg[9]/CLK

12 Clock: (R) clk

13 Endpoint: (R) d1/muld/d2_mul_temp_reg[23]/D

14 Clock: (R) clk

15

16 Capture Launch

17 Clock Edge:+ 12.000 0.000

18 Src Latency:+ -1.316 -1.316

19 Net Latency:+ 1.354 (P) 1.276 (P)

20 Arrival:= 12.037 -0.040

21

22 Setup:- 0.166

23 Cppr Adjust:+ 0.000

24 Required Time:= 11.871

25 Launch Clock:= -0.040

26 Data Path:+ 11.858

27 Slack:= 0.054

28 Timing Path:

29

30 #-----

31 # Timing Point

32 # Fanout Trans Delay Arrival

33 # (ns) (ns) (ns)

Flags Arc Edge Cell

Fig 12: The setup report after place and route

qr_decomp_postRoute_all.tarpt

qr_decomp_postRoute_all_hold.tarpt

1 #####

2 # Generated by: Cadence Innovus 21.13-s100_1

3 # OS: Linux x86_64(Host ID linux-lab-043.ece.uw.edu)

4 # Generated on: Tue Jun 4 21:40:31 2024

5 # Design: qr_decomp

6 # Command: opt_design -post_route -setup -hold -expanded_views

7 #####

8 Path 1: MET (0.101 ns) Hold Check with Pin d1/ib/in_buffer/r_unit/array_reg_reg[14][4]/CLK->D

9 View: ff_n40C_lv95.hold_view

10 Group: clk

11 Startpoint: (F) A[4]

12 Clock: (R) clk

13 Endpoint: (F) d1/ib/in_buffer/r_unit/array_reg_reg[14][4]/D

14 Clock: (R) clk

15

16

17 Capture Launch

18 Clock Edge:+ 0.000 0.000

19 Src Latency:+ -0.522 0.000

20 Net Latency:+ 0.569 (P) 0.000 (I)

21 Arrival:= 0.047 0.000

22

23 Hold:+ -0.040

24 Cppr Adjust:- 0.000

25 Required Time:= 0.007

26 Launch Clock:= 0.000

27 Data Path:+ 0.107

28 Slack:= 0.101

29 Timing Path:

30 #-----

31 # Timing Point

Trans Delay Arrival

Flags Arc Edge Cell Fanout

Fig 13: The hold report after place and route

qr_decomp_area.rpt						
~jee478/ee477-designs-module1/gcd/build/par-rundir						
Inst Name	Total Area	Buffer	Module Name			
Inst Count	Flop	Latch	Inverter	Clock Gate		
Combinational						
Macro	Physical					

3 qr_decomp						
8803	92671.379	481.712	5665.434	47890.931		
38633.302	0.000	2255.914	0.000	0.000		
4 c1						
controller_mapped		72	475.456			
0.000	48.797	325.312	101.347	0.000		
18.768	0.000	0.000				
5 c1/CLKGATE_RC_CG_HIER_INST0						
CLKGATE_RC_CG_MOD_mapped		1	18.768			
0.000	0.000	0.000				
18.768	0.000	0.000				
6 d1						
datapath_DATA_WIDTH16_ADDR_WIDTH4_mapped		8725	92099.581			
456.688	5545.318	47565.619	38531.955	0.000		
2237.146	0.000	0.000				
7 d1/buffer_dot						
buffer_3_DATA_WIDTH16_ADDR_WIDTH4_112_mapped		786	9589.197			
53.802	601.827	2437.338	6496.230	0.000		
337.824	0.000	0.000				
8 d1/buffer_dot/in_buffer						
fifo_DATA_WIDTH16_ADDR_WIDTH4_116_mapped		756	9307.677			
53.802	601.827	2315.971	6336.077	0.000		
337.824	0.000	0.000				
9 d1/buffer_dot/in_buffer/c_unit						
fifo_ctrl_ADDR_WIDTH4_VECTOR_LEN1_120_mapped		60	594.320			
0.000	3.754	347.834	242.733	0.000		
37.536	0.000	0.000				
10 d1/buffer_dot/in_buffer/c_unit/CLKGATE_RC_CG_HIER_INST1						
CLKGATE_RC_CG_MOD 1 mapped		1	18.768			

Fig 14: The area report after place and route

Open

qr_decomp_power.rpt

Save

≡

—

□

×

~\ee478\ee477-designs-module1\gcd\build\par-rundir

43

44 Total Power

45 -----

46 Total Internal Power: 5.65405060 52.6126%

47 Total Switching Power: 5.03385427 46.8415%

48 Total Leakage Power: 0.05866807 0.5459%

49 Total Power: 10.74657294

50 -----

51

52

53 Group Internal Power Switching Power Leakage Power Total Power Percentage (%)

54 -----

55

56 Sequential 3.389 0.3962 0.02635 3.812 35.47

57 Macro 0 0 0 0 0

58 IO 0 0 0 0 0

59 Combinational 1.571 2.572 0.02813 4.171 38.81

60 Clock (Combinational) 0.3524 1.851 0.002296 2.206 20.53

61 Clock (Sequential) 0.3411 0.2149 0.001889 0.5579 5.192

62 -----

63 Total 5.654 5.034 0.05867 10.75 100

64 -----

65

66

67 Clock Internal Power Switching Power Leakage Power Total Power Percentage (%)

68 -----

69

70 clk 0.6936 2.066 0.004185 2.764 25.72

71 -----

72 Total (excluding duplicates) 0.6936 2.066 0.004185 2.764 25.72

73 -----

74 Clock: clk

75 Clock Period: 0.012000 usec

76 Clock Toggle Rate: 166.6667 Mhz

Plain Text

Tab Width: 8

Ln 1, Col 1

INS



Fig 15: The power report after place and route

Observations after PAR:

Maximum frequency : **84.04 MHz**.

Area: **92671.379 um²**

Power: **10.74 mW**

Open ▾  qr_decomp_timing_setup.rpt
~\ee478\ee477-designs-module1\gcd\build\timing-par-rundir Save  - □ ×

tt_025C_lv80.extra_view

12 #
13 # CPPR : on
14 # Latch Slack Mode : max_borrow
15 # Clock Propagation : sdcControl
16 # Delay Calculation:
17 # Engine : default (Sign-off)
18 #
19 #####
20 Path 1: MET (0.231 ns) Setup Check with Pin d1/mul/d2_mul_temp_reg[23]/CLK->D
21 Retime Analysis: Data Path-Slew SI GP
22 View: ss_100C_lv60.setup_view
23 Group: clk
24 Startpoint: (R) d1/lq/q2_l_r_reg[9]/CLK
25 Clock: (R) clk
26 Endpoint: (R) d1/mul/d2_mul_temp_reg[23]/D
27 Clock: (R) clk
28
29 Capture Launch
30 Clock Edge:+ 12.000 0.000
31 Src Latency:+ 0.000 0.000
32 Net Latency:+ 0.000 (I) 0.000 (I)
33 Arrival:= 12.000 0.000
34
35 Setup:- 0.178
36 Cppr Adjust:+ 0.000
37 Required Time:= 11.822
38 Launch Clock:= 0.000
39 Data Path:+ 11.591
40 Slack:= 0.231
41 Slack(original):= 0.009
42
43 #-----
44 # Timing Point Flags Arc Edge Cell Fanout

Fig 16: The setup report after STA

```

13 #
14 #           Cppr : on
15 #           Latch Slack Mode : max_borrow
16 #           Clock Propagation : sdcControl
17 # Delay Calculation:
18 #           Engine : default (Sign-off)
19 #####
20 Path 1: MET (0.139 ns) Hold Check with Pin d1/ib/in_buffer/r_unit/array_reg_reg[15][3]/CLK->D
21   Retime Analysis: Data Path-Slew SI GP
22   View: ff_n40C_lv95.hold_view
23   Group: clk
24   Startpoint: (F) A[3]
25   Clock: (R) clk
26   Endpoint: (F) d1/ib/in_buffer/r_unit/array_reg_reg[15][3]/D
27   Clock: (R) clk
28
29           Capture      Launch
30   Clock Edge: + 0.000    0.000
31   Src Latency: + 0.000    0.000
32   Net Latency: + 0.000 (I) 0.000 (I)
33   Arrival: = 0.000      0.000
34
35           Hold: + -0.045
36   Cppr Adjust: - 0.000
37   Required Time: = -0.045
38   Launch Clock: = 0.000
39   Data Path: + 0.093
40   Slack: = 0.139
41   Slack(original): = 0.131
42
43 #-----
44 # Timing Point

```

	Capture	Launch
Clock Edge: +	0.000	0.000
Src Latency: +	0.000	0.000
Net Latency: +	0.000 (I)	0.000 (I)
Arrival: =	0.000	0.000
Hold: +	-0.045	
Cppr Adjust: -	0.000	
Required Time: =	-0.045	
Launch Clock: =	0.000	
Data Path: +	0.093	
Slack: =	0.139	
Slack(original): =	0.131	

Fig 17: The hold report after STA

c. Hardware simulation results: We have conducted comprehensive hardware simulations to verify the functionality and performance of our QR decomposition module. The simulations include:

1. **RTL Simulation:** Using tools such as Vivado, Quartus/ModelSim, and VCS, we performed RTL simulations of all submodules. These simulations verified the correctness of individual components, including fundamental vector operations, control units, and data path units. The simulation verified the interaction and compatibility between different components and ensured the overall functionality of the design.
 - o **Status:** RTL simulations have passed successfully for inputs where A and Q are positive, demonstrating the correct behavior of the individual submodules. We suspect the problem is because the normliazation module is not able to compute the 2's complete values correctly.

d. Verification results: DRC has passed as shown in the screenshot below:

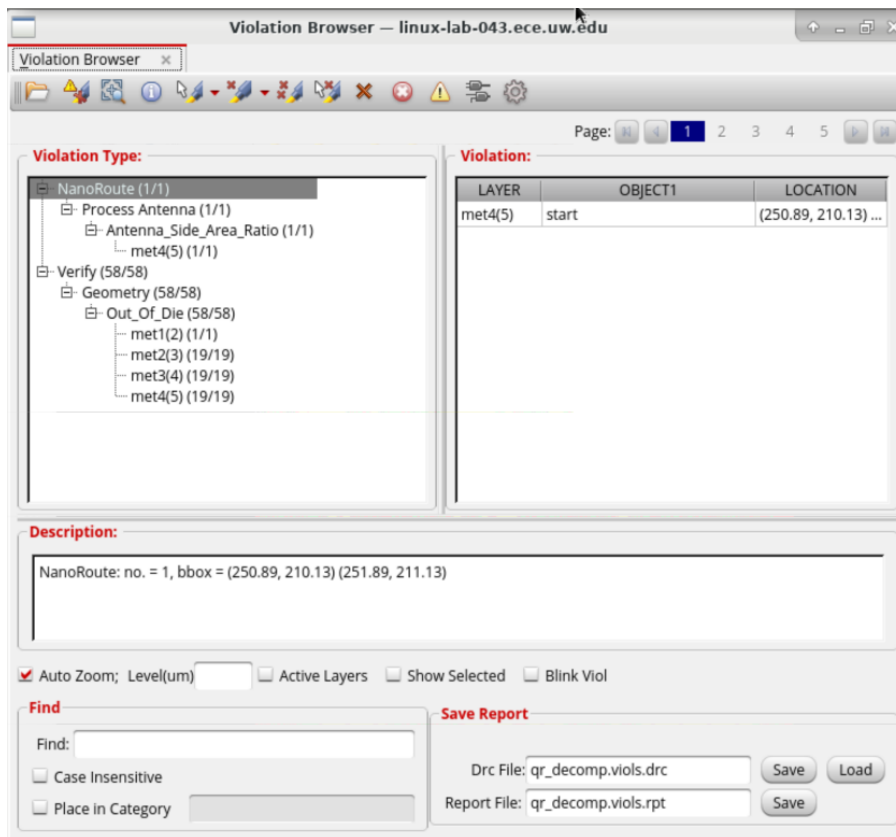


Fig 18: DRC has passed

Realistic Factors in Design

In developing our QR decomposition module, we consider various realistic factors to ensure its effectiveness, safety, and relevance in real-world applications:

Public Health, Safety, and Welfare

We adhere to industry standards and regulations to ensure the safety and reliability of our hardware design. This includes compliance with regulatory bodies such as the International Electrotechnical Commission (IEC) and the Institute of Electrical and Electronics Engineers (IEEE). Hazard analysis is conducted to identify and mitigate potential risks associated with the hardware implementation.

Global Factors

We assess the competitive landscape to understand existing solutions and differentiate our QR decomposition module by focusing on superior performance, scalability, and efficiency. Evaluating competing products helps us identify market gaps and opportunities for innovation.

Manufacturability

We consider manufacturability factors such as fabrication process compatibility, yield optimization, and supply chain logistics to ensure the scalability and cost-effectiveness of mass production.

Cultural Factors

We analyze cultural preferences, demographics, and market segmentation to tailor our QR decomposition module to specific industry verticals and customer needs. Understanding cultural factors enables us to design a solution that resonates with diverse user groups.

Social Factors

We prioritize user experience and ergonomics in the design process to create intuitive interfaces and user-friendly interactions. Considering human factors ensures that our QR decomposition module is accessible and usable by a wide range of users, regardless of their technical expertise.

Environmental Factors

We strive to minimize environmental impact by optimizing resource utilization, reducing energy consumption, and promoting sustainability throughout the product lifecycle. This includes selecting eco-friendly materials, implementing energy-efficient designs, and facilitating responsible disposal practices.

Economic Factors

We assess the economic feasibility of our QR decomposition module by evaluating production costs, pricing strategies, and return on investment. Cost-effective design choices are made to maximize value for customers while ensuring profitability and long-term sustainability.

Link to our repository: <https://github.com/ShreyaMahetaliya/Hardware-Accelerator-for-QR-Decomposition>

Quality Metrics

Our goal was to optimize performance and area. Our QR decomposition module is compared with an FPGA implementation using High-Level Synthesis (HLS) as described in the referenced paper. Through synthesis, our ASIC design achieved a post-synthesis frequency of 111 MHz, surpassing the post-synthesis performance reported in the HLS FPGA implementation paper at 100 MHz. Additionally, the post-place-and-route (PAR) frequency for the ASIC design was recorded at 84.04 MHz.

Comparing these metrics with the HLS FPGA implementation, our ASIC design demonstrates slightly better performance in terms of clock frequency post synthesis. This suggests that ASIC

implementations, specifically tailored to the algorithm and hardware specifications, can potentially offer higher operating frequencies and thus faster computation times compared to FPGA implementations, which are constrained by the capabilities of the FPGA hardware.

Furthermore, while our ASIC design showcases impressive performance metrics, it's essential to consider factors such as development time, cost, and flexibility. ASIC development typically involves higher upfront costs and longer design cycles compared to FPGA-based approaches. However, ASICs can offer significant benefits in terms of power efficiency, area optimization, and overall performance once fabricated.

Our project goals were not met since we were not able to significantly improve performance during the course of our project, but the above results indicate that our progress was in the right direction, and further optimization in terms of parallelizing computation would improve the performance of our design significantly as compared to the HLS implementation.

Agile Milestones

1. Vector operations:

Planned: Implement basic vector operations such as dot product, normalization, vector multiplication and division with a scalar and vector subtraction.

Completed: Designed and tested the lower modules.

2. Basic QR Factorization Implementation:

Planned: Develop and validate a basic QR factorization implementation, forming the foundation of our QR decomposition module.

Completed: Designed Architecture and Controller of entire QR decomposition module.

3. Block QR Implementation:

Planned: Implement block QR decomposition to optimize performance and leverage parallel computing resources effectively.

Completed: Designed the sequential QR decomposition module and debugging to get it working.

4. Optimized Version:

Planned: Fine-tune the QR decomposition module for enhanced performance, area, and power efficiency.

Completed: Finished simulation and implementation of QR decomposition module.

Limitations of the project

Despite our efforts to develop a robust and efficient QR decomposition module, there are certain limitations inherent in the current project:

1. **Division Module Accuracy:** Our current division module employs approximation techniques, which may affect the precision of the results in certain scenarios.
2. **Input Constraints:** At present, the QR decomposition module is designed to handle only positive and integer inputs, limiting its applicability to a wider range of matrices.
3. **Time Constraints:** The project timeline may limit the depth and scope of implementation, potentially restricting the exploration of advanced optimization techniques or comprehensive testing scenarios.
4. **Algorithmic Complexity:** The complexity of the QR decomposition algorithm, especially when implemented in hardware, may introduce challenges in terms of resource utilization, latency, and throughput optimization.
5. **Verification and Validation:** Comprehensive verification and validation of the hardware design may require significant time and effort, leading to potential gaps in testing coverage or inadequate identification of corner cases.

Future Optimization Opportunities

To address these limitations and further enhance the QR decomposition module, several areas can be optimized in the future:

1. **Improved Division Precision:** Develop more accurate division modules to enhance the precision of the QR decomposition results.
2. **Expanded Input Handling:** Extend the module's capability to handle a wider range of input types, including negative and non-integer values, to increase its applicability and flexibility.
3. **Algorithmic Refinement:** Investigate alternative algorithms or algorithmic optimizations to reduce computational complexity, improve performance, and enhance scalability for a broader range of matrix sizes and types. Specifically, enhancing performance and scalability by leveraging parallelism in the QR decomposition process. Since the Gram Schmidt algorithm inherently lacks parallelism, employing a block method to parallelize it effectively can significantly improve processing efficiency.
4. **Hardware Architecture Optimization:** Explore architectural optimizations, such as parallel processing techniques, pipelining, and memory hierarchy enhancements, to maximize resource utilization, minimize latency, and increase throughput.
5. **Verification Methodologies:** Implement advanced verification methodologies, including formal verification, assertion-based verification, and constrained-random testing, to ensure comprehensive coverage of design functionality and robustness against corner cases.
6. **Power and Energy Efficiency:** Investigate power-aware design techniques, dynamic voltage and frequency scaling (DVFS), and low-power circuit design methodologies to optimize energy efficiency and extend battery life for portable or energy-constrained applications.

7. **Fault Tolerance and Reliability:** Enhance fault tolerance mechanisms, error detection and correction codes, and reliability features to mitigate the impact of hardware faults and improve the overall robustness and resilience of the QR decomposition module.

By incorporating these strategies into future iterations of the project, we aim to overcome current limitations, improve performance, and deliver a more efficient and scalable QR decomposition solution tailored for scientific and engineering applications.