AIM OF THE PROJECT

Implementation of linear equation solver on FPGA using VHDL . An N*N matrix can be implemented successfully without much delay .

MOTIVATION

Big linear equations with multiple variable are too difficult for a human being to solve manually. These equations are way too complex to be solved in generic calculators. So we planned to design a FPGA based linear equation solver which can solve equations containing upto 8 variables.

Almost all computer graphics are plagued (for lack of a better word) with applications of linear algebra. For video games, you have to cope with 3D transformation matrices as well as some losses but fast image compression/display methods. In the latter case, the bitmap of an image is taken and take the Singular Value Decomposition of the matrix, and preserve only the values of the Singular Value Decomposition. Now the matrix generated for this purpose generally reaches an order of >5.

Linear algebra helps in solving Markov chains, a very useful probabilistic tool that is used in all sorts of areas, from biological population dynamics models to economics predictions to traffic flow models to incompressible fluid flow dynamics.

Modern web search engines use many techniques to estimate the relevance of pages. An important one is link analysis based on the concept of eigenvector centrality, which comes from linear algebra.

Apart from the above mentioned applications, there are a lot of areas where complex linear programming is used. Thus a FPGA based linear equation solver solves them all.

INTRODUCTION

The Linear Equation Solver designed used LU Decomposition method, for the solving a set 8 linear equations having 8 unknown variables.

LU DECOMPOSITION METHOD:-

Suppose we have the system of equations AX = B.

The motivation for an LU decomposition is based on the observation that systems of equations involving triangular coefficient matrices are easier to deal with. Indeed, the whole point of Gaussian elimination is to replace the coefficient matrix with one that is triangular. The LU decomposition is another approach designed to exploit triangular systems.

We suppose that we can write A = LU where L is a lower triangular matrix and U is an upper triangular matrix. Our aim is to find L and U and once we have done so we have found an LU decomposition of A.

Explicitly,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots\ldots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots\ldots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 & \ldots\ldots & 0 \\ l_{21} & l_{22} & 0 & \ldots\ldots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & l_{n3} & \ldots\ldots & l_{nn} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & \ldots\ldots & u_{1n} \\ 0 & 1 & u_{23} & \ldots\ldots & u_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \ldots\ldots & 1 \end{pmatrix}$$

Therefore, by LU-decomposition, the system of linear equations Ax=B can be solved in three steps:

I. Construct the lower triangular matrix L and upper triangular matrix U.

II. Using forward substitution, solve Ly=B.

III. Solve U x =y, backward substitution.

Algorithm for LU Decomposition:-

Pseudo-code for a simple LU factorization of N x N A matrix:-

```
 for k = 1 to N-1 do {for each diagonal element}
     for i = k + 1 to N do {for each element below it}
          $a_{i,k} \leftarrow a_{i,k}/a_{k,k}$ {normalize}
     end for
     for j = k + 1 to N do {for each column right of current diagonal element}
          for l = k + 1 to N do {for each element below it}
               $a_{l,j} \leftarrow a_{l,j} - a_{l,k} \times a_{k,j}$
          end for
     end for
end for
```

Algorithm for Forward Substitution:-

Forward substitution to solve Ly=B

```
x(1)=b(1)
   for i = 2 : n
        s=0
        for j = 1 : i-1
             s = s + a(i,j)× x(j)
        end for
        x(i) = b(i) - s
   end for
```

Algorithm for Backward Substitution:-

back substitution to solve Uy=B:-

```
x(n) = x(n) / a(n, n)
     for i = n-1 : -1 : 1
          s=0
          for j = i+1 to n
               s = s + a( i, j) * x(j)
          end for
          x(i) = (x(i) - s)/a(i,i)
     end for
```

Reasons for choosing LU Decomposition:-

LU Decomposition is computationally more efficient than Gaussian elimination when we are solving several sets of equations with the same coefficient matrix but different right hand sides.

For large *n*, for LU Decomposition, the computational time is proportional to $\frac{4n^3}{3}$, while for Gaussian Elimination, the computational time is proportional to $\frac{n^4}{3}$. So for large *n*, the ratio of the computational time for Gaussian elimination to computational for LU Decomposition is $\frac{n^4}{3}/\frac{4n^3}{3} = \frac{n}{4}$.

As an example, to find the inverse of a 2000×2000 coefficient matrix by Gaussian Elimination would take n/4=2000/4=500 times the time it would take to find the inverse by LU Decomposition.

FLOWCHARTS:-

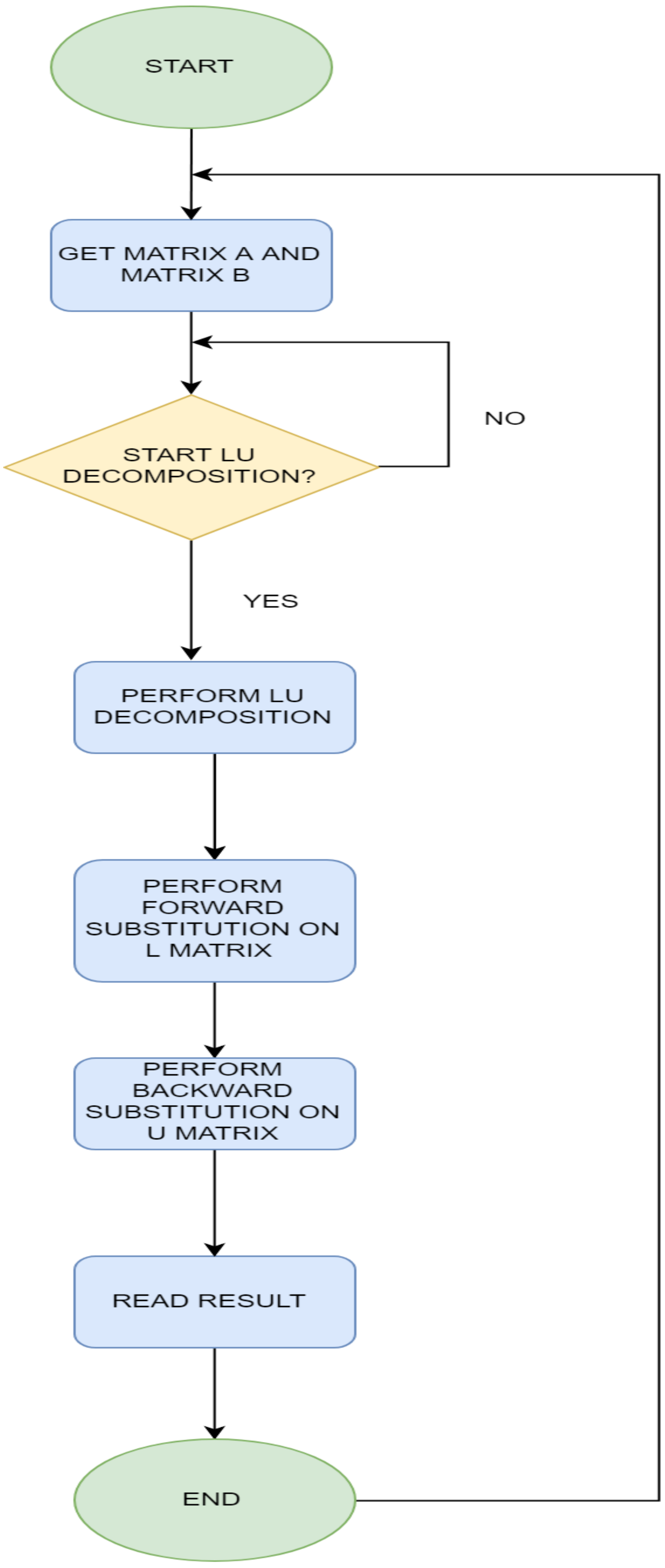

START
GET MATRIX A AND MATRIX B
START LU DECOMPOSITION?
NO
YES
PERFORM LU DECOMPOSITION
PERFORM FORWARD SUBSTITUTION ON L MATRIX
PERFORM BACKWARD SUBSTITUTION ON U MATRIX
READ RESULT
END

Figure 1 Top level flowchart of linear equation solver

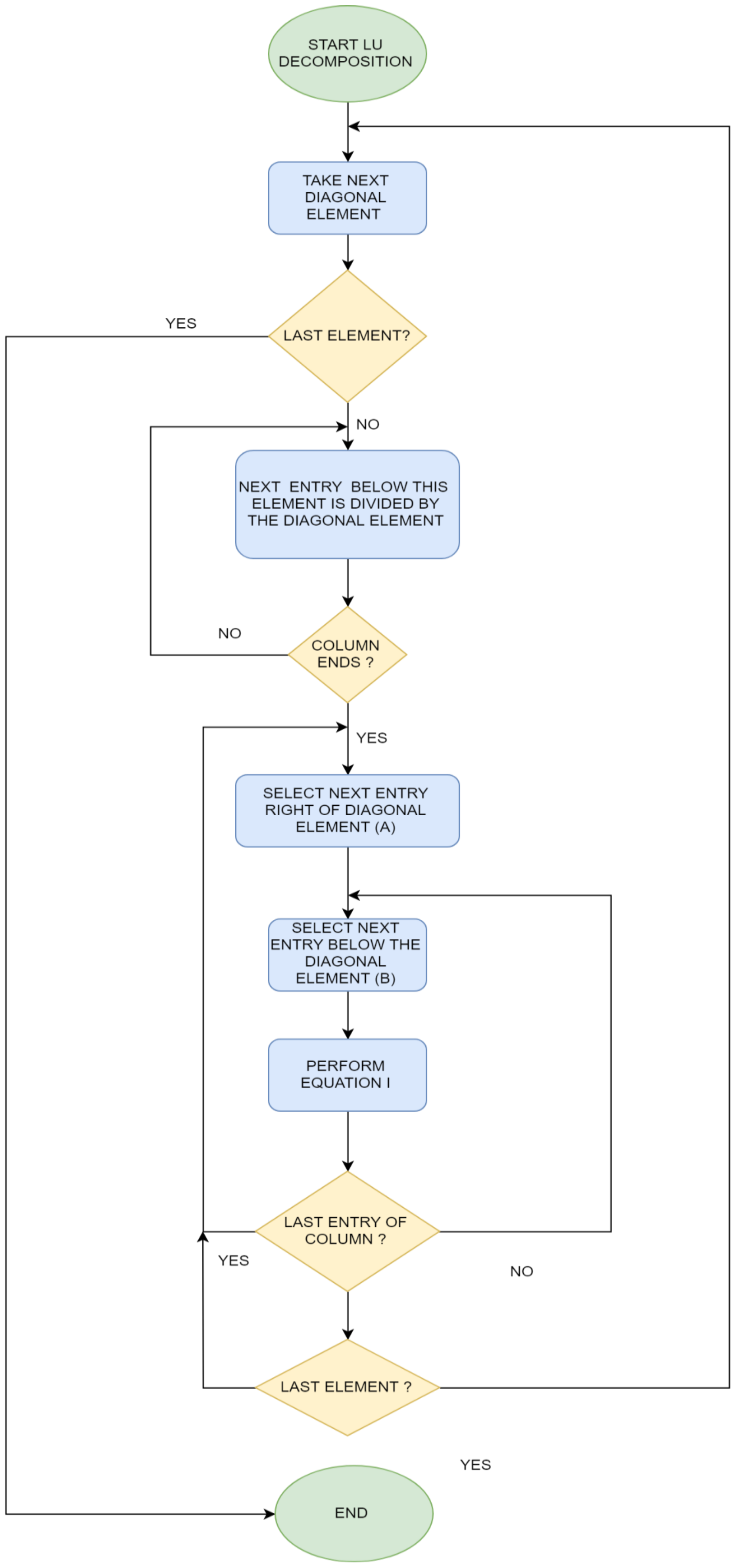

START LU DECOMPOSITION
TAKE NEXT DIAGONAL ELEMENT
LAST ELEMENT?
YES
NO
NEXT ENTRY BELOW THIS ELEMENT IS DIVIDED BY THE DIAGONAL ELEMENT
COLUMN ENDS ?
NO
YES
SELECT NEXT ENTRY RIGHT OF DIAGONAL ELEMENT (A)
SELECT NEXT ENTRY BELOW THE DIAGONAL ELEMENT (B)
PERFORM EQUATION I
LAST ENTRY OF COLUMN ?
YES
NO
LAST ELEMENT ?
YES
END

Figure 2 LU Decomposition

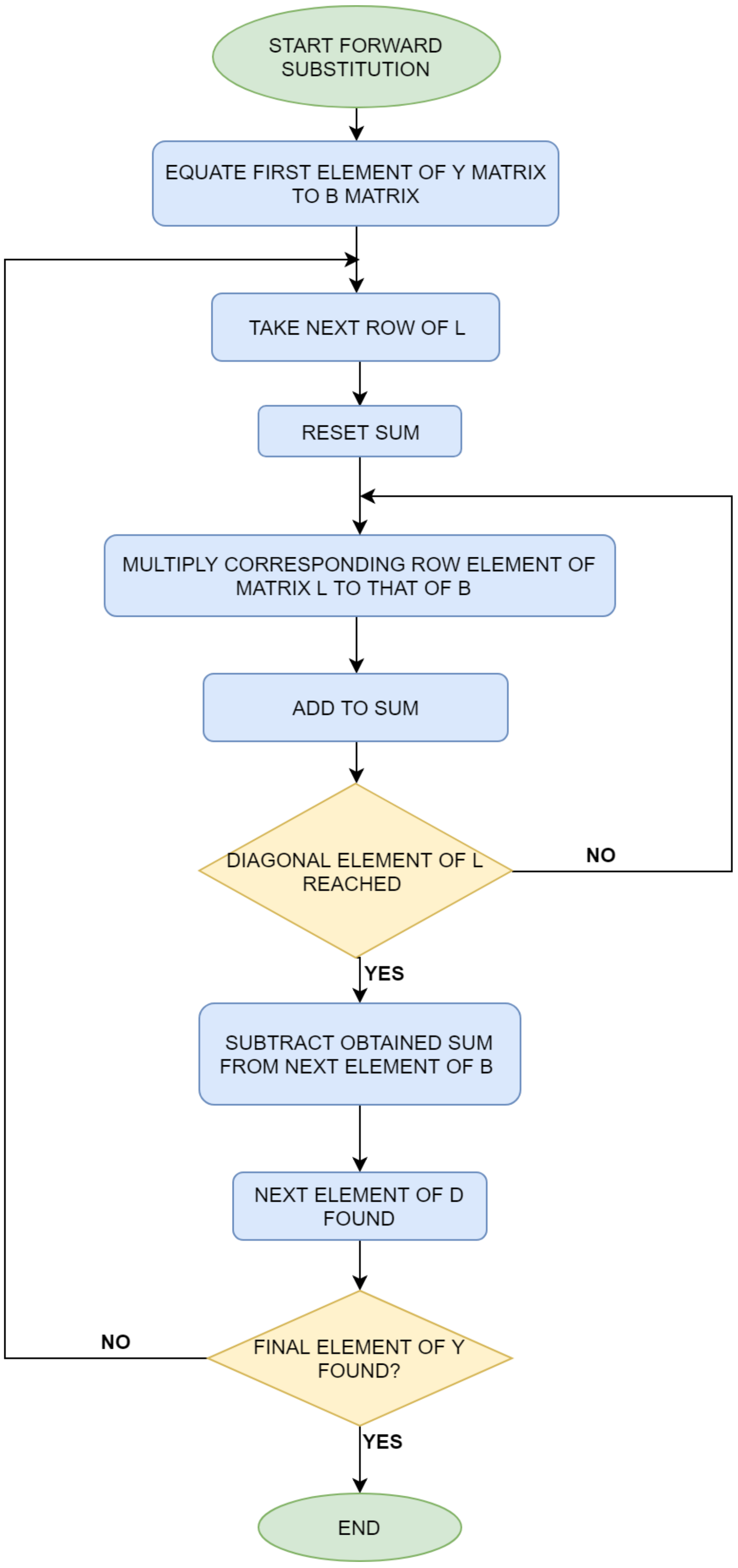

START FORWARD SUBSTITUTION
EQUATE FIRST ELEMENT OF Y MATRIX TO B MATRIX
TAKE NEXT ROW OF L
RESET SUM
MULTIPLY CORRESPONDING ROW ELEMENT OF MATRIX L TO THAT OF B
ADD TO SUM
DIAGONAL ELEMENT OF L REACHED
NO
YES
SUBTRACT OBTAINED SUM FROM NEXT ELEMENT OF B
NEXT ELEMENT OF D FOUND
FINAL ELEMENT OF Y FOUND?
NO
YES
END

Figure 3 Solving Ly=B

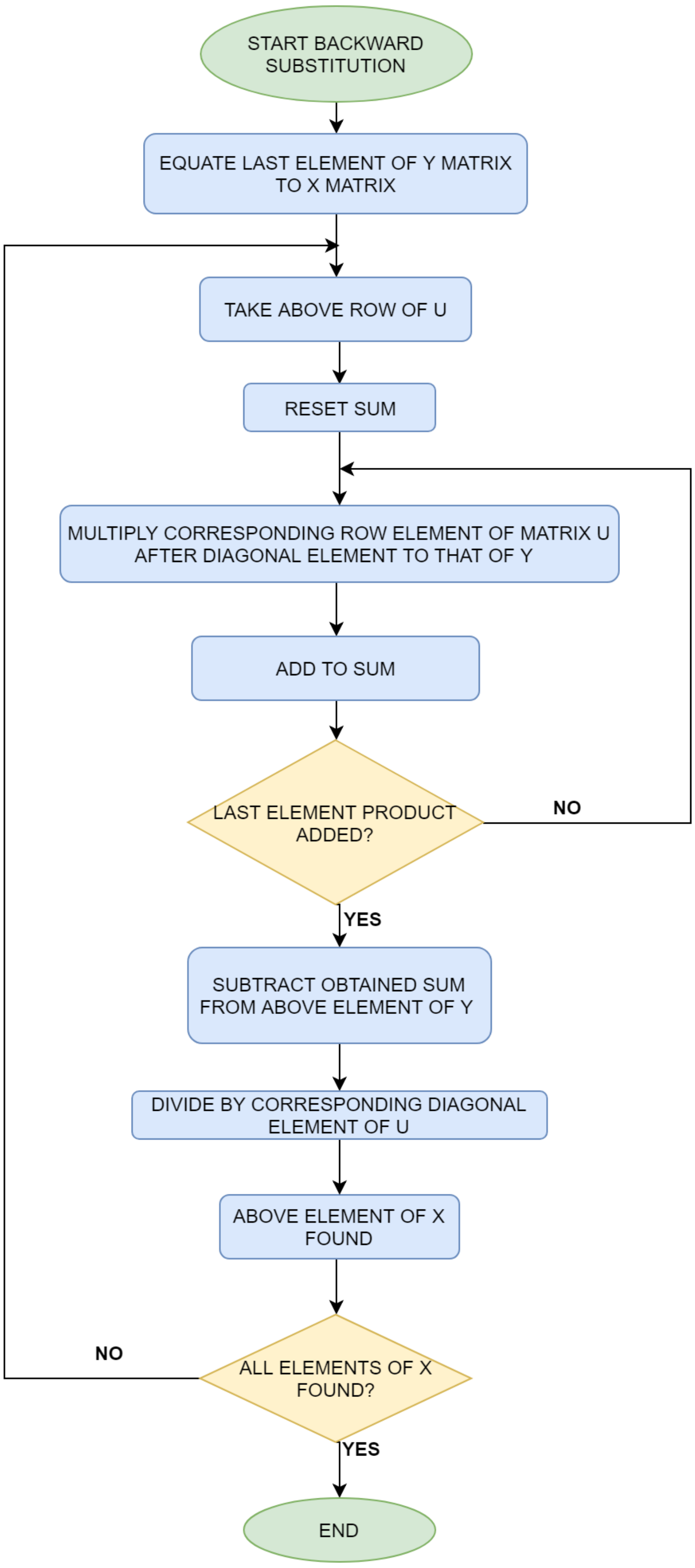

START BACKWARD SUBSTITUTION
EQUATE LAST ELEMENT OF Y MATRIX TO X MATRIX
TAKE ABOVE ROW OF U
RESET SUM
MULTIPLY CORRESPONDING ROW ELEMENT OF MATRIX U AFTER DIAGONAL ELEMENT TO THAT OF Y
ADD TO SUM
LAST ELEMENT PRODUCT ADDED?
NO
YES
SUBTRACT OBTAINED SUM FROM ABOVE ELEMENT OF Y
DIVIDE BY CORRESPONDING DIAGONAL ELEMENT OF U
ABOVE ELEMENT OF X FOUND
NO
ALL ELEMENTS OF X FOUND?
YES
END

Figure 4 Solving Ux=Y

GUIDELINES TO BUILD PROJECT IN NIOS II HARDWARE

1 Open Project File

2 Add all files --> <Project Directory>\LU_Linear_Equation_Solver\synthesis\submodules

3 Add file "LU_Linear_Equation_Solver.vhd"<Project Directory > \LU_Linear_Equation_Solver\synthesis

4 Set "LU_Linear_Equation_Solver.vhd" as top level entity

5 Compile Design

6 Program the Deo – Nano FPGA

7 Tools -> NIOS II Software Build Tools for Eclipse

8 Make a new workspace directory.

9 File -> New -> Nios II Application BSP from template.

10 Select the sopcinfo file inside GS_nios1 folder (<Project Directory>\LU_Linear_Equation_Solver.sopcinfo).

11 Open hello_world_small.c from <Project Directory>\software

12 Copy the contents from above file to hello_world_small.c in the project .

13 Build the project.

14 Run as Nios II hardware .

.RTL MODELSIM SIMULATION

1. Open Project

2. Add file “LU_Componants.vhd” from <Project Directory> \LU_Components

3. Add files “Float_32_Add.vhd” , “Float_32_Div.vhd” and “Float_32_Mult.vhd” from <Project Directory>\IP_Componants.

4. Go to Hierarchy , right click on LU_Controller , Select settings , goto General , enter LU_Controller as top level entity

5. Run Analysis & Synthesis. (Don’t Compile)

6. Run RTL Simulation

7. Run Vsim RTL simulation “LU_Controller_8x8_modelsim.txt.txt” <Project Directory> \Testbenches.

AN EXAMPLE

$$\begin{bmatrix} 6 & 4 & 1 & -3 & -2 & -3 & 5 & 8 \\ -1 & 10 & 2 & 4 & 3 & 1 & 9 & 2 \\ 4 & 2 & 9 & 5 & -1 & 2 & -5 & -3 \\ 5 & -4 & -3 & 9 & 4 & 7 & 8 & 6 \\ 3 & 2 & 8 & 7 & 10 & 6 & 5 & 1 \\ -5 & 5 & 4 & 3 & 1 & 7 & 5 & 8 \\ 2 & 1 & 5 & 6 & -8 & 5 & 11 & 6 \\ 2 & 4 & 5 & 7 & -5 & 6 & 4 & 2 \end{bmatrix} \times \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{bmatrix}$$

Y MATRIX FOUND FROM LY=B

$$\begin{bmatrix} 1.000 \\ 2.166 \\ 2.469 \\ 5.34 \\ 1.6 \\ 5.96 \\ 1.97 \\ 1.437 \end{bmatrix}$$

RESULTANT X MATRIX FOUND

$$\begin{bmatrix} 0.49 \\ 0.3256 \\ 0.0602 \\ -0.657 \\ -0.23 \\ 1.454 \\ 0.1723 \\ -0.282 \end{bmatrix}$$

TIMING ANALYSIS

FOR SLOW 1200MILIVOLTS 85C MODEL

SUMMARY

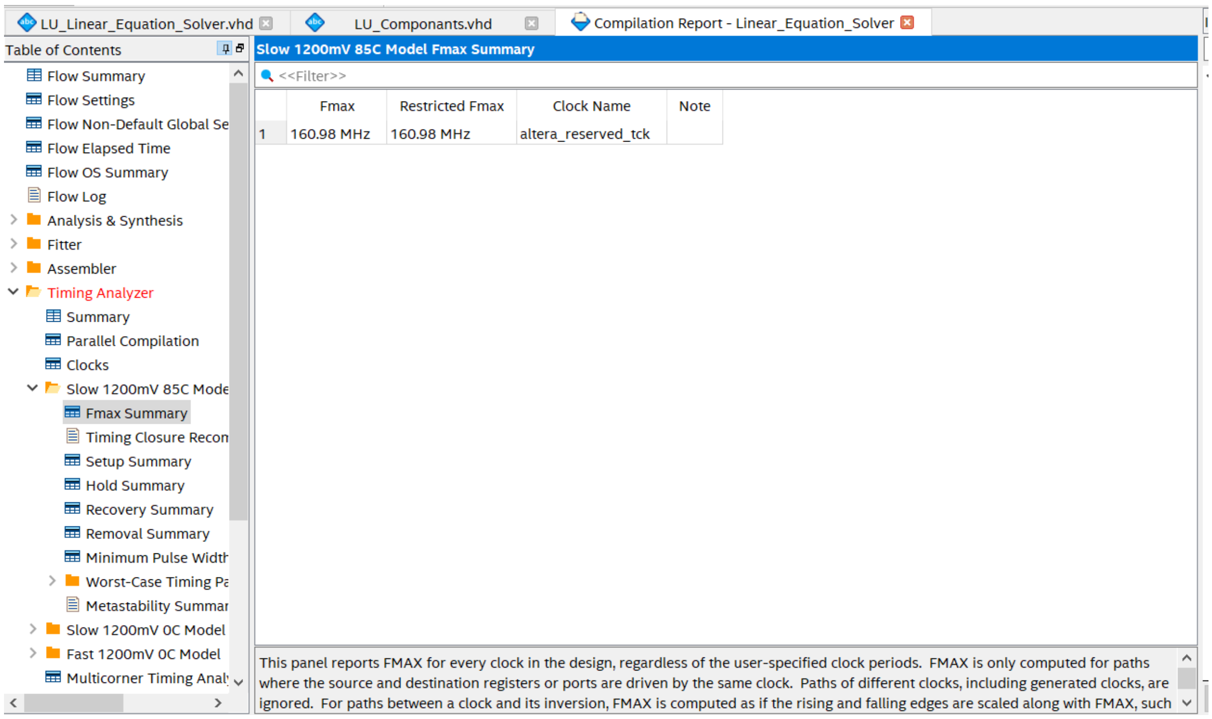

LU_Linear_Equation_Solver.vhd | LU_Componants.vhd | Compilation Report - Linear_Equation_Solver

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Se
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - Clocks
  - Slow 1200mV 85C Mode
    - Fmax Summary
    - Timing Closure Recon
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Widtl
    - Worst-Case Timing Pa
    - Metastability Summar
  - Slow 1200mV 0C Model
  - Fast 1200mV 0C Model
  - Multicorner Timing Anal

Slow 1200mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	160.98 MHz	160.98 MHz	altera_reserved_tck	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such

SETUP SUMMARY

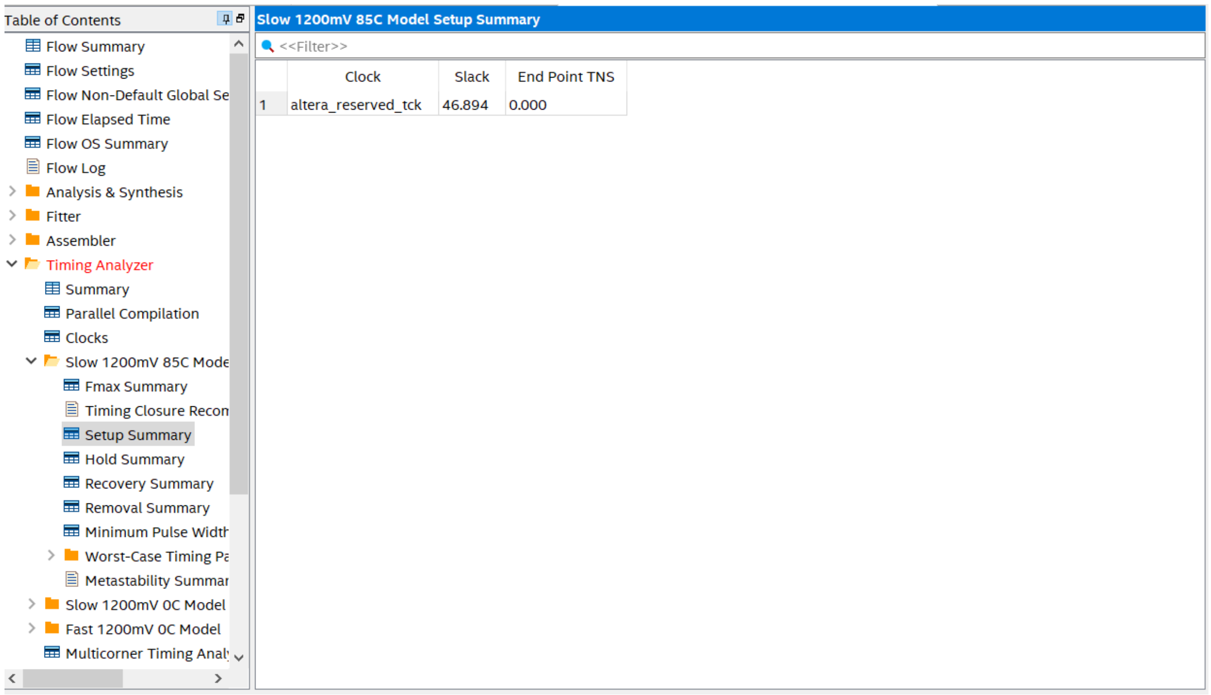


Table of Contents

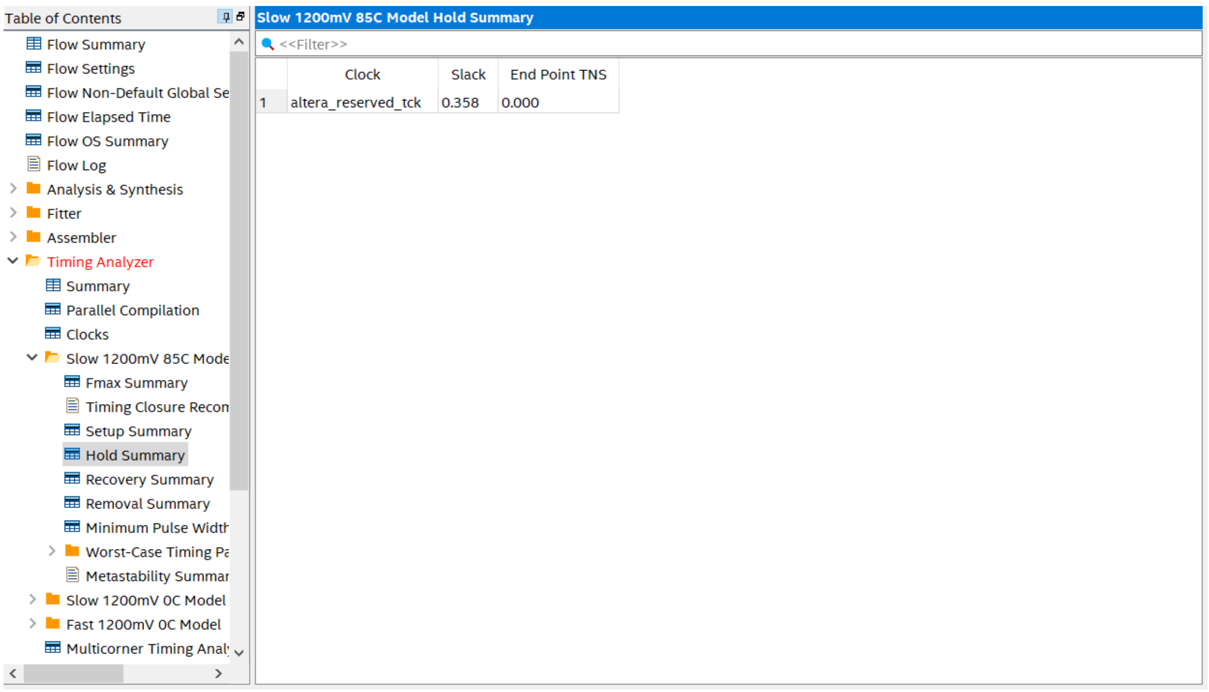
- Flow Summary
- Flow Settings
- Flow Non-Default Global Se
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - Clocks
  - Slow 1200mV 85C Mode
    - Fmax Summary
    - Timing Closure Recon
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Widtl
    - Worst-Case Timing Pa
    - Metastability Summar
  - Slow 1200mV 0C Model
  - Fast 1200mV 0C Model
  - Multicorner Timing Anal

Slow 1200mV 85C Model Setup Summary

<<Filter>>

	Clock	Slack	End Point TNS
1	altera_reserved_tck	46.894	0.000

HOLD SUMMARY

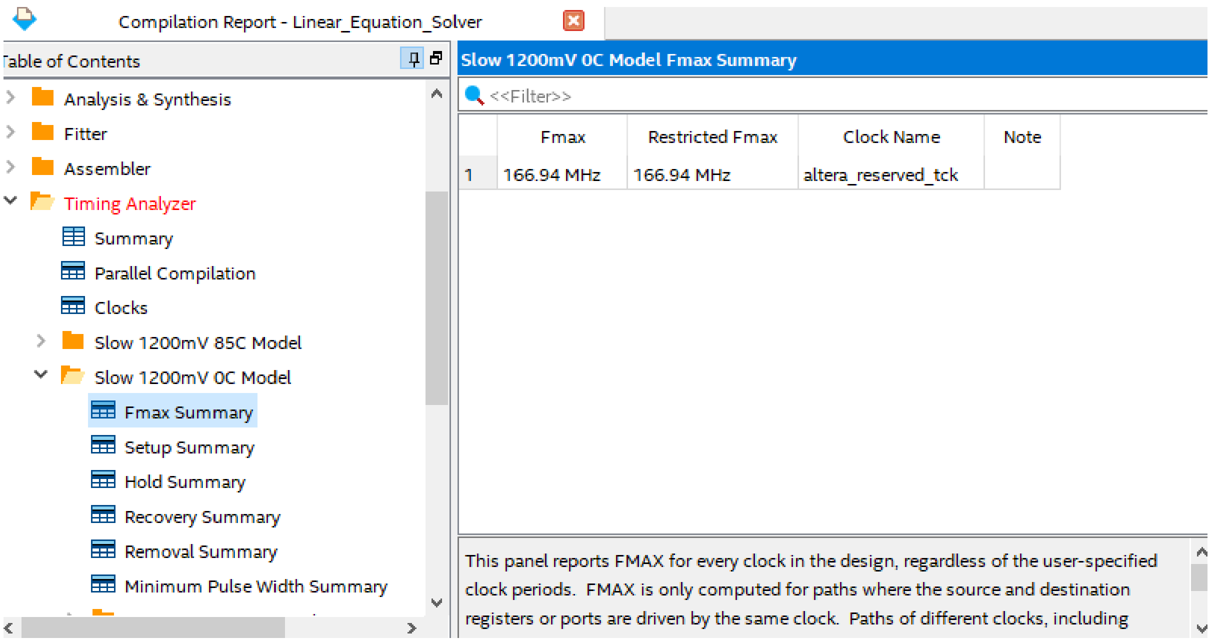


Slow 1200mV 85C Model Hold Summary

	Clock	Slack	End Point TNS
1	altera_reserved_tck	0.358	0.000

FOR SLOW 1200 MILIVOLTS OC MODEL

SUMMARY

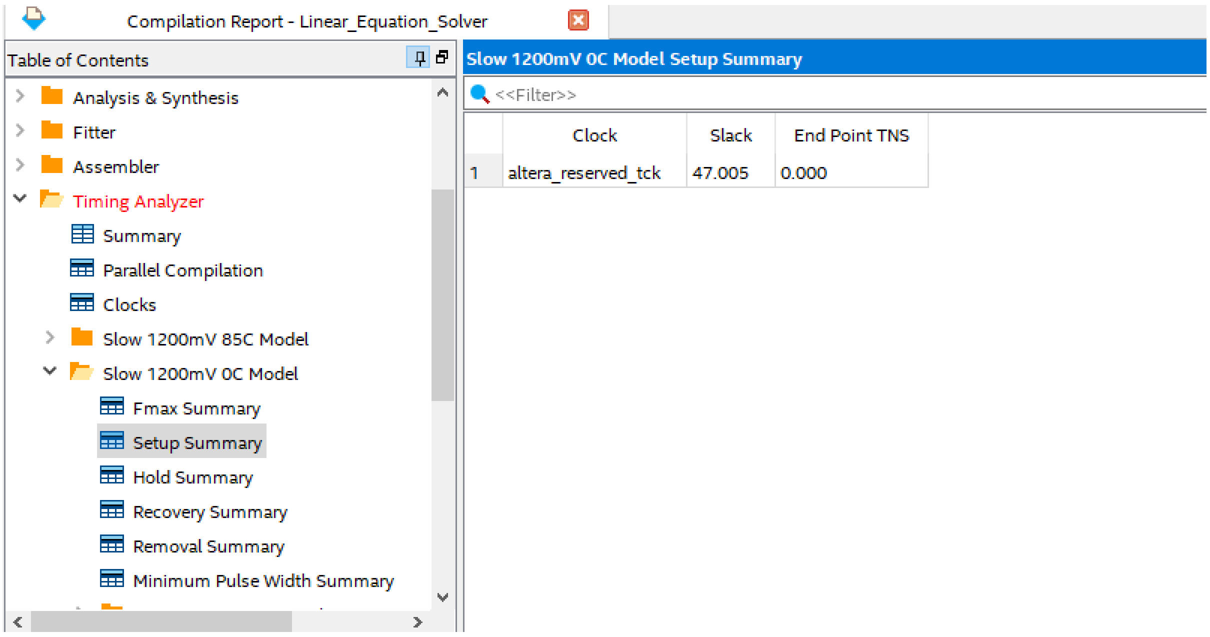


Slow 1200mV 0C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	166.94 MHz	166.94 MHz	altera_reserved_tck	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including

SETUP SUMMARY



Compilation Report - Linear_Equation_Solver

Table of Contents

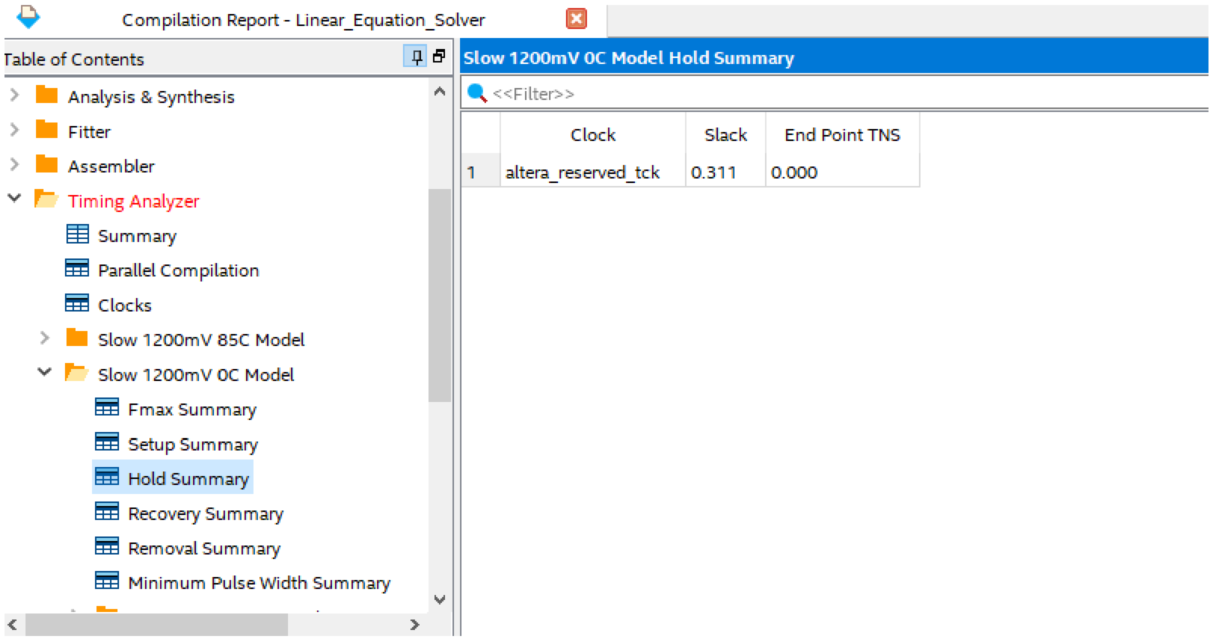
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - Clocks
  - Slow 1200mV 85C Model
  - Slow 1200mV 0C Model
    - Fmax Summary
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Summary

Slow 1200mV 0C Model Setup Summary

<<Filter>>

	Clock	Slack	End Point TNS
1	altera_reserved_tck	47.005	0.000

HOLD SUMMARY



Compilation Report - Linear_Equation_Solver

Table of Contents

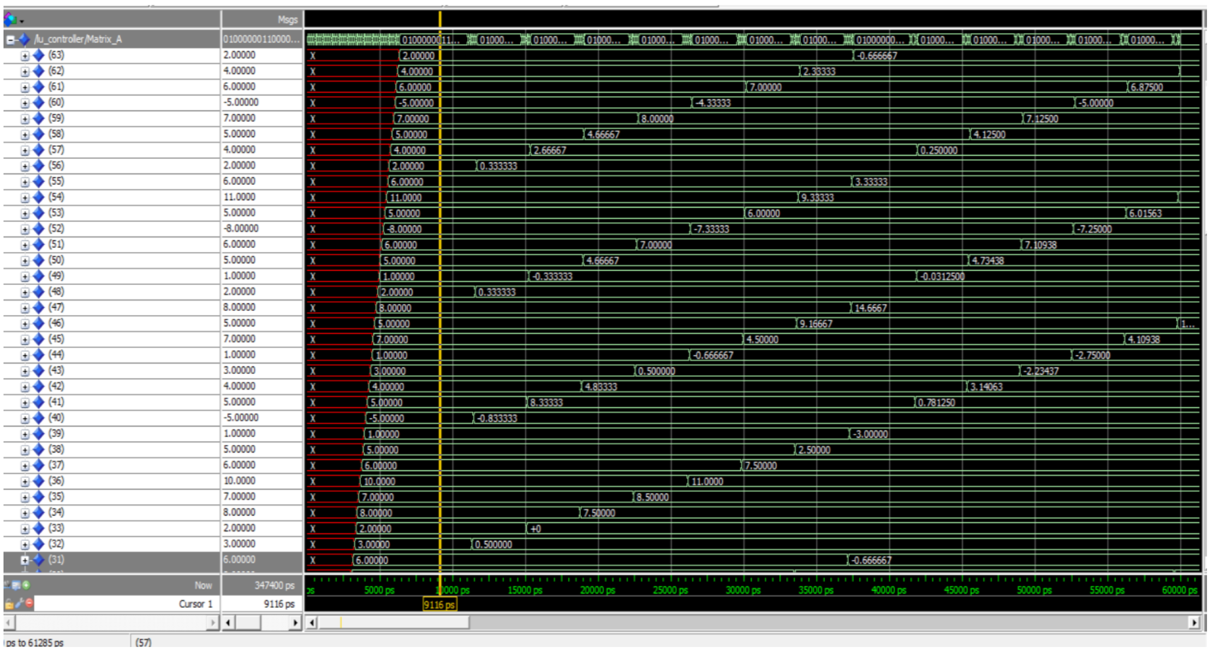
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - Clocks
  - Slow 1200mV 85C Model
  - Slow 1200mV 0C Model
    - Fmax Summary
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Summary

Slow 1200mV 0C Model Hold Summary

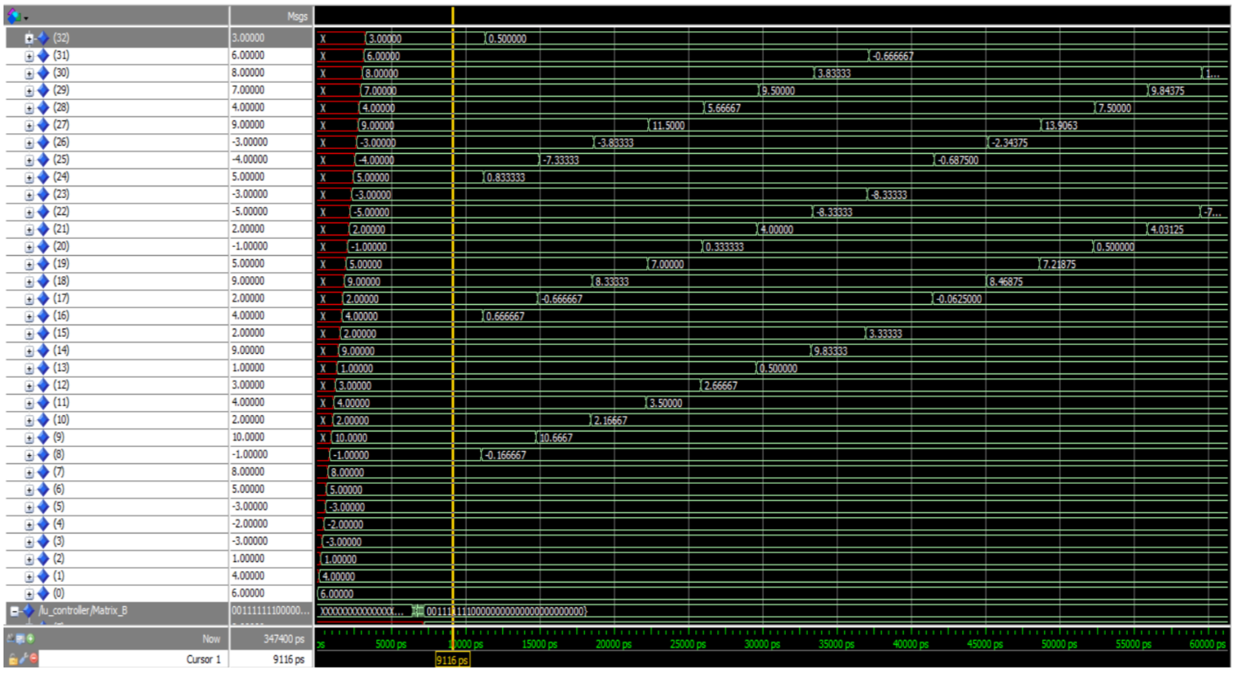
<<Filter>>

	Clock	Slack	End Point TNS
1	altera_reserved_tck	0.311	0.000

A MATRIX BEFORE LU DECOMPOSITION

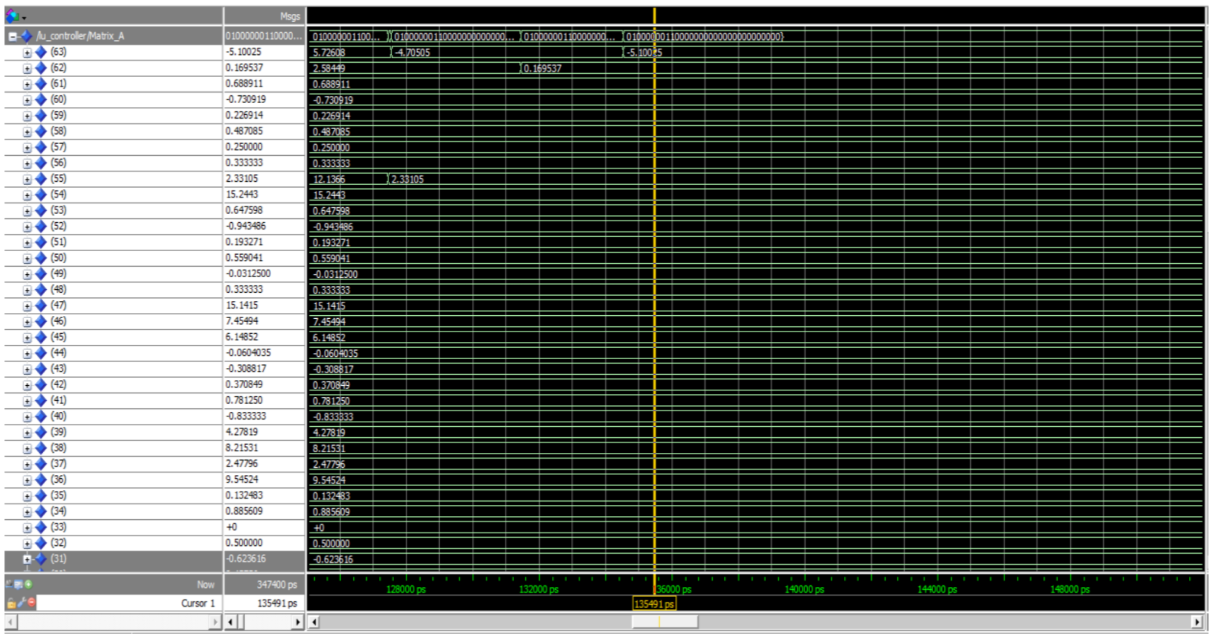


```
                              Msgs
/lu_controller/Matrix_A       01000000110000...
  (63)                        2.00000
  (62)                        4.00000
  (61)                        6.00000
  (60)                        -5.00000
  (59)                        7.00000
  (58)                        5.00000
  (57)                        4.00000
  (56)                        2.00000
  (55)                        6.00000
  (54)                        11.0000
  (53)                        5.00000
  (52)                        -8.00000
  (51)                        6.00000
  (50)                        5.00000
  (49)                        1.00000
  (48)                        2.00000
  (47)                        8.00000
  (46)                        5.00000
  (45)                        7.00000
  (44)                        1.00000
  (43)                        3.00000
  (42)                        4.00000
  (41)                        5.00000
  (40)                        -5.00000
  (39)                        1.00000
  (38)                        5.00000
  (37)                        6.00000
  (36)                        10.0000
  (35)                        7.00000
  (34)                        8.00000
  (33)                        2.00000
  (32)                        3.00000
  (31)                        6.00000
                        Now   347400 ps
                   Cursor 1   9116 ps
ps to 61285 ps     (57)
```

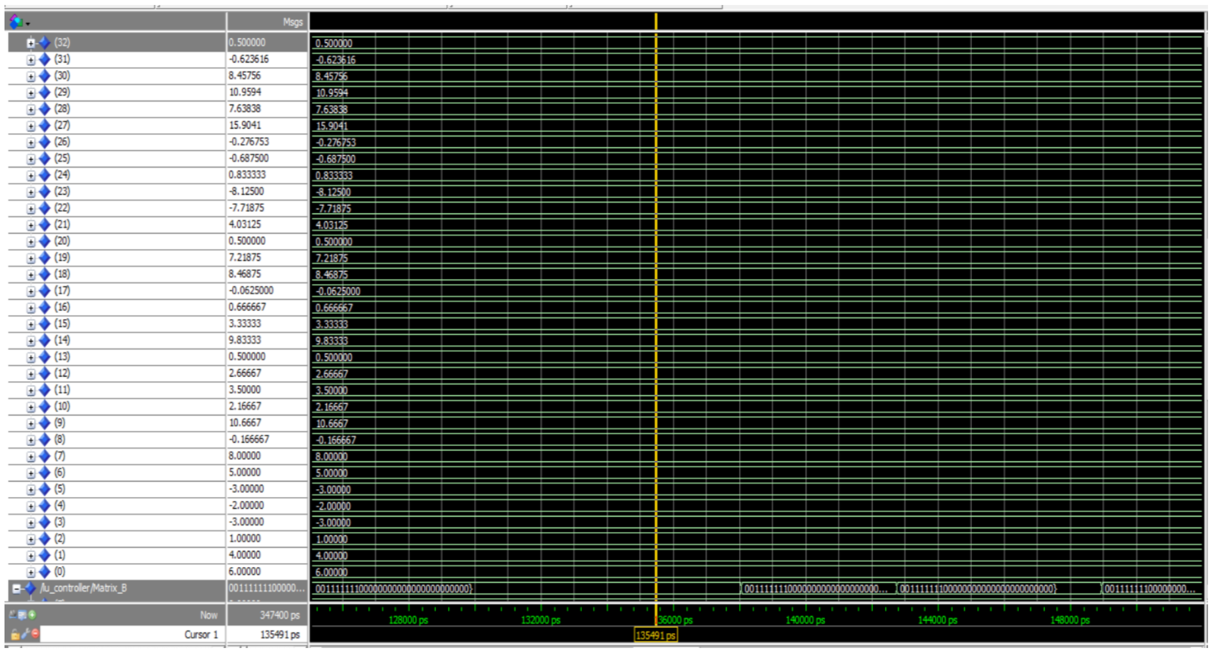


```
                              Msgs
  (32)                        3.00000
  (31)                        6.00000
  (30)                        8.00000
  (29)                        7.00000
  (28)                        4.00000
  (27)                        9.00000
  (26)                        -3.00000
  (25)                        -4.00000
  (24)                        5.00000
  (23)                        -3.00000
  (22)                        -5.00000
  (21)                        2.00000
  (20)                        -1.00000
  (19)                        5.00000
  (18)                        9.00000
  (17)                        2.00000
  (16)                        4.00000
  (15)                        2.00000
  (14)                        9.00000
  (13)                        1.00000
  (12)                        3.00000
  (11)                        4.00000
  (10)                        2.00000
  (9)                         10.0000
  (8)                         -1.00000
  (7)                         8.00000
  (6)                         5.00000
  (5)                         -3.00000
  (4)                         -2.00000
  (3)                         -3.00000
  (2)                         1.00000
  (1)                         4.00000
  (0)                         6.00000
/lu_controller/Matrix_B       001111111100000...
                        Now   347400 ps
                   Cursor 1   9116 ps
```

A MATRIX AFTER LU DECOMPOSITION

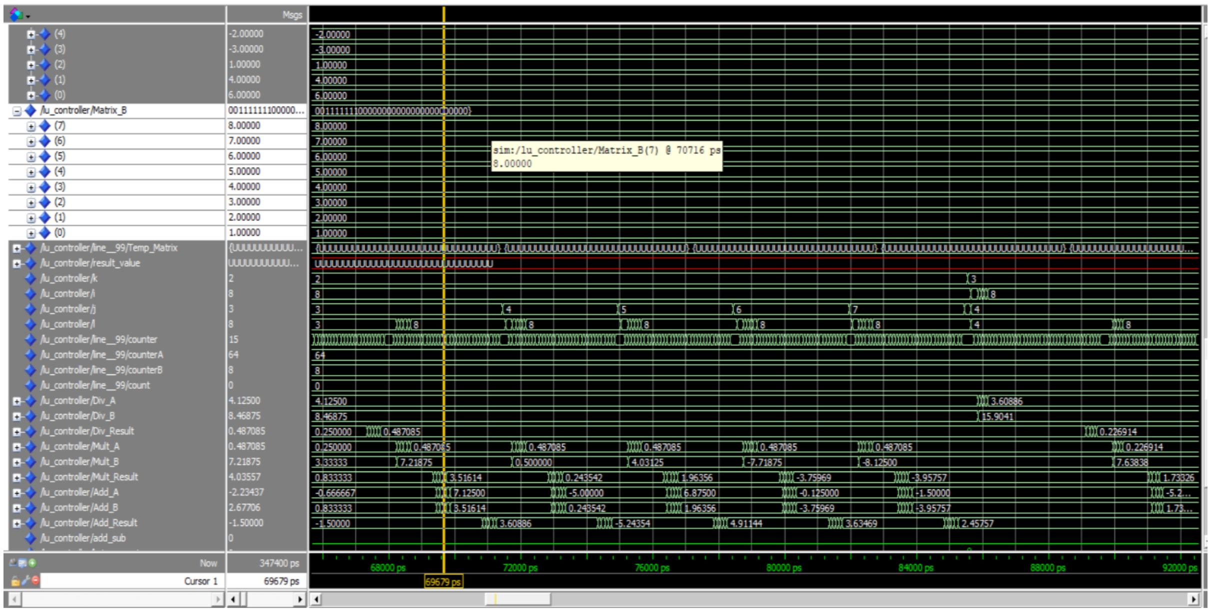


	Msgs	
/lu_controller/Matrix_A	01000000110000...	01000000110 0... 01000000110000000000000... 0100000011000000... 01000001100000000000000000000}
(63)	-5.10025	5.72608 -4.70505 -5.10025
(62)	0.169537	2.58449 0.169537
(61)	0.688911	0.688911
(60)	-0.730919	-0.730919
(59)	0.226914	0.226914
(58)	0.487085	0.487085
(57)	0.250000	0.250000
(56)	0.333333	0.333333
(55)	2.33105	12.1366 2.33105
(54)	15.2443	15.2443
(53)	0.647598	0.647598
(52)	-0.943486	-0.943486
(51)	0.193271	0.193271
(50)	0.559041	0.559041
(49)	-0.0312500	-0.0312500
(48)	0.333333	0.333333
(47)	15.1415	15.1415
(46)	7.45494	7.45494
(45)	6.14852	6.14852
(44)	-0.0604035	-0.0604035
(43)	-0.308817	-0.308817
(42)	0.370849	0.370849
(41)	0.781250	0.781250
(40)	-0.833333	-0.833333
(39)	4.27819	4.27819
(38)	8.21531	8.21531
(37)	2.47796	2.47796
(36)	9.54524	9.54524
(35)	0.132483	0.132483
(34)	0.885609	0.885609
(33)	+0	+0
(32)	0.500000	0.500000
(31)	-0.623616	-0.623616
Now	347400 ps	128000 ps 132000 ps 136000 ps 140000 ps 144000 ps 148000 ps
Cursor 1	135491 ps	135491 ps



	Msgs	
(32)	0.500000	0.500000
(31)	-0.623616	-0.623616
(30)	8.45756	8.45756
(29)	10.9594	10.9594
(28)	7.63838	7.63838
(27)	15.9041	15.9041
(26)	-0.276753	-0.276753
(25)	-0.687500	-0.687500
(24)	0.833333	0.833333
(23)	-8.12500	-8.12500
(22)	-7.71875	-7.71875
(21)	4.03125	4.03125
(20)	0.500000	0.500000
(19)	7.21875	7.21875
(18)	8.46875	8.46875
(17)	-0.0625000	-0.0625000
(16)	0.666667	0.666667
(15)	3.33333	3.33333
(14)	9.83333	9.83333
(13)	0.500000	0.500000
(12)	2.66667	2.66667
(11)	3.50000	3.50000
(10)	2.16667	2.16667
(9)	10.6667	10.6667
(8)	-0.166667	-0.166667
(7)	8.00000	8.00000
(6)	5.00000	5.00000
(5)	-3.00000	-3.00000
(4)	-2.00000	-2.00000
(3)	-3.00000	-3.00000
(2)	1.00000	1.00000
(1)	4.00000	4.00000
(0)	6.00000	6.00000
/lu_controller/Matrix_B	00111111100000...	00111111100000000000000000000000} 0011111110000000000000000... 00111111100000000000000000000000} 0011111110000000...
Now	347400 ps	128000 ps 132000 ps 136000 ps 140000 ps 144000 ps 148000 ps
Cursor 1	135491 ps	135491 ps

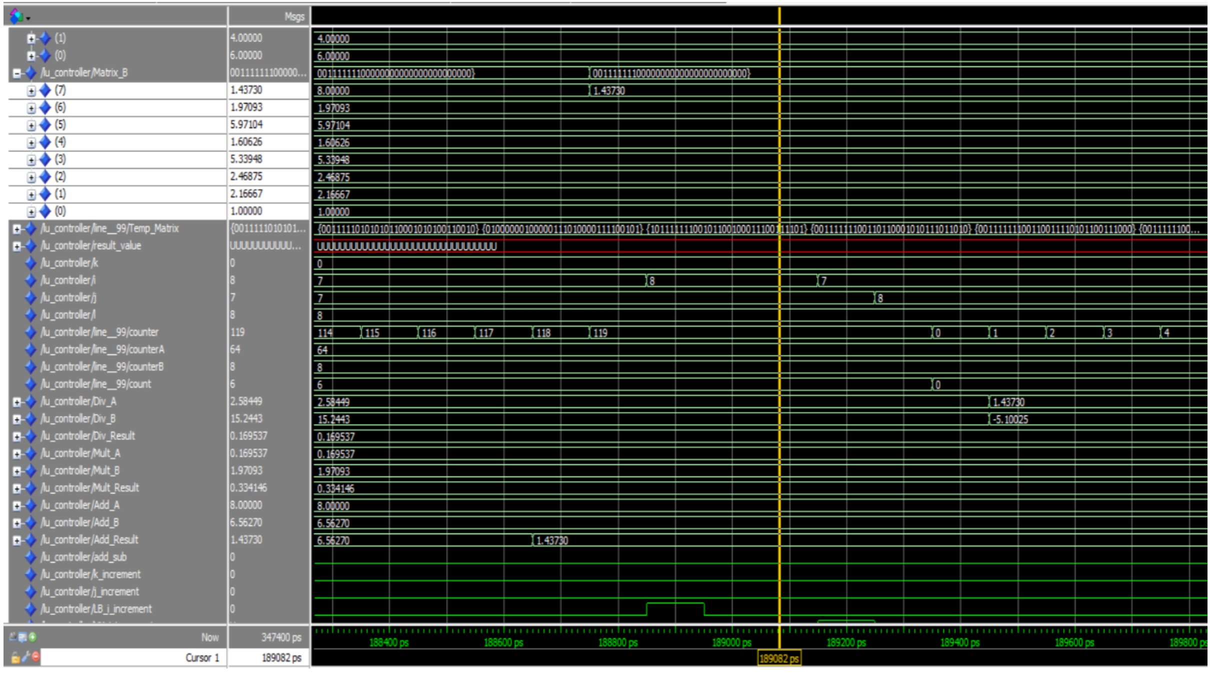
B MATRIX



```
                                              Msgs
  (4)                                         -2.00000
  (3)                                         -3.00000
  (2)                                         1.00000
  (1)                                         4.00000
  (0)                                         6.00000
/lu_controller/Matrix_B                       00111111100000...
  (7)                                         8.00000
  (6)                                         7.00000
  (5)                                         6.00000
  (4)                                         5.00000
  (3)                                         4.00000
  (2)                                         3.00000
  (1)                                         2.00000
  (0)                                         1.00000
/lu_controller/line__99/Temp_Matrix           {UUUUUUUUUUU...
/lu_controller/result_value                   UUUUUUUUUUU...
/lu_controller/k                              2
/lu_controller/i                              8
/lu_controller/j                              3
/lu_controller/l                              8
/lu_controller/line__99/counter               15
/lu_controller/line__99/counterA              64
/lu_controller/line__99/counterB              8
/lu_controller/line__99/count                 0
/lu_controller/Div_A                          4.12500
/lu_controller/Div_B                          8.46875
/lu_controller/Div_Result                     0.487085
/lu_controller/Mult_A                         0.487085
/lu_controller/Mult_B                         7.21875
/lu_controller/Mult_Result                    4.03557
/lu_controller/Add_A                          -2.23437
/lu_controller/Add_B                          2.67706
/lu_controller/Add_Result                     -1.50000
/lu_controller/add_sub                        0
Now                                           347400 ps
Cursor 1                                      69679 ps
```

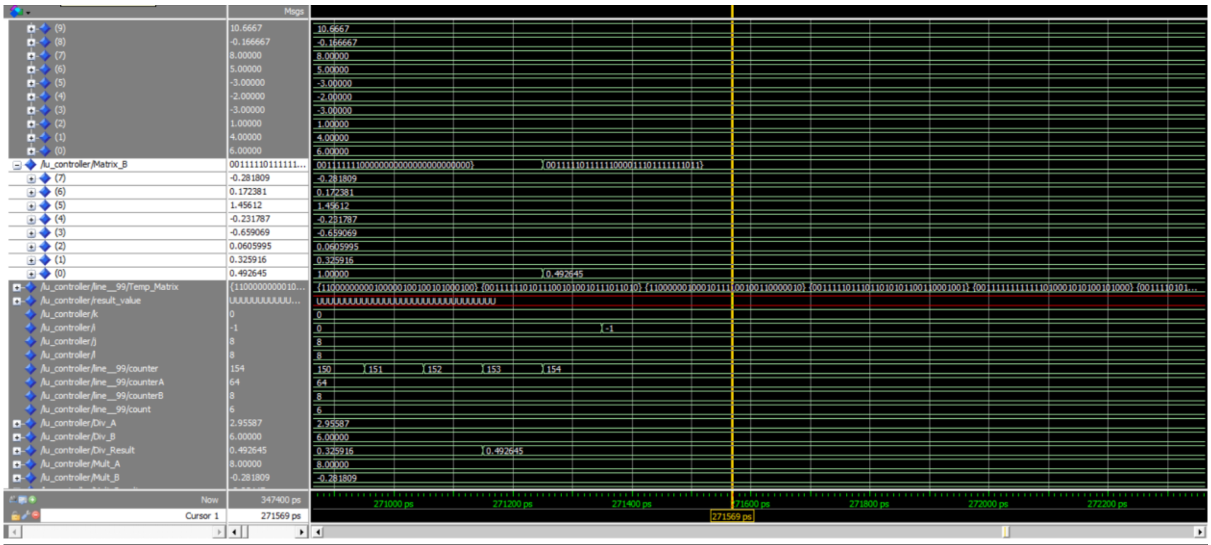
```
sim:/lu_controller/Matrix_B(7) @ 70716 ps
8.00000
```

Y MATRIX AFTER EXECUTING LY=B



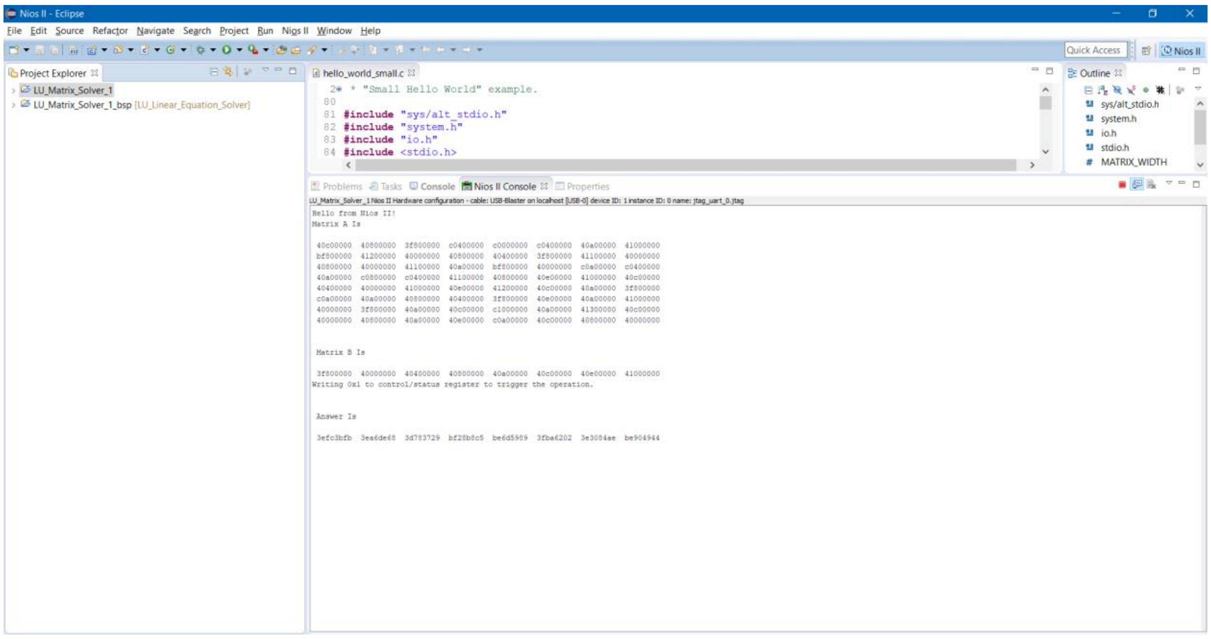
```
                                              Msgs
  (1)                                         4.00000
  (0)                                         6.00000
/lu_controller/Matrix_B                       00111111100000...
  (7)                                         1.43730
  (6)                                         1.97093
  (5)                                         5.97104
  (4)                                         1.60626
  (3)                                         5.33948
  (2)                                         2.46875
  (1)                                         2.16667
  (0)                                         1.00000
/lu_controller/line__99/Temp_Matrix           {0011111010101...
/lu_controller/result_value                   UUUUUUUUUUU...
/lu_controller/k                              0
/lu_controller/i                              8
/lu_controller/j                              7
/lu_controller/l                              8
/lu_controller/line__99/counter               119
/lu_controller/line__99/counterA              64
/lu_controller/line__99/counterB              8
/lu_controller/line__99/count                 6
/lu_controller/Div_A                          2.58449
/lu_controller/Div_B                          15.2443
/lu_controller/Div_Result                     0.169537
/lu_controller/Mult_A                         0.169537
/lu_controller/Mult_B                         1.97093
/lu_controller/Mult_Result                    0.334146
/lu_controller/Add_A                          8.00000
/lu_controller/Add_B                          6.56270
/lu_controller/Add_Result                     1.43730
/lu_controller/add_sub                        0
/lu_controller/k_increment                    0
/lu_controller/j_increment                    0
/lu_controller/LB_i_increment                 0
Now                                           347400 ps
Cursor 1                                      189082 ps
```

Final Solution after UX=Y



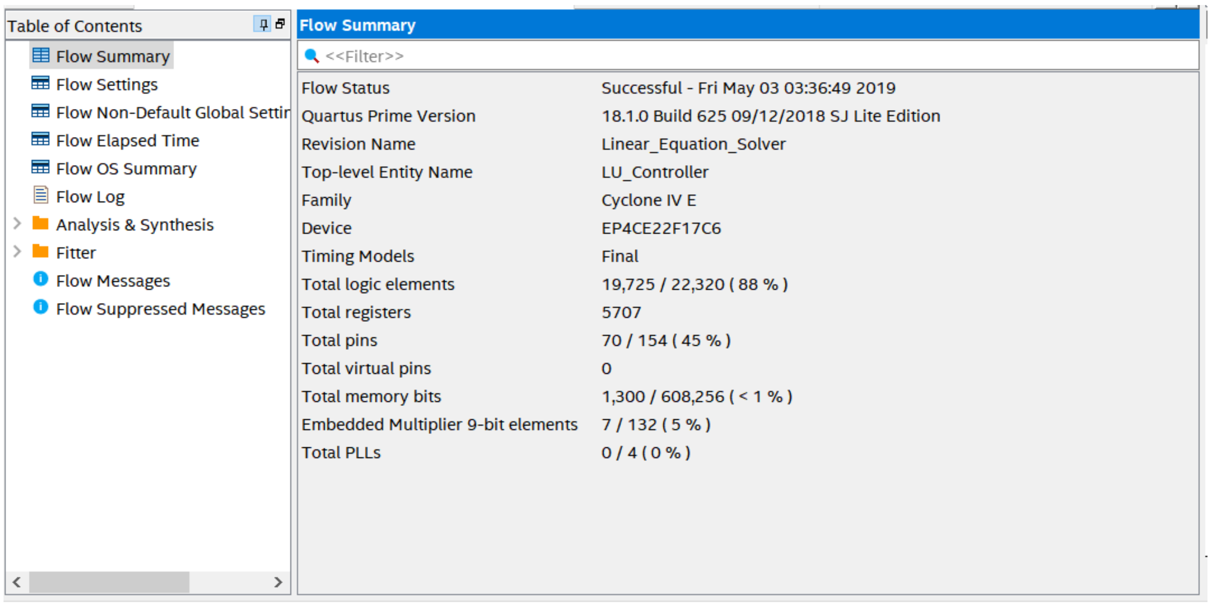
```
                                    Msgs
   (9)                              10.6667
   (8)                              -0.166667
   (7)                              8.00000
   (6)                              5.00000
   (5)                              -3.00000
   (4)                              -2.00000
   (3)                              -3.00000
   (2)                              1.00000
   (1)                              4.00000
   (0)                              6.00000
/lu_controller/Matrix_B             00111110111111...
   (7)                              -0.281809
   (6)                              0.172381
   (5)                              1.45612
   (4)                              -0.231787
   (3)                              -0.659069
   (2)                              0.0605995
   (1)                              0.325916
   (0)                              0.492645
/lu_controller/line__99/Temp_Matrix {110000000000010...
/lu_controller/result_value         UUUUUUUUUUU...
/lu_controller/k                    0
/lu_controller/i                    -1
/lu_controller/j                    8
/lu_controller/l                    8
/lu_controller/line__99/counter     154
/lu_controller/line__99/counterA    64
/lu_controller/line__99/counterB    8
/lu_controller/line__99/count       6
/lu_controller/Div_A                2.95587
/lu_controller/Div_B                6.00000
/lu_controller/Div_Result           0.492645
/lu_controller/Mult_A               8.00000
/lu_controller/Mult_B               -0.281809
                         Now        347400 ps
                    Cursor 1        271569 ps
```

Output from NIOS II



```
LU_Matrix_Solver_1 Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtag_uart_0.jtag
Hello from Nios II!
Matrix A Is

 40c00000  40800000  3f800000  c0400000  c0000000  c0400000  40a00000  41000000
 bf800000  41200000  40000000  40800000  40400000  3f800000  41100000  40000000
 40800000  40000000  41100000  40a00000  bf800000  40000000  c0a00000  c0400000
 40a00000  c0800000  c0400000  41100000  40800000  40e00000  41000000  40c00000
 40400000  40000000  41000000  40e00000  41200000  40c00000  40a00000  3f800000
 c0a00000  40a00000  40800000  40400000  3f800000  40e00000  40a00000  41000000
 40000000  3f800000  40a00000  40c00000  c1000000  40a00000  41300000  40c00000
 40000000  40800000  40a00000  40e00000  c0a00000  40c00000  40800000  40000000


 Matrix B Is

 3f800000  40000000  40400000  40800000  40a00000  40c00000  40e00000  41000000
Writing 0x1 to control/status register to trigger the operation.


 Answer Is

 3efc3bfb  3ea6de68  3d783729  bf28b8c5  be6d5989  3fba6202  3e3084ae  be904944
```

FITTER ANALYSIS



Flow Status	Successful - Fri May 03 03:36:49 2019
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Linear_Equation_Solver
Top-level Entity Name	LU_Controller
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	19,725 / 22,320 (88 %)
Total registers	5707
Total pins	70 / 154 (45 %)
Total virtual pins	0
Total memory bits	1,300 / 608,256 (< 1 %)
Embedded Multiplier 9-bit elements	7 / 132 (5 %)
Total PLLs	0 / 4 (0 %)

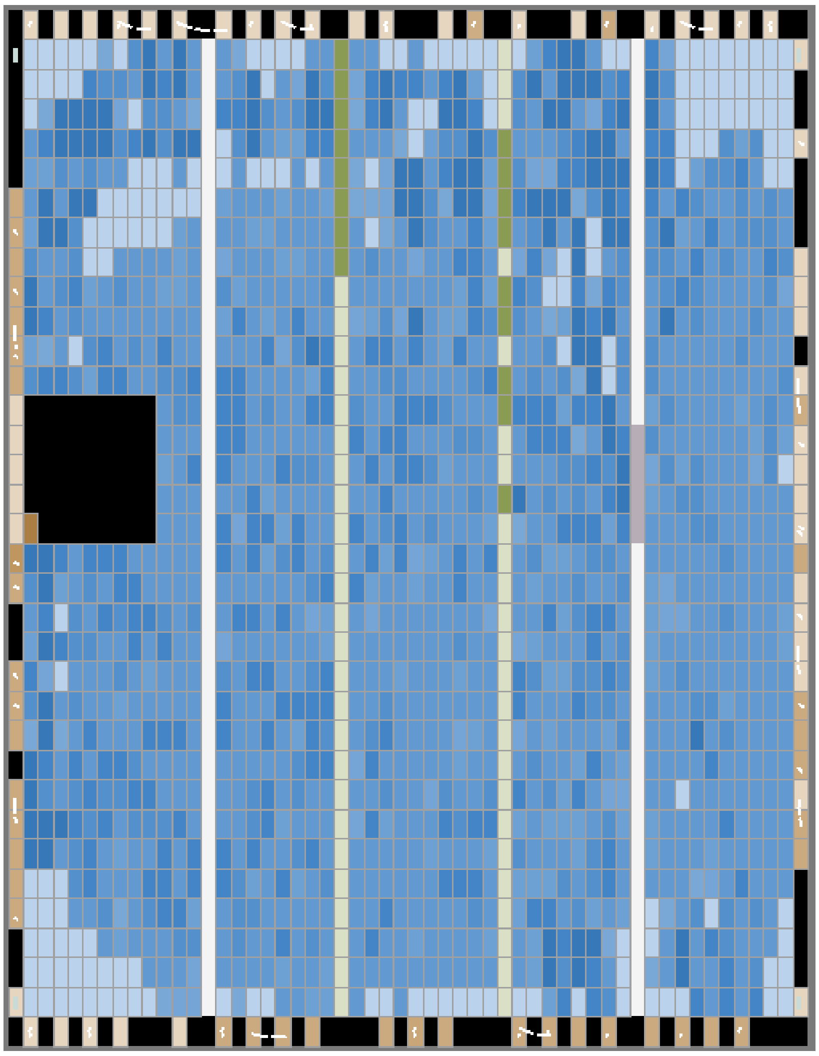
LOGIC ELEMENTS USED- 19725/22320 (95%)

TOTAL PINS – 70/154

THE DESIGN , IS FITTED PROPERLY INTO THE DE0-NANO FGPA , AS LONG AS FPGA HAS ENOUGH MEMORY .

THE DESIGN SUCCESSFULLY IMPLEMENTS A N×N MATRIX AND THUS A N×N LINEAR EQUATION SOLVER

CHIP-PLANNER



FUTURE SCOPE

1 It is assumed that the matrix entered is already pivoted.

2 The design encounters difficulties if diagonal elements are zero.

REFERENCES

1 Portable and Scalable FPGA-Based Acceleration of a Direct Linear System Solver by Wei Zhang , 2008, Department of Electrical and Computer Engineering University of Toronto.

2 Steven Chapra and Raymond Canale, Numerical Methods for Engineers, Fourth Edition, McGrawHill, 2002 (see Sections 10.1-10.2)

3 http://csep10.phys.utk.edu/guidry/phys594/lectures/linear_algebra/