Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

An internship report submitted by

DALAALI SHAHEEN - URK21CS2018
EBIN ALEX - URK21CS2044
ESSWARI S - URK21CS2065

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Dr. P. GETZI JEBA LEELIPUSHPAM

Mr. ABHISHEK NANDY

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec-3 of the UGC Act, 1956)

Karunya Nagar, Coimbatore - 641 114. INDIA

Team Name: The Elites

Team Members

The project was carried out by the following team members:

- Ebin Alex - ebinalex@karunya.edu.in
- Dalaali Shaheen - dalaalishaheen@karunya.edu.in
- Esswari S - esswaris@karunya.edu.in

Mentor

We would like to express our gratitude to our mentors for providing guidance and support throughout the internship:

- Dr. P. Getzi Jeba (Academic Mentor) - getzi@karunya.edu
- Abhishek Nandy (Industry Mentor) - abhisheknandy@theprograms.in

CHAPTER - 1

Introduction

This report contains the project developed during the Intel Unnati internship by the team "The Elites" at Intel's FPGA Division. The objective of the internship was to work on slow and fast division algorithms in computer architecture, specifically targeting the Intel FPGA Cyclone V.

Development Environment

The code was developed using the Intel Quartus Prime Lite platform. The implementation is written in Verilog, a hardware description language widely used in FPGA design.

Objectives

- To develop the Finite State machine (Mealy / Moore) based on the design specifications.
- To develop the Verilog HDL code for the developed FSM.
- To simulate and verify the design for its functionality with proper test cases.
- To synthesize the design and perform timing analysis.
- To implement the design on the targeted FPGA.

Summary

- Week 1 - We have developed the block diagram of the given design; identified proper inputs and outputs along with designing the FSM and validated the test cases.
- Week 2 - We have developed the Verilog code of the given FSM, developed the testbench and verified the functionality and analyzed the code coverage reports.
- Week 3 - We have synthesized the developed HDL code, analyzed the area and power report and applied proper timing constraints and analyzed the timing reports to achieve proper timing closure.
- Week 4 - We have implemented the design on the targeted FPGA, developed the project report, uploaded the code, working model along with the video demo on the github site.

Video link

- Newton Raphson Method - Fast_division - Newton_Raphson
- Restoring Method - Slow_division - Restoring_Division

Github link - Team_The_Elites

CHAPTER - 2

Fast Division Algorithm

Algorithm

Step 1: Initialize the inputs - dividend, divisor and count(n=1,2,...).
Step 2: Compute r(n) using the formula.
Step 3: If the value of |f(r(n))| is the most closest value to the dividend, displaying the value which is the output. If not, increment the value of count.
Step 4: Repeat the steps 3 and 4 until the condition is met.
Step 5: The output should display the corresponding quotient and remainder.
Step 6: Check the results.

Example Operation:

Consider the following: Dividend: 10 (in binary: 1010) Divisor: 3 (in binary: 0011)

Step 1: Initialize the variables:
x0 = 0
x1 = 0
x2 = 0
x3 = 0
x4 = 0

Step 2: Assign the values of the dividend and divisor to x0 and divisor, respectively:
x0 = 1010 (dividend)
divisor = 0011 (divisor)

Step 3: Calculate x1:
x1 = x0 - (x0 / divisor) * divisor
x1 = 1010 - (1010 / 0011) * 0011
x1 = 1010 - (5 * 0011)
x1 = 1010 - 0110
x1 = 0100

Step 4: Calculate x2:
x2 = x1 - (x1 / divisor) * divisor
x2 = 0100 - (0100 / 0011) * 0011
x2 = 0100 - (2 * 0011)
x2 = 0100 - 0110
x2 = 1110 (carry out of MSB)

Step 5: Calculate x3:
x3 = x2 - (x2 / divisor) * divisor
x3 = 1110 - (1110 / 0011) *
0011
x3 = 1110 - (6 * 0011)
x3 = 1110 - 1101
x3 = 0001

Step 6: Calculate x4 (quotient):
x4 = x0 / divisor
x4 = 1010 / 0011
x4 = 0011

The final results are:
quotient = 0011 (3 in decimal)
remainder = 0001 (1 in decimal)

So, when dividing the dividend 10 by the divisor 3 using the provided module, the quotient is 3, and the remainder is 1, which matches the expected results.

Flowchart

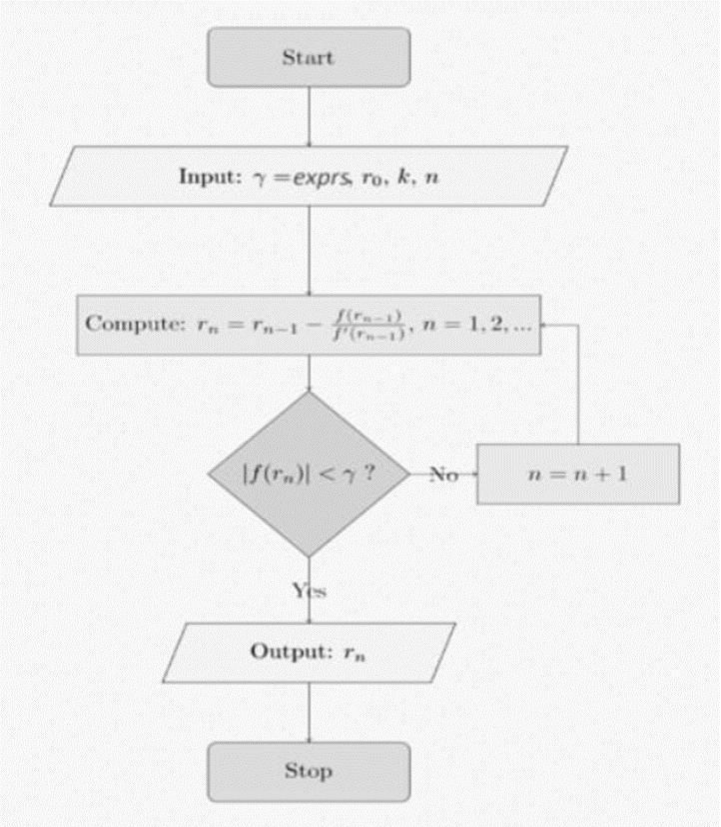

Fig. 1 - Flowchart of newton raphson division

FSM

Since the Newton-Raphson division algorithm does not lend itself directly to a finite state machine (FSM) representation, it is not possible to create a traditional FSM diagram for it. The Newton-Raphson division algorithm is an iterative numerical method that relies on repeated calculations and comparisons until a desired level of precision is achieved.

Slow Division algorithm

Algorithm

Registers used: A, M, Q, n (counter)

Step 1: Load the initial values for the registers.
A = 0 (Accumulator), Qres = 0, M = Divisor, Q = Dividend and n is the count value which equals the number of bits of dividend.
Step 2: Shift left {A,Q}.
Step 3: Perform A = A - M.
Step 4: Check the sign bit of A. If 0, goto step 5. If 1, goto step 6.
Step 5: Set LSB of Q as 0. Goto step 7.
Step 6: Set LSB of Q as 1. Restore the value of A which was present before the subtraction.
Step 7: Decrement count.
Step 8: Check if counter value n is zero. If yes, goto next step. Else, goto step 3.
Step 9: Stop.

Example Operation:

Consider the following: Dividend = 15 (place in Q) and Divisor = 8 (place in M)

Count n = 4

Count	Operation	A	Q
4	Initial Values	0000	1111
	Shift Left A,Q	0001	1110
	A = A - M	1001	1110
	Q[0] = 0, Restore A	0001	1110
3	Shift Left A,Q	0011	1100
	A = A - M	1011	1100
	Q[0] = 0, Restore A	0011	1100
2	Shift Left A,Q	0111	1000
	A = A - M	1111	1000
	Q[0] = 0, Restore A	0111	1000
1	Shift Left A,Q	1111	0000
	A = A - M	0111	0000
	Q[0] = 1	0111	0001
0	Final Result: Remainder = 0111 = 7, Quotient = 0001 = 1		

Flowchart

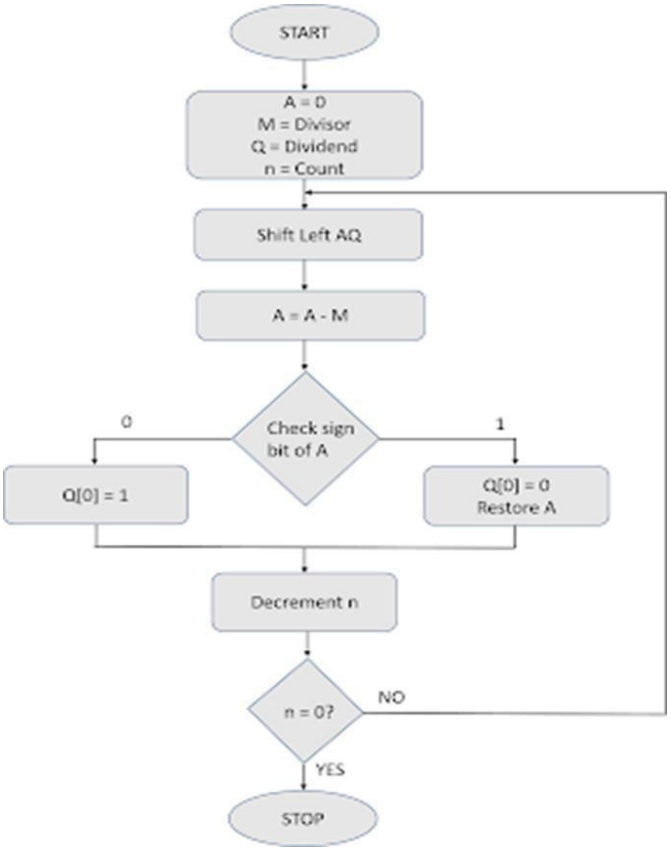

Fig. 2 - Flowchart of restoring division

FSM Diagram - MEALY MACHINE

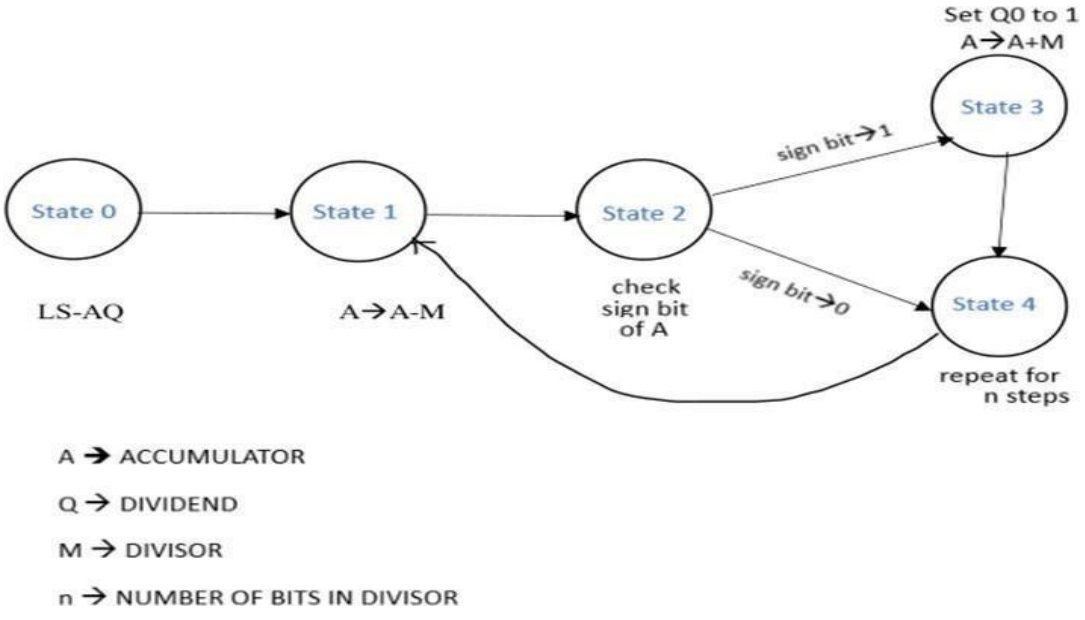

Fig. 3 - FSM of restoring division (mealy machine)

Analysis Diagram

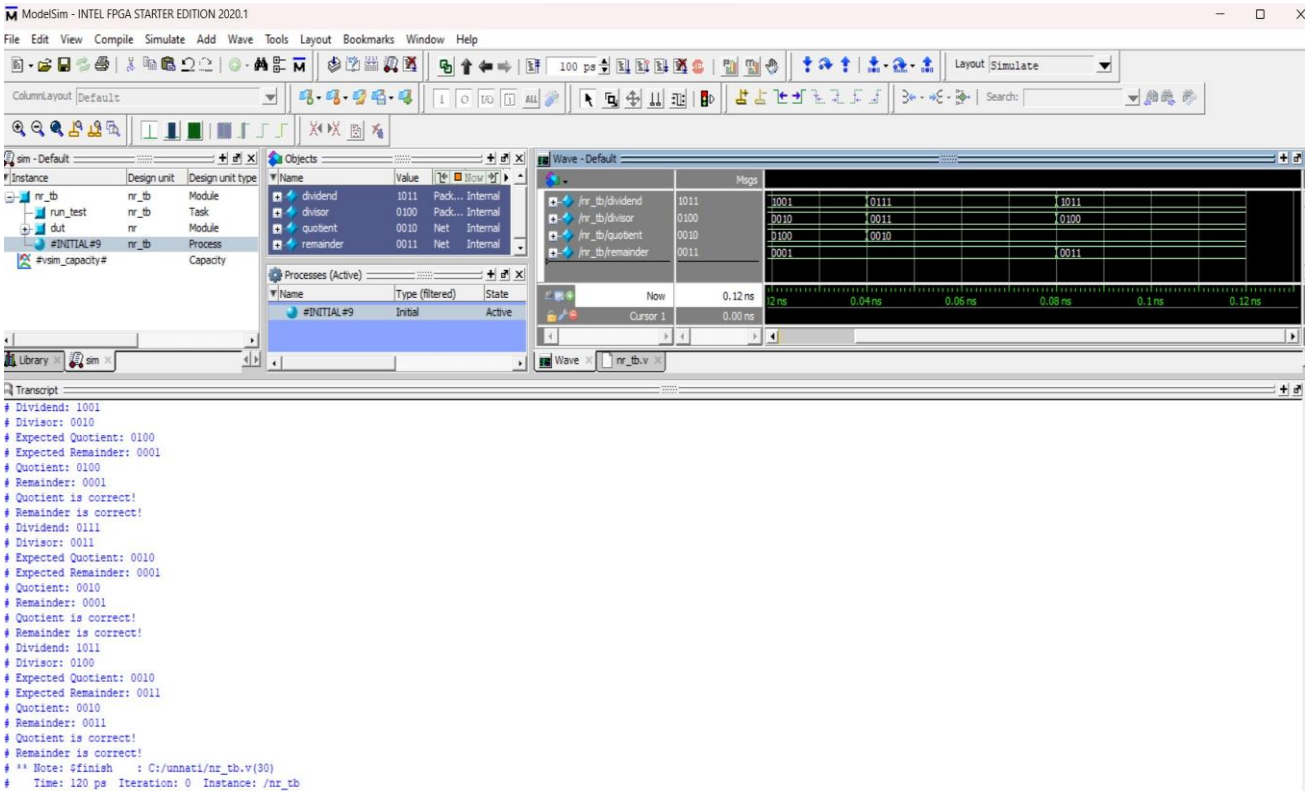

Fig. 4(a) - Analysis diagram of fast division

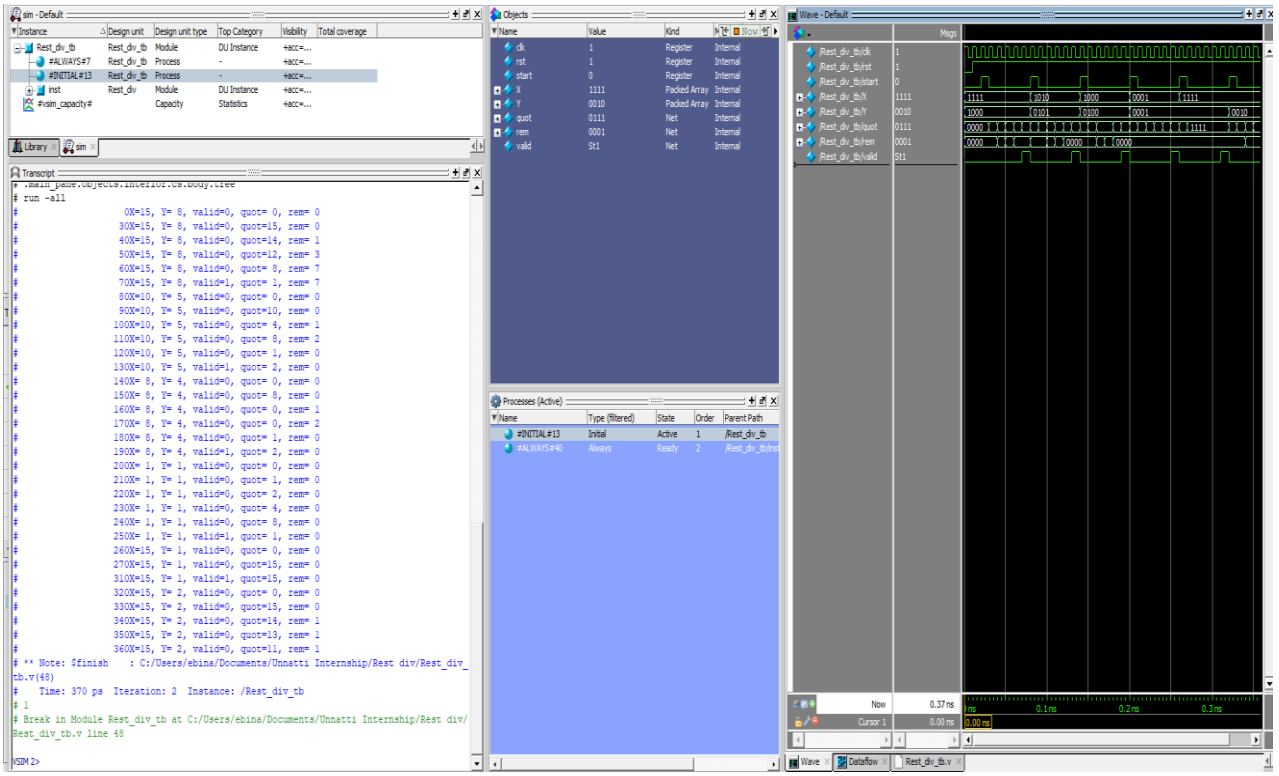

Fig. 4(a) - Analysis diagram of slow division

CHAPTER - 3

FPGA Implementation:

Fast Division Algorithm

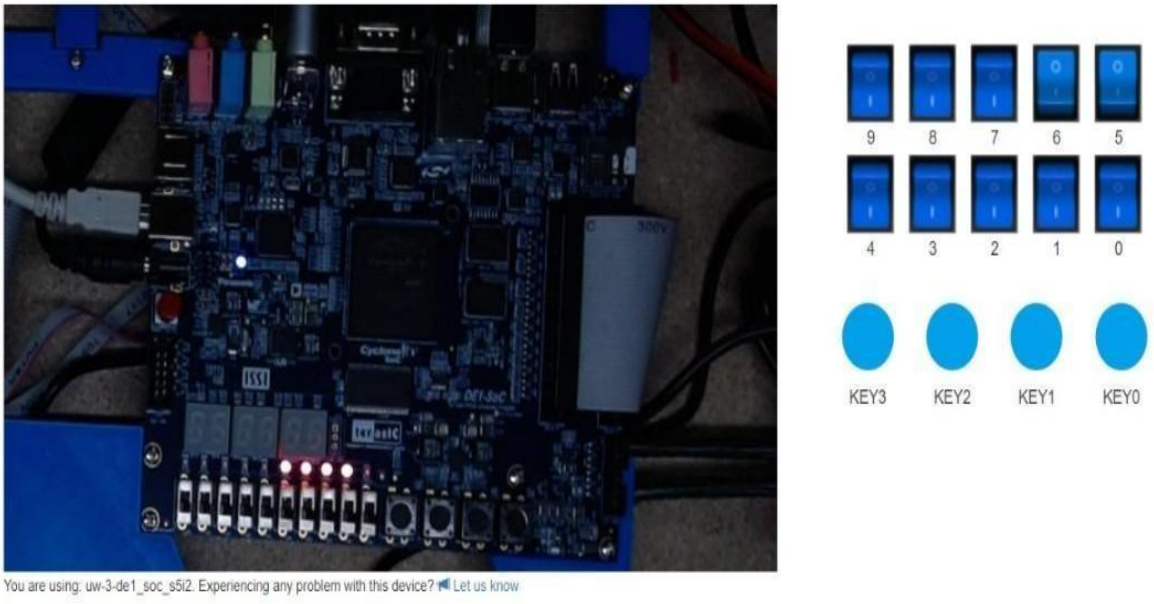

Fig. 5(a) - Labsland (fast division algorithm)

Area report

Fitter Resource Usage Summary

<<Filter>>

	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	34 / 32,070	< 1 %
2	ALMs needed [=A-B+C]	34	
1	[A] ALMs used in final placement [=a+b+c+d]	34 / 32,070	< 1 %
1	[a] ALMs used for LUT logic and registers	0	
2	[b] ALMs used for LUT logic	34	
3	[c] ALMs used for registers	0	
4	[d] ALMs used for memory (up to half of total ALMs)	0	
2	[B] Estimate of ALMs recoverable by dense packing	0 / 32,070	0 %
3	[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 32,070	0 %
1	[a] Due to location constrained logic	0	
2	[b] Due to LAB-wide signal conflicts	0	
3	[c] Due to LAB input limits	0	
4	[d] Due to virtual I/Os	0	
3			
4	Difficulty packing design	Low	
5			
6	Total LABs: partially or completely used	6 / 3,207	< 1 %
1	-- Logic LABs	6	
2	-- Memory LABs (up to half of total LABs)	0	
7			
8	Combinational ALUT usage for logic	68	
1	-- 7 input functions	0	
2	-- 6 input functions	0	
3	-- 5 input functions	12	
4	-- 4 input functions	18	

Fitter Resource Usage Summary

<<Filter>>

	Resource	Usage	%
5	-- <=3 input functions	38	
9	Combinational ALUT usage for route-throughs	0	
10			
11	Dedicated logic registers	0	
1	-- By type:		
1	-- Primary logic registers	0 / 64,140	0 %
2	-- Secondary logic registers	0 / 64,140	0 %
2	-- By function:		
1	-- Design implementation registers	0	
2	-- Routing optimization registers	0	
12			
13	Virtual pins	0	
14	I/O pins	16 / 457	4 %
1	-- Clock pins	0 / 8	0 %
2	-- Dedicated input pins	0 / 21	0 %
15			
16	Hard processor system peripheral utilization		
1	-- Boot from FPGA	0 / 1 (0 %)	
2	-- Clock resets	0 / 1 (0 %)	
3	-- Cross trigger	0 / 1 (0 %)	
4	-- S2F AXI	0 / 1 (0 %)	
5	-- F2S AXI	0 / 1 (0 %)	
6	-- AXI Lightweight	0 / 1 (0 %)	
7	-- SDRAM	0 / 1 (0 %)	
8	-- Interrupts	0 / 1 (0 %)	

Fig. 6(a) - Area report of fast division algorithm

Power report

215031 Total thermal power estimate for the design is 420.58 mW

Quartus Prime Power Analyzer was successful. 0 errors, 5 warnings

Fig. 7(a) - Power report of fast division algorithm

Pin planning

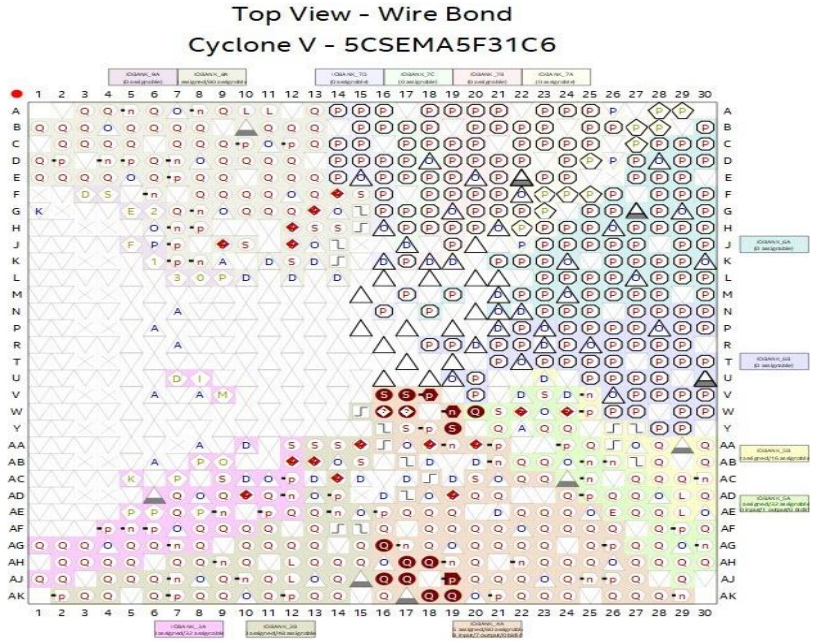

Fig. 8(a) - Pin planning of fast division algorithm

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	
dividend[3]	Input	PIN_AK18	4A	B4A_N0	PIN_AK18	3.3-V LVTTL		16mA (default)	
dividend[2]	Input	PIN_AK19	4A	B4A_N0	PIN_AK19	3.3-V LVTTL		16mA (default)	
dividend[1]	Input	PIN_AJ19	4A	B4A_N0	PIN_AJ19	3.3-V LVTTL		16mA (default)	
dividend[0]	Input	PIN_AJ17	4A	B4A_N0	PIN_AJ17	3.3-V LVTTL		16mA (default)	
divisor[3]	Input	PIN_AJ16	4A	B4A_N0	PIN_AJ16	3.3-V LVTTL		16mA (default)	
divisor[2]	Input	PIN_AH18	4A	B4A_N0	PIN_AH18	3.3-V LVTTL		16mA (default)	
divisor[1]	Input	PIN_AH17	4A	B4A_N0	PIN_AH17	3.3-V LVTTL		16mA (default)	
divisor[0]	Input	PIN_AG16	4A	B4A_N0	PIN_AG16	3.3-V LVTTL		16mA (default)	
quotient[3]	Output	PIN_V16	4A	B4A_N0	PIN_V16	3.3-V LVTTL		16mA (default)	1
quotient[2]	Output	PIN_W16	4A	B4A_N0	PIN_W16	3.3-V LVTTL		16mA (default)	1
quotient[1]	Output	PIN_V17	4A	B4A_N0	PIN_V17	3.3-V LVTTL		16mA (default)	1
quotient[0]	Output	PIN_V18	4A	B4A_N0	PIN_V18	3.3-V LVTTL		16mA (default)	1
remainder[3]	Output	PIN_W17	4A	B4A_N0	PIN_W17	3.3-V LVTTL		16mA (default)	1
remainder[2]	Output	PIN_W19	4A	B4A_N0	PIN_W19	3.3-V LVTTL		16mA (default)	1
remainder[1]	Output	PIN_Y19	4A	B4A_N0	PIN_Y19	3.3-V LVTTL		16mA (default)	1
remainder[0]	Output	PIN_W20	5A	B5A_N0	PIN_W20	3.3-V LVTTL		16mA (default)	1
<<new node>>									

Fig. 8(b) - Pin planning of fast division algorithm

Slow Division Algorithm

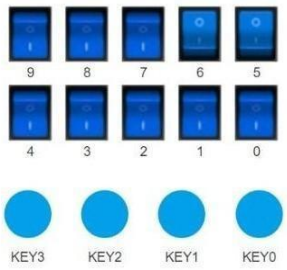

Fig. 5(b) - Labsland of slow division algorithm

Area report

Fitter Resource Usage Summary

<<Filter>>

	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	9 / 32,070	< 1 %
2	ALMs needed [=A-B+C]	9	
1	[A] ALMs used in final placement [=a+b+c+d]	9 / 32,070	< 1 %
1	[a] ALMs used for LUT logic and registers	6	
2	[b] ALMs used for LUT logic	3	
3	[c] ALMs used for registers	0	
4	[d] ALMs used for memory (up to half of total ALMs)	0	
2	[B] Estimate of ALMs recoverable by dense packing	0 / 32,070	0 %
3	[C] Estimate of ALMs unavailable [=a+b+c+d]	0 / 32,070	0 %
1	[a] Due to location constrained logic	0	
2	[b] Due to LAB-wide signal conflicts	0	
3	[c] Due to LAB input limits	0	
4	[d] Due to virtual I/Os	0	
3			
4	Difficulty packing design	Low	
5			
6	Total LABs: partially or completely used	2 / 3,207	< 1 %
1	-- Logic LABs	2	
2	-- Memory LABs (up to half of total LABs)	0	

Fig. 6(c) - Area report of slow division algorithm

	Resource	Usage	%
8	▼ Combinational ALUT usage for logic	17	
1	-- 7 input functions	0	
2	-- 6 input functions	0	
3	-- 5 input functions	0	
4	-- 4 input functions	8	
5	-- <=3 input functions	9	
9	Combinational ALUT usage for route-throughs	0	
10			
11	▼ Dedicated logic registers	12	
1	▼ -- By type:		
1	-- Primary logic registers	12 / 64,140	< 1 %
2	-- Secondary logic registers	0 / 64,140	0 %
2	▼ -- By function:		
1	-- Design implementation registers	12	
2	-- Routing optimization registers	0	
12			
13	Virtual pins	0	
14	▼ I/O pins	20 / 457	4 %
1	-- Clock pins	1 / 8	13 %
2	-- Dedicated input pins	0 / 21	0 %

Fig. 6(c) - Area report of slow division algorithm

Power report

```
215049 Average toggle rate for this design is 0.000 millions of transitions / sec
215031 Total thermal power estimate for the design is 420.82 mW
       Quartus Prime Power Analyzer was successful. 0 errors, 6 warnings
```

Fig. 7(b) - Power report of slow division algorithm

Pin planning

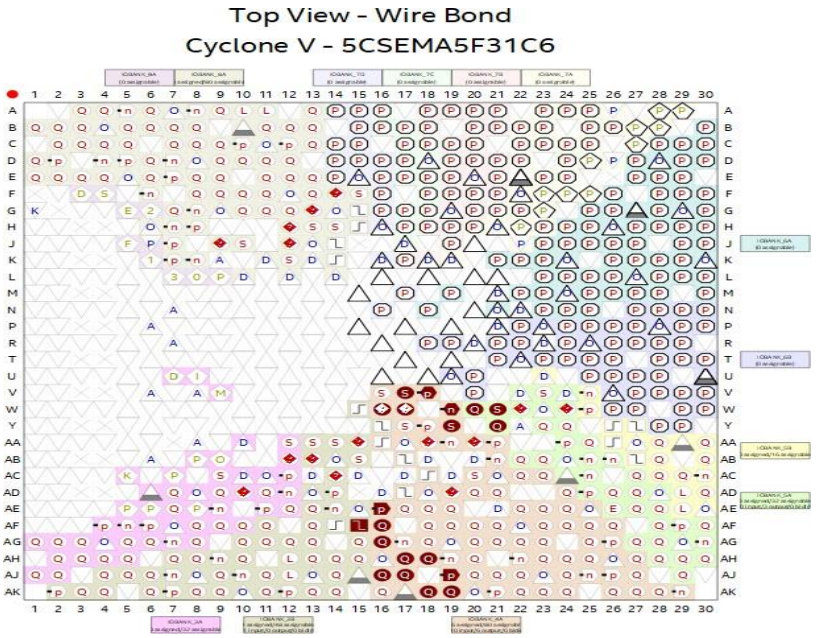

Fig. 8(c) - Pin planning of slow division algorithm

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
X[3]	Input	PIN_AK18	4A	B4A_N0	PIN_AK18	3.3-V LVTTL		16mA (default)	
X[2]	Input	PIN_AK19	4A	B4A_N0	PIN_AK19	3.3-V LVTTL		16mA (default)	
X[1]	Input	PIN_AJ19	4A	B4A_N0	PIN_AJ19	3.3-V LVTTL		16mA (default)	
X[0]	Input	PIN_AJ17	4A	B4A_N0	PIN_AJ17	3.3-V LVTTL		16mA (default)	
Y[3]	Input	PIN_AJ16	4A	B4A_N0	PIN_AJ16	3.3-V LVTTL		16mA (default)	
Y[2]	Input	PIN_AH18	4A	B4A_N0	PIN_AH18	3.3-V LVTTL		16mA (default)	
Y[1]	Input	PIN_AH17	4A	B4A_N0	PIN_AH17	3.3-V LVTTL		16mA (default)	
Y[0]	Input	PIN_AG16	4A	B4A_N0	PIN_AG16	3.3-V LVTTL		16mA (default)	
clk	Input	PIN_AE16	4A	B4A_N0	PIN_AE16	3.3-V LVTTL		16mA (default)	
quot[3]	Output	PIN_W16	4A	B4A_N0	PIN_W16	3.3-V LVTTL		16mA (default)	1 (default)
quot[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	3.3-V LVTTL		16mA (default)	1 (default)
quot[1]	Output	PIN_V18	4A	B4A_N0	PIN_V18	3.3-V LVTTL		16mA (default)	1 (default)
quot[0]	Output	PIN_W17	4A	B4A_N0	PIN_W17	3.3-V LVTTL		16mA (default)	1 (default)
rem[3]	Output	PIN_W19	4A	B4A_N0	PIN_W19	3.3-V LVTTL		16mA (default)	1 (default)
rem[2]	Output	PIN_Y19	4A	B4A_N0	PIN_Y19	3.3-V LVTTL		16mA (default)	1 (default)
rem[1]	Output	PIN_W20	5A	B5A_N0	PIN_W20	3.3-V LVTTL		16mA (default)	1 (default)
rem[0]	Output	PIN_W21	5A	B5A_N0	PIN_W21	3.3-V LVTTL		16mA (default)	1 (default)
rst	Input	PIN_AF16	4A	B4A_N0	PIN_AF16	3.3-V LVTTL		16mA (default)	
start	Input	PIN_AF15	3B	B3B_N0	PIN_AF15	3.3-V LVTTL		16mA (default)	
valid	Output	PIN_Y21	5A	B5A_N0	PIN_Y21	3.3-V LVTTL		16mA (default)	1 (default)
<<new node>>									

Fig. 8(c) - Pin planning of slow division algorithm

Conclusion

This report provides a comprehensive overview of the slow and fast division algorithms that were explored in this project. The flowcharts and input/output table provide a visual representation of how the algorithms work, and the example problems demonstrate how the algorithms can be used to divide two numbers. The validation methods for the slow and fast division algorithms provide a way to verify that the algorithms are producing the correct results.

Reference

1. Xilinx. (2015). 7 Series FPGAs SelectIO Resources User Guide, ug471 (v1.6) ed.
2. Microchip. (2006). PIC24FJ128GA Family Data Sheet, ds39747c ed.
3. Cummins, C. E. (2000). Nonblocking assignments in Verilog synthesis, coding styles that kill!, in SNUG 2000 San Jose.
4. Hamid, M. (2010). Writing Efficient Testbenches, Xilinx, xapp199 (v1.1) ed.
5. Digital system design with FPGA - Implementation using Verilog and VHDL by Cem Ünsalan and Bora Tar.