FPGA Implementation Documentation – ALU & Memory

Abstract

This work presents the intermediate implementation of two digital design modules - a memory block and an Arithmetic Logic Unit (ALU) on the **Basys 3Artix-7 FPGA** using Vivado.

The memory module was designed to store and retrieve 16-bit data words at specified addresses, with initialization and basic read/write operations verified through simulation as well as implementation on the FPGA.

The ALU module was implemented to perform a range of arithmetic and logical operations on two 16-bit operands, with status flags (zr, ng) displayed via the on-board seven-segment display.

Both modules were successfully synthesized, mapped to LUTs, and verified through testbenches. Preliminary quality of results (QoR) analysis shows efficient utilization of FPGA resources, no congestion, and all timing constraints satisfied, confirming the correctness and feasibility of the approach for further integration and testing.

ALU Implementation

The Arithmetic Logic Unit (ALU) was implemented as a parameterized 16-bit module capable of performing a wide range of arithmetic and logical operations based on six control inputs (zx, nx, zy, ny, f, no). Two 16-bit operands are selected from pre-initialized memory locations using 5-bit address inputs, allowing flexible testing across positive, negative, and patterned data values. Depending on the opcode settings, the ALU supports operations such as zeroing inputs, bitwise negation, addition, subtraction, and logical AND, along with conditional negation of the output.

To provide status feedback, the ALU generates two condition flags: zero (zr), which is asserted when the result equals zero, and negative (ng), which indicates a negative output in two's complement form. These flags are also mapped to the on-board seven-segment display, offering real-time visualization during FPGA testing. The ALU design was simulated extensively using binary test vectors, then synthesized and mapped successfully to the FPGA with correct functional verification.

The ALU was implemented using a **hardcoded register initialization strategy**. Instead of directly providing operands, two memory arrays were defined with 16-bit values preloaded at specific addresses. During execution, the ALU receives only the operand addresses, retrieves the corresponding data, and performs the required computation. This approach streamlined operand handling and enabled testing across a wide set of positive, negative, and patterned values.

Block Diagram

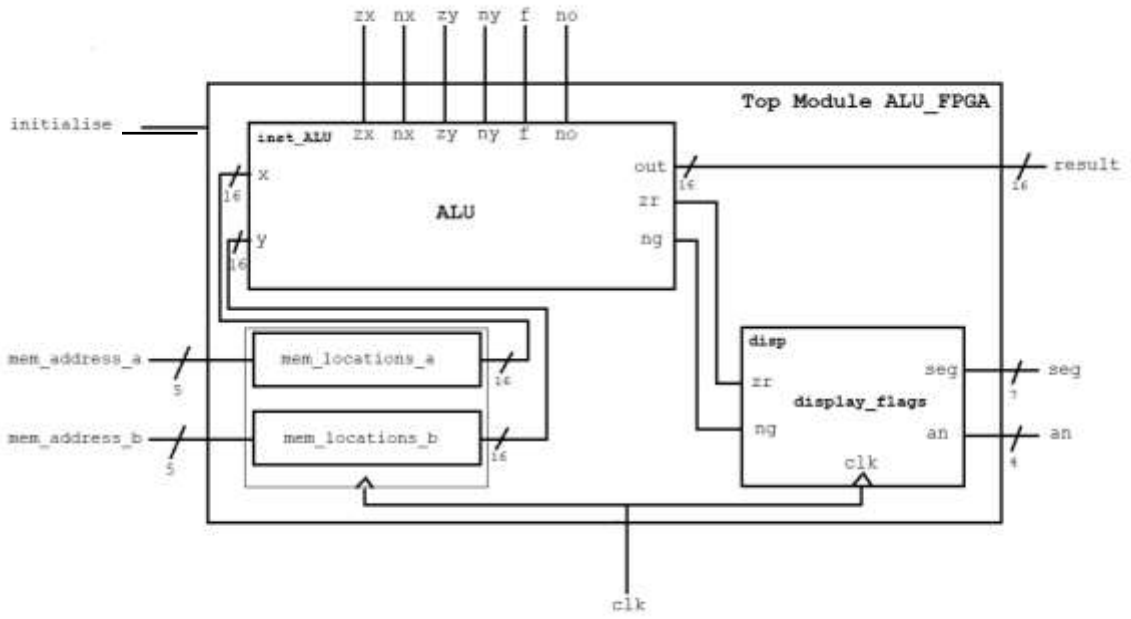


Figure 1: Block diagram of the ALU

The top-level design integrates an ALU with memory arrays and a display unit. Two 5-bit address inputs (mem_address_a and mem_address_b) select operands from internal memory blocks, which are then fed into the ALU along with control signals. The ALU computes the result and generates status flags (zr, ng), which are passed to the display module for visualization on the seven-segment display (seg, an).

IO Table

Direction	Name	Width	Comments
Input	clk	1	Clock input to the registers
Input	initialise	1	Initializes the registers with the values
Input	mem_address_a	5	Address of the operand ‘a’
Input	mem_address_b	5	Address of the operand ‘b’
Input	zx	1	Zeros input x
Input	nx	1	Negates input x
Input	zy	1	Zeros input y
Input	ny	1	Negates input y
Input	f	1	To choose between AND and Add operations
Input	no	1	Negates the output
Output	result	16	Result of the ALU operation
Output	seg	7	To interface the Flags onto the 7 Segment display
Output	an	4	To choose display digit

Quality of Result

Congestion

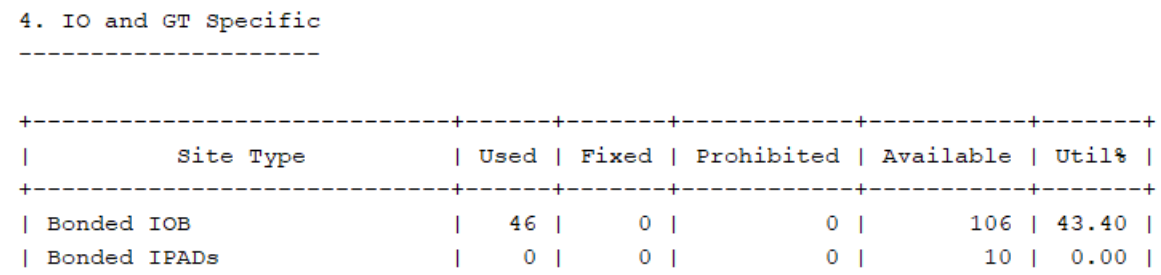
The congestion analysis shows no congestion windows above level 5, indicating that the design is well placed and routed without routing bottlenecks. This confirms efficient logic distribution across the FPGA fabric.

```
| Command      : report_design_analysis -congestion
| Design       : ALU_FPGA
| Device       : xc7a35t
| Design State : Routed

Report Design Analysis

Table of Contents
-----------------
1. Placer Final Level Congestion Reporting
2. Initial Estimated Router Congestion Reporting
3. SLR Net Crossing Reporting

1. Placer Final Level Congestion Reporting
------------------------------------------

+-----------+------+-------+------------+--------+-----------------+---------------+-----+---------+------+-----+-----+-------+-----+-------------+
| Direction | Type | Level | Congestion | Window | Combined LUTs | Avg LUT Input | LUT | LUTRAM | Flop | MUXF | RAMB | DSP | CARRY | SRL | Cell Names |
+-----------+------+-------+------------+--------+-----------------+---------------+-----+---------+------+-----+-----+-------+-----+-------------+
* No congestion windows are found above level 5


2. Initial Estimated Router Congestion Reporting
------------------------------------------------

+-----------+------+-------+-----------------+--------+-----------------+---------------+-----+---------+------+-----+-----+-------+-----+-------------+
| Direction | Type | Level | Percentage Tiles | Window | Combined LUTs | Avg LUT Input | LUT | LUTRAM | Flop | MUXF | RAMB | DSP | CARRY | SRL | Cell Names |
+-----------+------+-------+-----------------+--------+-----------------+---------------+-----+---------+------+-----+-----+-------+-----+-------------+
* No initial estimated congestion windows are found above level 5
```

Image 1: Congestion Report

Resource Utilization

The utilization summary shows the number of LUTs, flip-flops, IO blocks consumed by the design. The values are well within the available resources of the Basys 3 Artix 7 board, confirming that the design fits comfortably on the FPGA with margin for further expansion.



```
4. IO and GT Specific
---------------------

+-------------------------------+------+-------+------------+-----------+-------+
|           Site Type           | Used | Fixed | Prohibited | Available | Util% |
+-------------------------------+------+-------+------------+-----------+-------+
| Bonded IOB                    |   46 |     0 |          0 |       106 | 43.40 |
| Bonded IPADs                  |    0 |     0 |          0 |        10 |  0.00 |
```

Image 2: IO Blocks Utilisation Summary

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	79	0	0	20800	0.38
LUT as Logic	79	0	0	20800	0.38
LUT as Memory	0	0	0	9600	0.00
Slice Registers	4	0	0	41600	<0.01
Register as Flip Flop	4	0	0	41600	<0.01

Image 3: LUT, Register & Flipflop usage

7. Primitives

Ref Name	Used	Functional Category
LUT6	48	LUT
LUT2	31	LUT
OBUF	29	IO
IBUF	17	IO
LUT3	16	LUT
LUT4	5	LUT
CARRY4	4	CarryLogic
FDRE	3	Flop & Latch
LUT5	2	LUT
LUT1	1	LUT
FDSE	1	Flop & Latch
BUFG	1	Clock

Image 4: List of all the primitives used

Timing Summary

The timing report indicates that all paths meet the required constraints, with positive slack values. This means the design achieves timing closure and can operate reliably at the specified clock frequency.

Image 5: Timing Summary

Screenshots

Elaborated Circuit Schematic

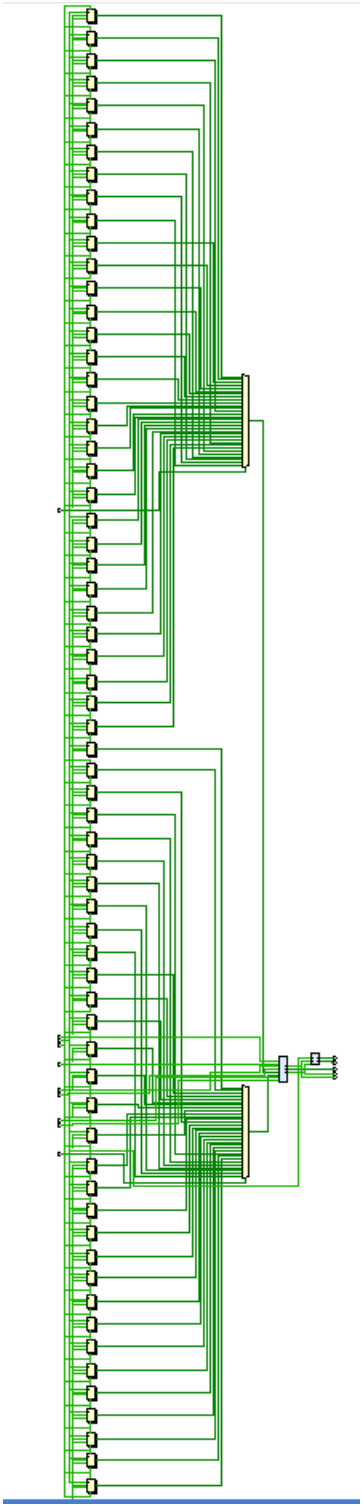


Image 6: Entire elaborated Circuit

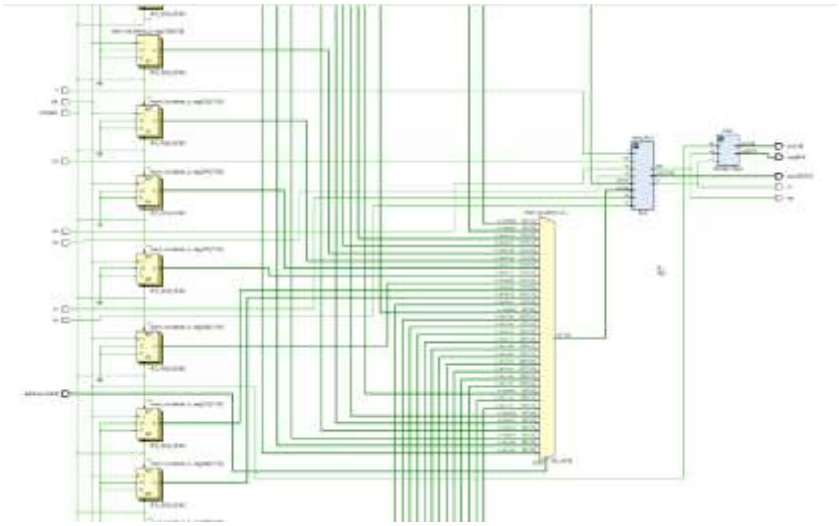


Image 7: View of the register selecting mux along with few registers, ALU and Display interface module instantiations

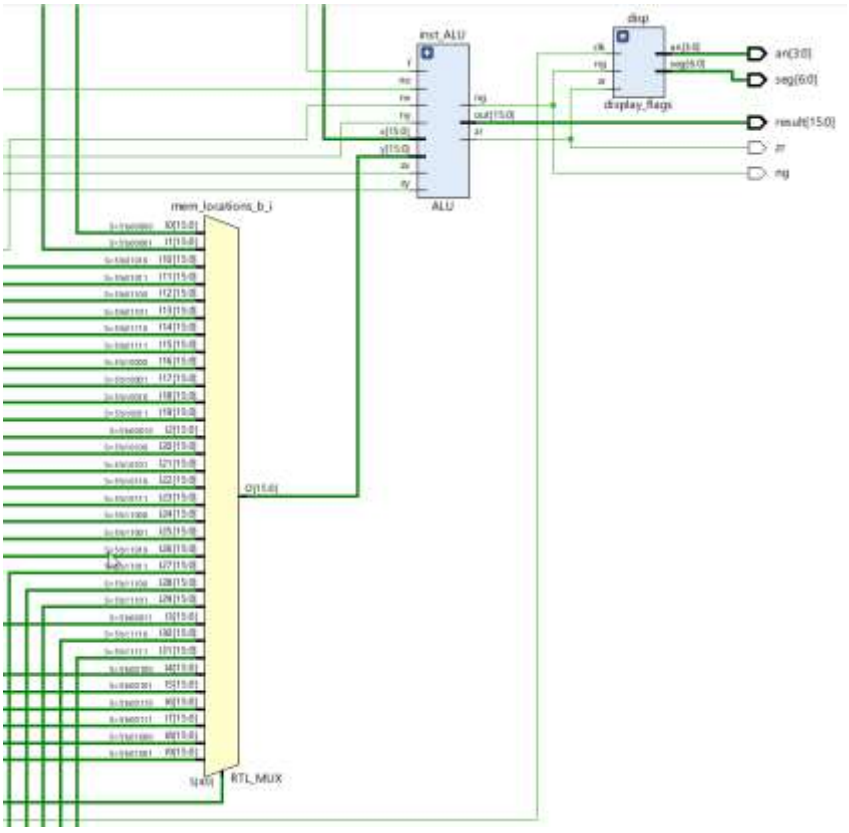


Image 8: View of the register selecting mux, ALU and display_flag module instantiations and outputs

Mapping Diagram to LUTs (Synthesized design)

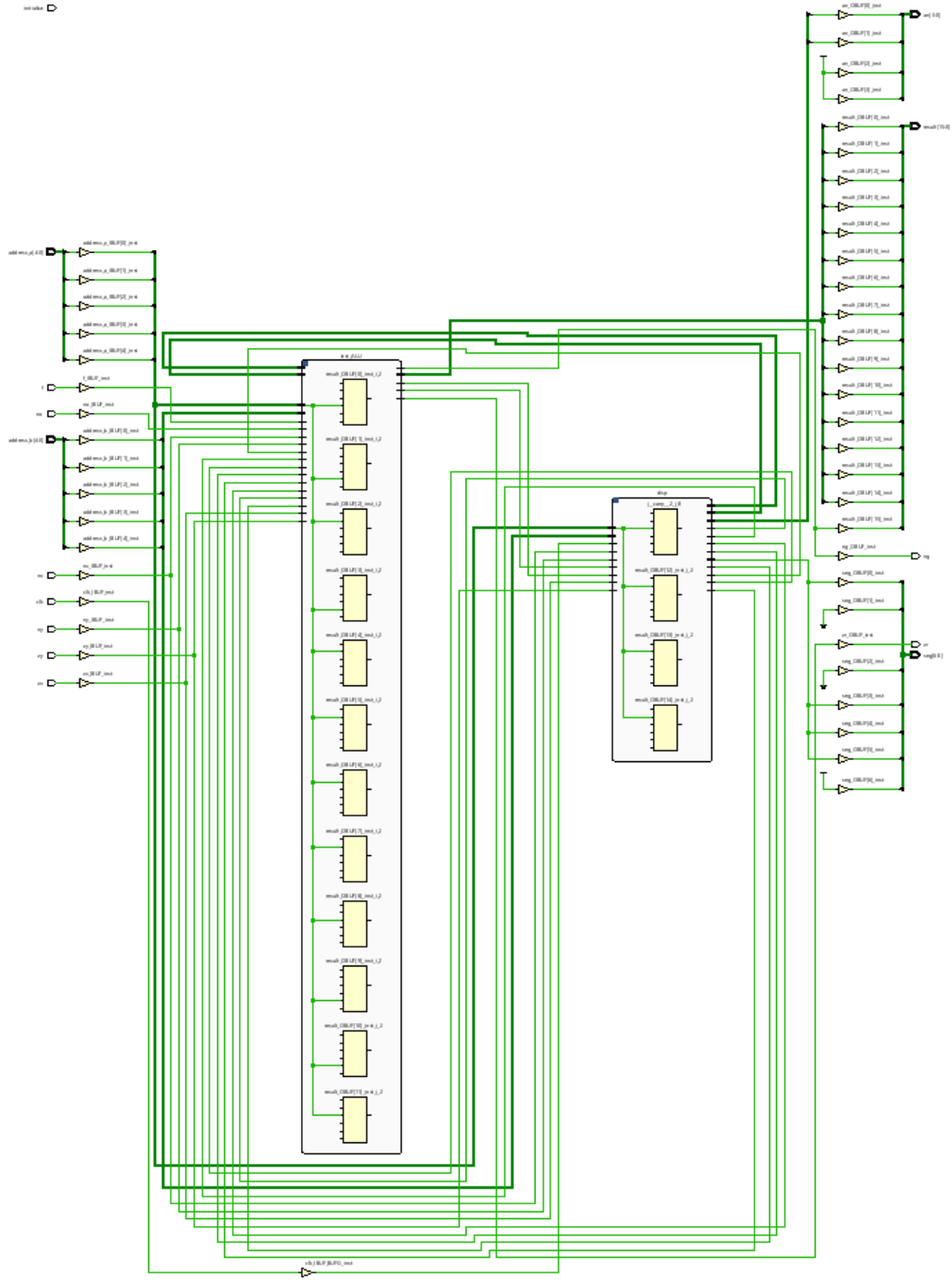


Image 9: Diagram showing the design being mapped to 6-input LUTs (ALU and display_flags module instantiations)

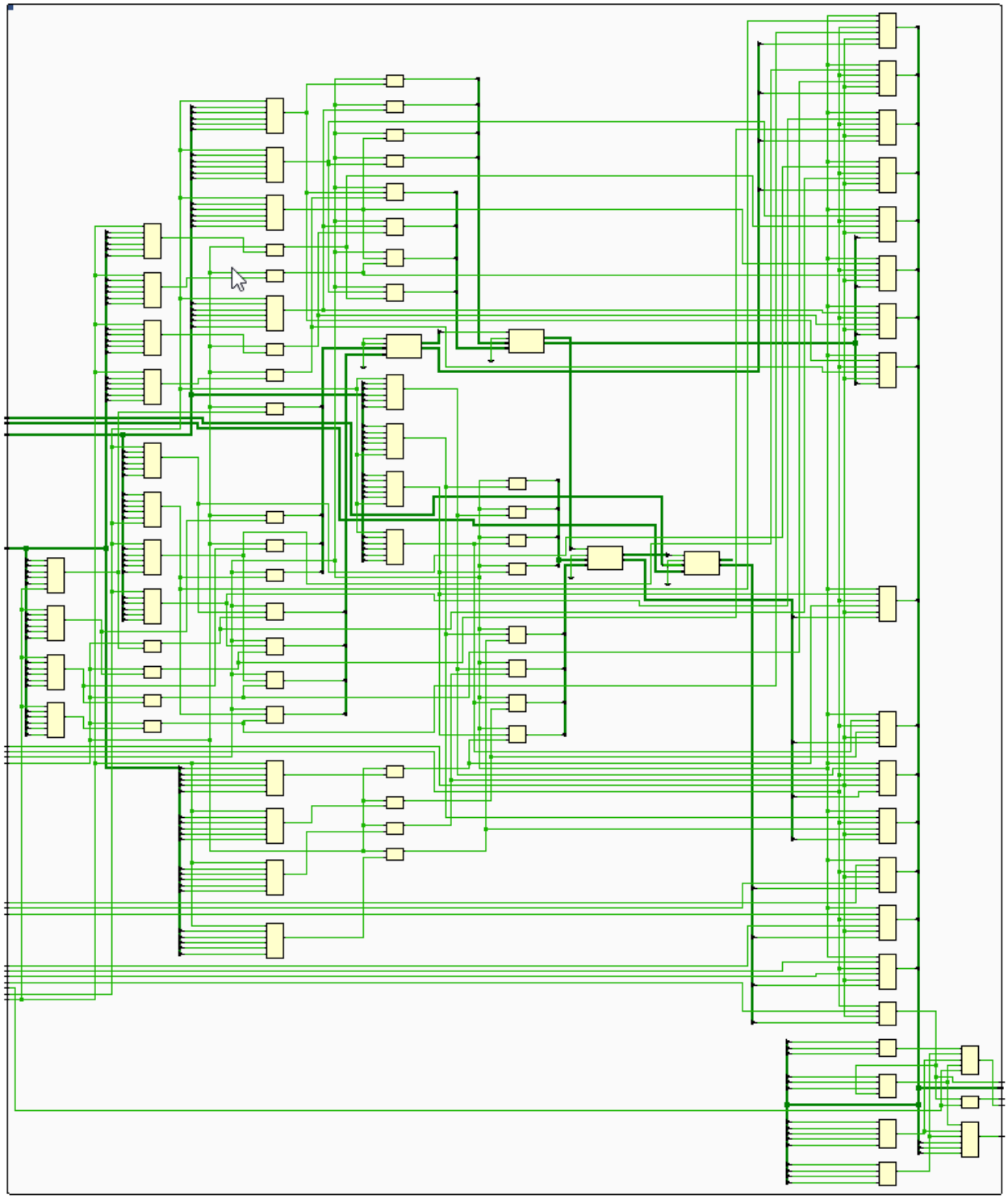


Image 10: Diagram showing the ALU module instance being mapped onto LUTs and flipflops

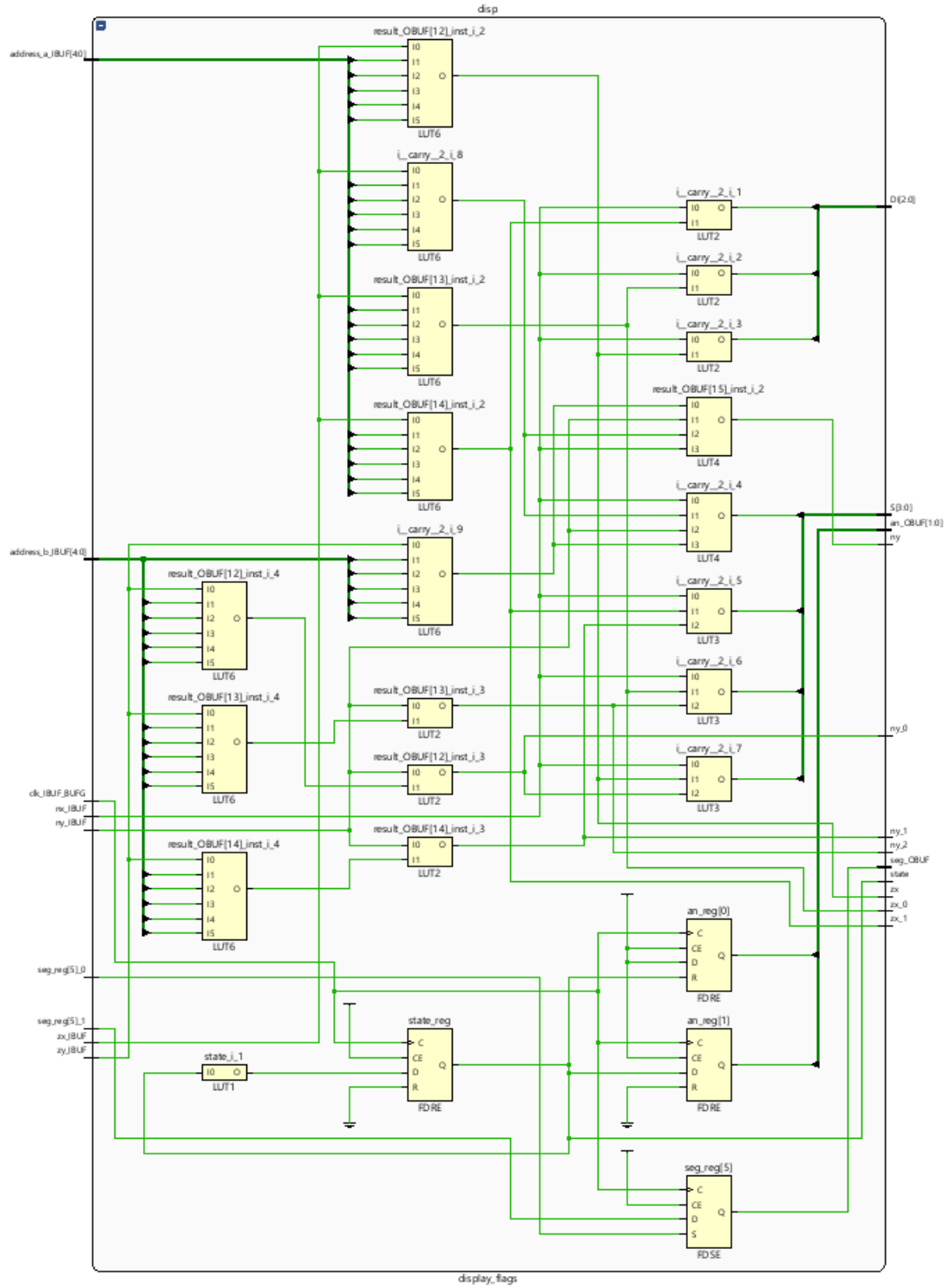


Image 11: Diagram showing the display_flags module instance being mapped onto LUTs and flipflops

Constraints

XDC file: Xilinx Design Constraint File for ALU Implementation

The design uses a Xilinx Design Constraint (XDC) file to specify pin assignments for the Basys 3 Artix-7 FPGA board. The file maps all the top-level input and output signals to the appropriate physical pins, including the on-board 100 MHz clock source that drives the design, pushbuttons, switches, LEDs, and the seven-segment display. The constraints ensure correct interfacing with the board peripherals. At this stage, **no additional timing, power, or design constraints** have been specified apart from the functional pin mappings.

Verification Strategy and Proof of Correctness

The functionality of the ALU design was verified using a two-fold approach. First, a comprehensive Verilog testbench was developed to simulate various input combinations, including arithmetic, logical, and edge-case operations. The simulation results were compared against the expected outputs (e.g., zero flag, negative flag, and computed results) to confirm correctness. Second, the design was synthesized, implemented, and deployed on the FPGA board, where the outputs were observed directly through the LED indicators and seven-segment display. This combination of simulation and hardware testing ensured that the ALU behaved as intended under both controlled testbench conditions and real hardware execution.

Testbench file: Testbench file for ALU implementation

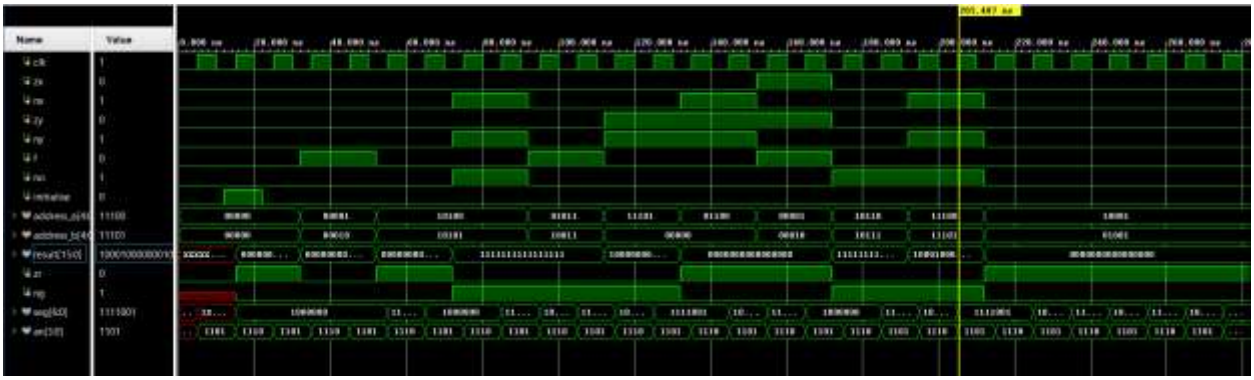


Image 12: Behavioral Simulation of ALU_FPGA module using a testbench

FPGA Implementation video: ALU Implementation video

Design Files:

1. Top Module
2. ALU module
3. Segmented Display Interface Module

Memory Implementation

The memory module was implemented using a **register array** in Verilog, declared as reg [15:0] RAM16K_SCREEN [0:1023], which provides a storage capacity of 1024 words, each 16 bits wide. The design allows both reading and writing operations.

Values are written into memory by latching an address from the input switches using a pushbutton and then storing the 16-bit input data into the corresponding memory location on the rising edge of the clock when the write control signal is asserted. Reading is achieved by continuously driving the output with the contents of the currently latched address, ensuring that the output always reflects the data stored at the selected memory location.

Block Diagram

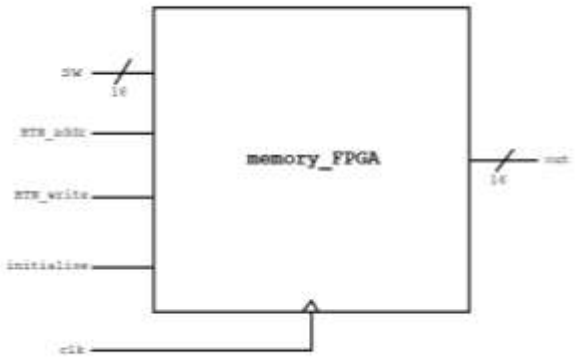


Figure 2: Block Diagram of Memory

The memory_FPGA module implements a 16-bit wide memory with read/write capability. The 16 switches (SW[15:0]) serve as data or address inputs based on control. The BTN_addr button latches the switch value as the memory address, while BTN_write writes the switch data into the selected location. The initialise signal can preload or reset memory contents. On each clock (clk), the chosen memory word is read and driven onto the 16-bit out port.

IO Table

Direction	Name	Width	Comments
Input	clk	1	Clock input to the registers
Input	BTN_addr	1	Latches switch input as the memory address
Input	BTN_write	1	Latches switch input as the value input
Input	initialize	1	Initializes the registers with the values
Input	sw	16	Provides data or address via switches
Output	out	16	Outputs data read from memory

Quality of Result

Congestion

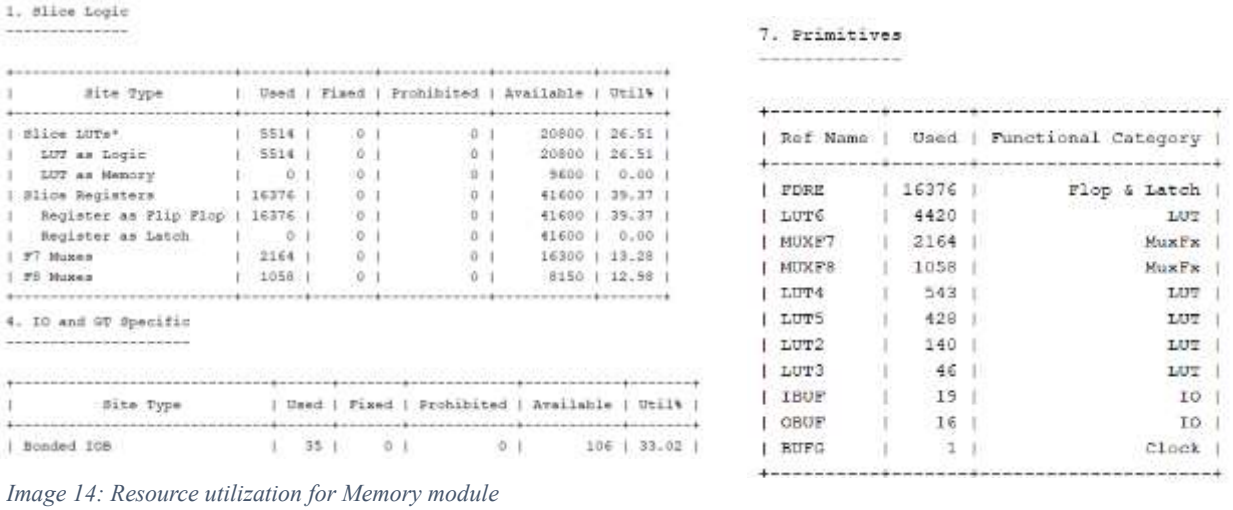
The congestion analysis for the memory module shows no significant routing congestion within the FPGA fabric. Since the design primarily consists of a memory array with simple control logic (address latch, write enable, and output register), the resource utilization is low, and routing demand is minimal.

```
| Design       : memory_FPGA
| Device       : xc7a35t
| Design State : Routed

Report Design Analysis

Table of Contents
-----------------
1. Placer Final Level Congestion Reporting
2. Initial Estimated Router Congestion Reporting
3. SLR Net Crossing Reporting

1. Placer Final Level Congestion Reporting
------------------------------------------

+-----------+------+-------+------------+--------+----------------+----------------+-----+---------+------+------+-----+-------+-----+------------+
| Direction | Type | Level | Congestion | Window | Combined LUTs | Avg LUT Input | LUT | LUTRAM | Flop | MUXF | RAMB | DSP | CARRY | SRL | Cell Names |
+-----------+------+-------+------------+--------+----------------+----------------+-----+---------+------+------+-----+-------+-----+------------+
* No congestion windows are found above level 5

2. Initial Estimated Router Congestion Reporting
------------------------------------------------

+-----------+------+-------+-----------------+--------+----------------+----------------+-----+---------+------+------+-----+-------+-----+------------+
| Direction | Type | Level | Percentage Tiles | Window | Combined LUTs | Avg LUT Input | LUT | LUTRAM | Flop | MUXF | RAMB | DSP | CARRY | SRL | Cell Names |
+-----------+------+-------+-----------------+--------+----------------+----------------+-----+---------+------+------+-----+-------+-----+------------+
* No initial estimated congestion windows are found above level 5
```

Image 13: Congestion report of Memory module

Resource Utilisation

The memory design shows moderate utilization of LUTs and registers, with about one-quarter of LUTs and less than half of the available registers in use. The absence of LUTs configured as memory indicates that the design relies entirely on flip-flops for storage rather than distributed RAM. The use of F7 and F8 multiplexers suggests combinational logic is being optimized for wider functions. Overall, the design is resource-efficient, fits comfortably within the FPGA capacity, and leaves enough margin for additional modules or expansion.



```
1. Slice Logic
--------------

+-------------------------+-------+-------+------------+-----------+-------+
|        Site Type        | Used  | Fixed | Prohibited | Available | Util% |
+-------------------------+-------+-------+------------+-----------+-------+
| Slice LUTs*             |  5514 |     0 |          0 |     20800 | 26.51 |
|   LUT as Logic          |  5514 |     0 |          0 |     20800 | 26.51 |
|   LUT as Memory         |     0 |     0 |          0 |      9600 |  0.00 |
| Slice Registers         | 16376 |     0 |          0 |     41600 | 39.37 |
|   Register as Flip Flop | 16376 |     0 |          0 |     41600 | 39.37 |
|   Register as Latch     |     0 |     0 |          0 |     41600 |  0.00 |
| F7 Muxes                |  2164 |     0 |          0 |     16300 | 13.28 |
| F8 Muxes                |  1058 |     0 |          0 |      8150 | 12.98 |
+-------------------------+-------+-------+------------+-----------+-------+

4. IO and GT Specific
---------------------

+----------------------------+------+-------+------------+-----------+-------+
|         Site Type          | Used | Fixed | Prohibited | Available | Util% |
+----------------------------+------+-------+------------+-----------+-------+
| Bonded IOB                 |   35 |     0 |          0 |       106 | 33.02 |

7. Primitives
-------------

+----------+-------+-------------------------+
| Ref Name |  Used | Functional Category |
+----------+-------+-------------------------+
| FDRE     | 16376 |         Flop & Latch |
| LUT6     |  4420 |                  LUT |
| MUXF7    |  2164 |                MuxFx |
| MUXF8    |  1058 |                MuxFx |
| LUT4     |   543 |                  LUT |
| LUT5     |   428 |                  LUT |
| LUT2     |   140 |                  LUT |
| LUT3     |    46 |                  LUT |
| IBUF     |    19 |                   IO |
| OBUF     |    16 |                   IO |
| BUFG     |     1 |                Clock |
+----------+-------+-------------------------+
```

Image 14: Resource utilization for Memory module

Timing Summary

The Design Timing Summary shows that all user-specified timing constraints are met. The setup slack (0.945 ns), hold slack (0.388 ns), and pulse width slack (4.500 ns) are positive, indicating there are no timing violations and the design is stable for implementation.

Image 15: Design Timing Summary for Memory module

Screenshots

Mapping Diagram to LUTs (Synthesized design)

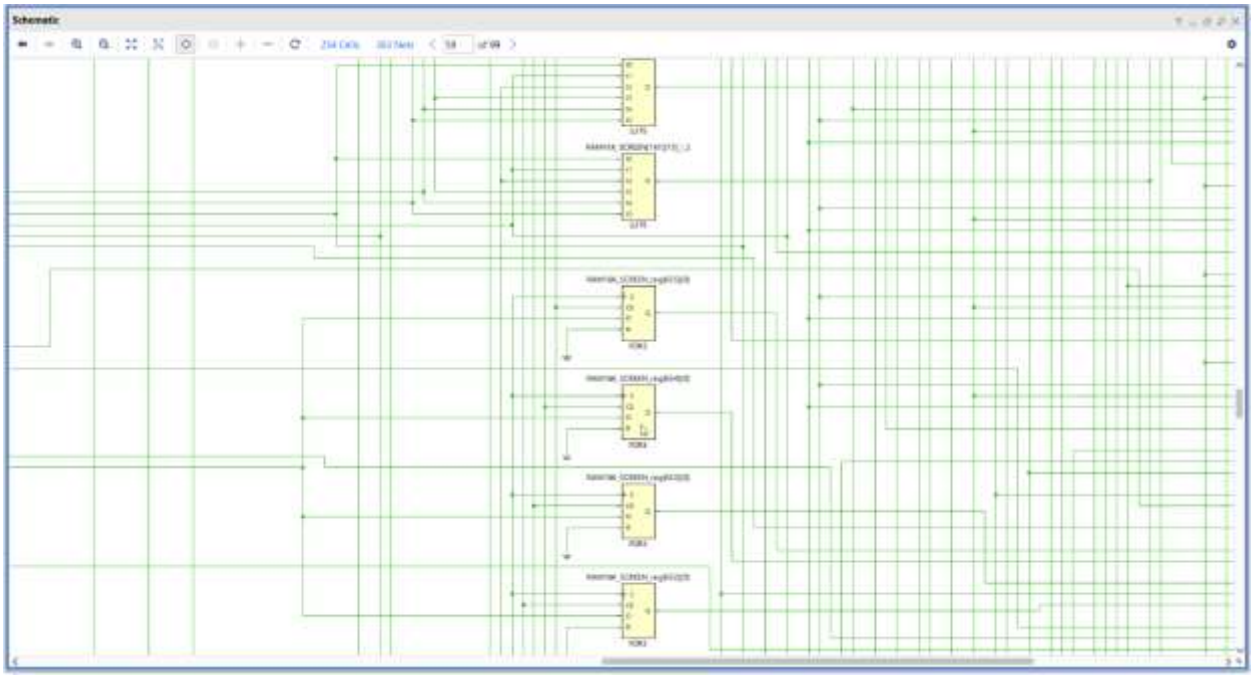


Image 16: Part of the LUT mapping diagram of the Memory module

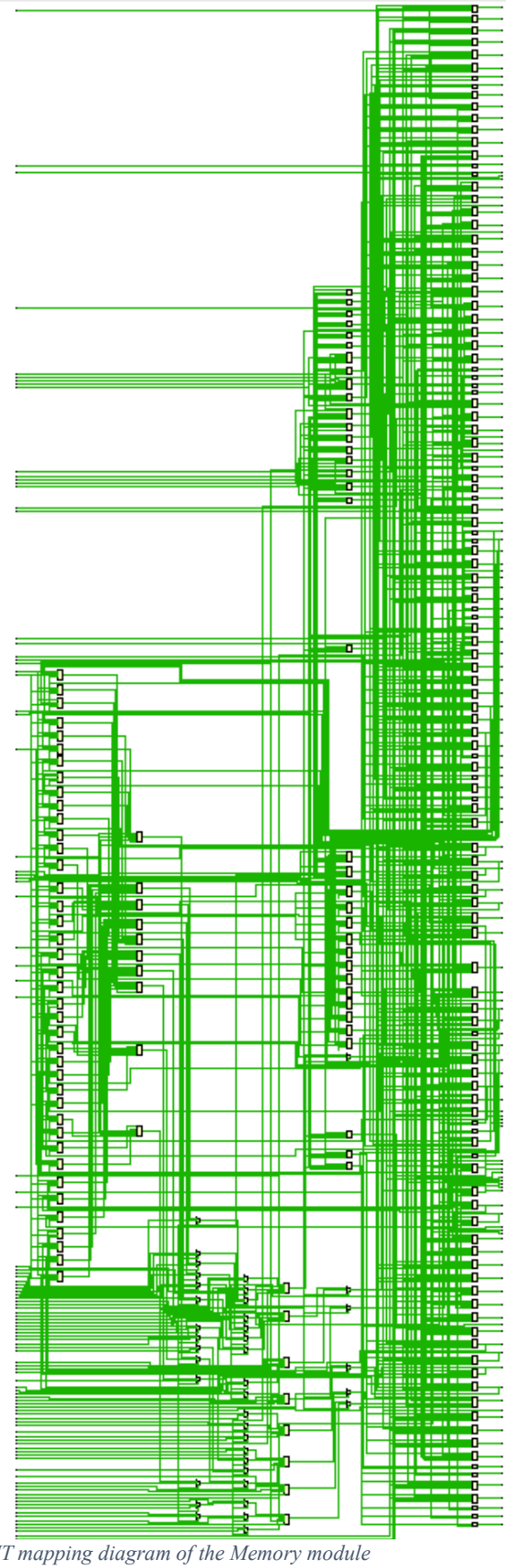


Image 17: LUT mapping diagram of the Memory module

Elaborated Circuit Schematic

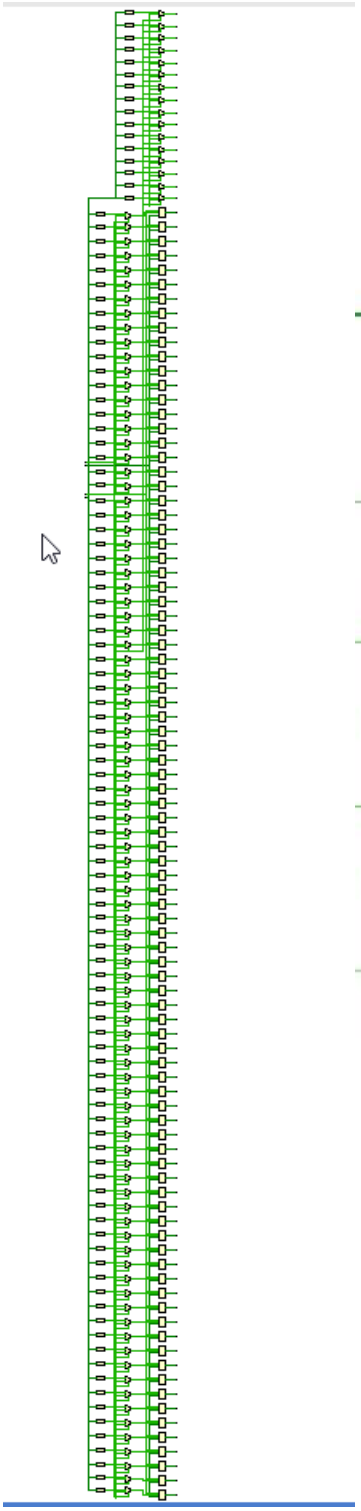


Image 18: Entire Elaborated circuit for Memory Module

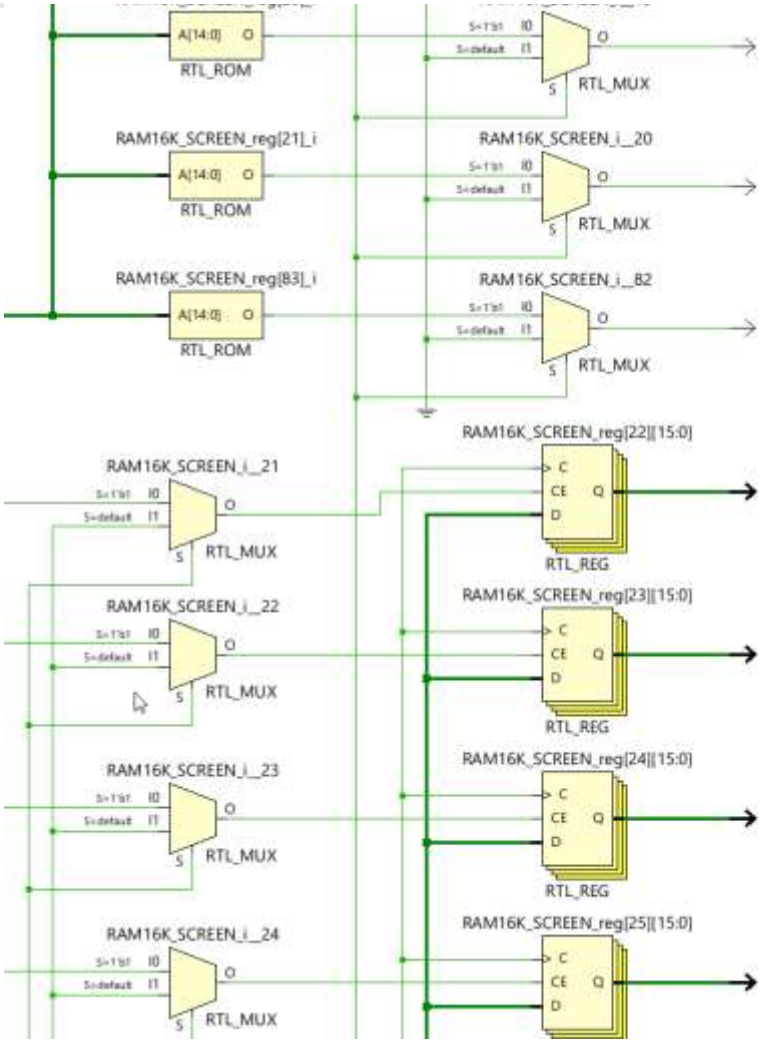


Image 19: Part of Elaborated circuit for Memory Module

Constraints

XDC File [Xilinx Design Constraint File from Memory Implementation]

The constraint file for the memory module specifies the mapping of FPGA pins to the inputs and outputs of the design. The on-board clock is used as the primary timing source to drive the memory operations. Switches are mapped to provide address and data inputs, while pushbuttons are assigned for latching the address and triggering write operations. The output of the memory is mapped to the FPGA LEDs for verification. At this stage, **no additional power or timing constraints** have been applied, and only essential pin assignments required for correct operation on the FPGA board are included.

Verification Strategy and Proof of Correctness

The memory design was verified using a two-step approach. First, a simulation testbench was developed to apply various binary inputs, check address latching, and confirm correct read/write functionality across multiple test cases. Second, the design was implemented on the FPGA, where switches and pushbuttons were used to provide inputs, and the stored values were observed on the output LEDs. This ensured functional correctness in both simulation and hardware.

Testbench file: [Testbench File for Memory Implementation]

Image 20: Behavioral Simulation Result of Memory Implementation

FPGA Implementation video: [Memory Implementation Video]

Design File: [Memory]