DESIGN DOCUMENTATION

ARITHMETIC LOGIC UNIT

Raghul G

EMP ID : 6097

C O N T E N T S

Introduction:

This project presents the design and implementation of a Digital Arithmetic and Logical Unit (ALU) as a comprehensive Design Under Test (DUT). The ALU is capable of executing a wide range of arithmetic and logical operations based on input command codes. Developed using Verilog, the design emphasizes reconfigurability, functional completeness, and precise control logic validation.

The primary objective of this ALU is to support 13 arithmetic and 14 logical operations while ensuring accurate and efficient performance. The architecture includes distinct modules for input handling, command decoding, operation execution, and output flag generation. It integrates clock-driven logic with configurable command-based multiplexers and internal registers, enabling sequential and pipelined processing.

A unique aspect of the design is its two-cycle delay for multiplication operations, while all other operations produce results in a single clock cycle. The ALU outputs status flags such as Carry Out (COUT), Overflow (OFLOW), Equality (EQ), Greater Than (GT), Less Than (LT), and Error (ERR), which aid in performance analysis and debugging.

Objective :

The objective of this project is to design and implement a reconfigurable Arithmetic and Logical Unit (ALU) capable of performing a wide range of arithmetic and logical operations. The ALU aims to,

- Support and demonstrate functional coverage of 13 arithmetic and 14 logical operations.
- Implement input control logic using clock (CLK), reset (RST), and control enable (CE) signals.
- Efficiently manage input data using internal registers and multiplexers.
- Generate appropriate output flags (COUT, OFLOW, EQ, GT, LT, ERR) to indicate operation results and system status.

- Ensure correct operation timing, with single-cycle execution for standard operations and two-cycle latency for multiplication.

Architecture:

The ALU architecture is modular and reconfigurable, designed to perform arithmetic and logical operations based on a control command. The design is composed of the following key components:

- **Input Section**: Accepts control and data inputs such as clock (CLK), reset (RST), control enable (CE), mode select (MODE), input valid (INP_VAL), command (CMD), operands (OPA and OPB), and carry-in (C_in).
- **Register Block**: Temporarily stores incoming signals in internal registers (t_MODE, t_INP_VAL, t_CMD, t_OPA, t_OPB, t_C_in) to synchronize operations with the clock.
- **Mode-Based MUX**: A 2x1 multiplexer selects either the Arithmetic Unit or Logical Unit based on the MODE signal.
- **Arithmetic Unit**: Contains a 4x1 multiplexer (for INP_VALID), an arithmetic command decoder, and a multiplier register (t_MUL) to handle complex operations like multiplication with a 2-cycle delay.
- **Logical Unit**: Includes a 4x1 multiplexer (for INP_VALID) and logic command decoder to perform various bitwise and shift/rotate operations.
- **Output Section**: Produces the result (RES) and sets status flags—Carry Out (COUT), Overflow (OFLOW), Equality (EQ), Greater Than (GT), Less Than (LT), and Error (ERR)—to indicate operation status and data validity.

Functions:

- **Arithmetic**
  - 0 : Addition
  - 1 : Subtraction
  - 2 : Addition with Carry
  - 3 : Subtraction with Carry
  - 4 : Increment A
  - 5 : Decrement A
  - 6 : Increment B
  - 7 : Decrement B
  - 8 : Comparison
  - 9 : (OPA + 1) multiplied with (OPB + 1)
  - 10 : (OPA << 1) multiplied with OPB
  - 11 : Signed Addition with Comparison
  - 12 : Signed Subtraction with Comparison

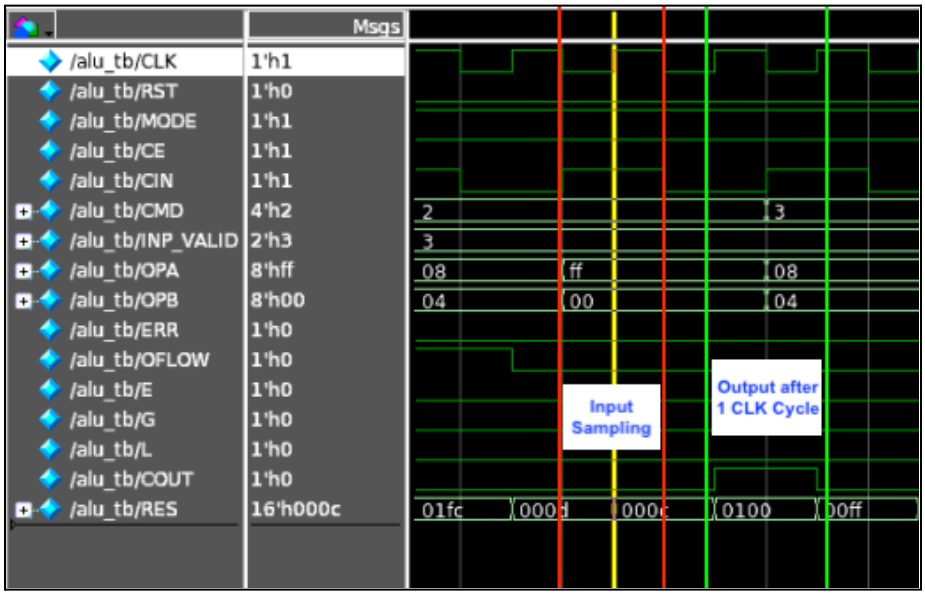
- **Logical**
  - 0: AND
  - 1: NAND
  - 2: OR
  - 3: NOR
  - 4: XOR
  - 5: XNOR
  - 6: NOT_A
  - 7: NOT_B
  - 8: Shift Right OPA by 1
  - 9: Shift Left OPA by 1
  - 10: Shift Right OPB by 1
  - 11: Shift Left OPB by 1
  - 12: Rotate Left OPA by OPB
  - 13: Rotate Right OPA by OPB

Result:

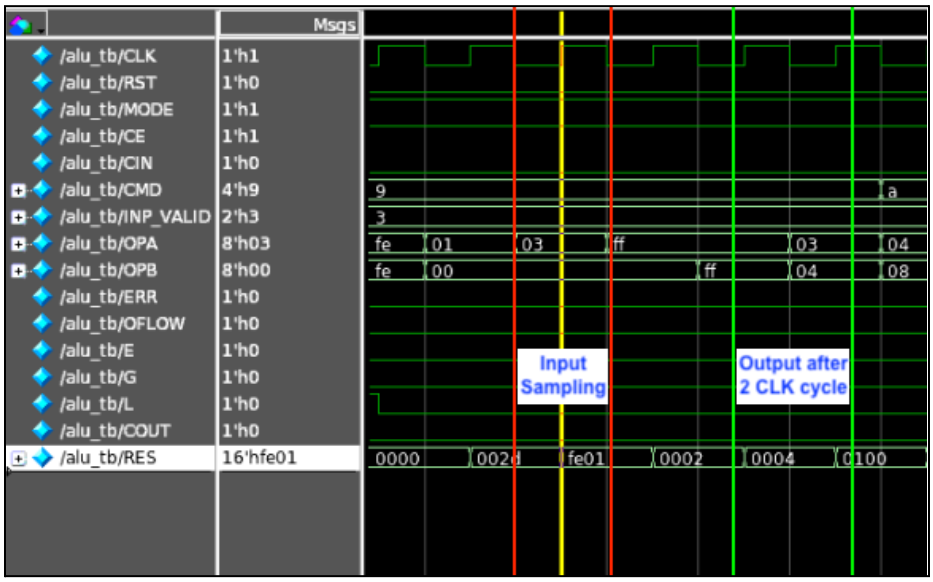
Upon simulation and testing, the DUT successfully:

- *Executes all 13 arithmetic operations including signed and shifted variations.*
- *Executes all 14 logical operations including bitwise, shift, and rotate.*
- *Correctly generates output flags and results.*

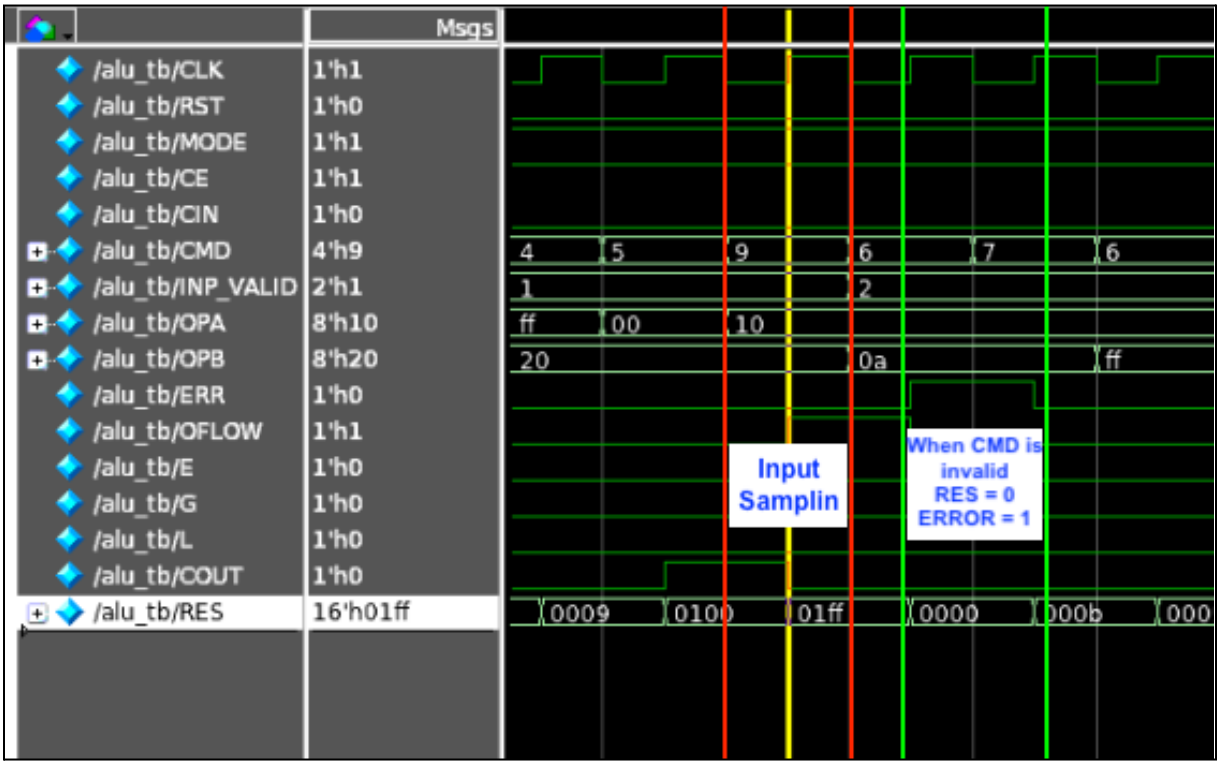
For General Operations - ADD_Cin: (1 - Clock Cycle Delay)



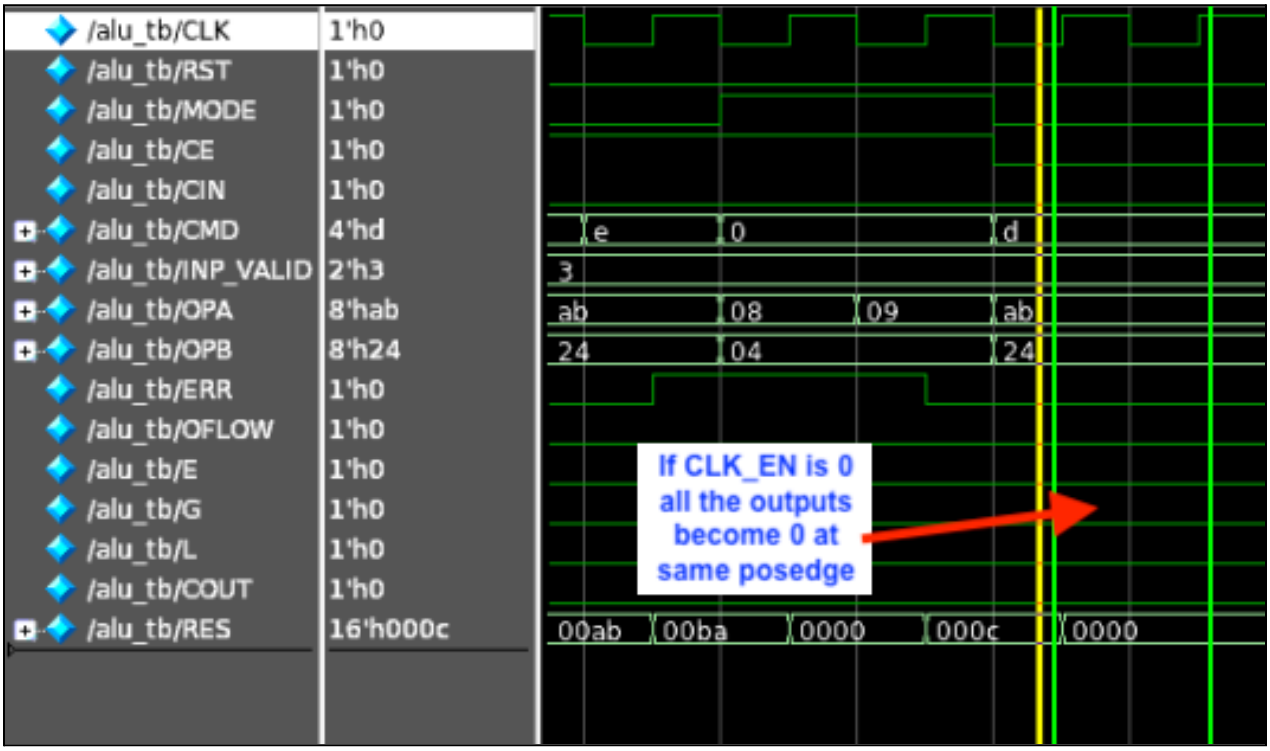
For Multiplication Operations - (A+1) x (B+1): (2 - Clock Cycle Delay)



For Error case - Invalid Command:



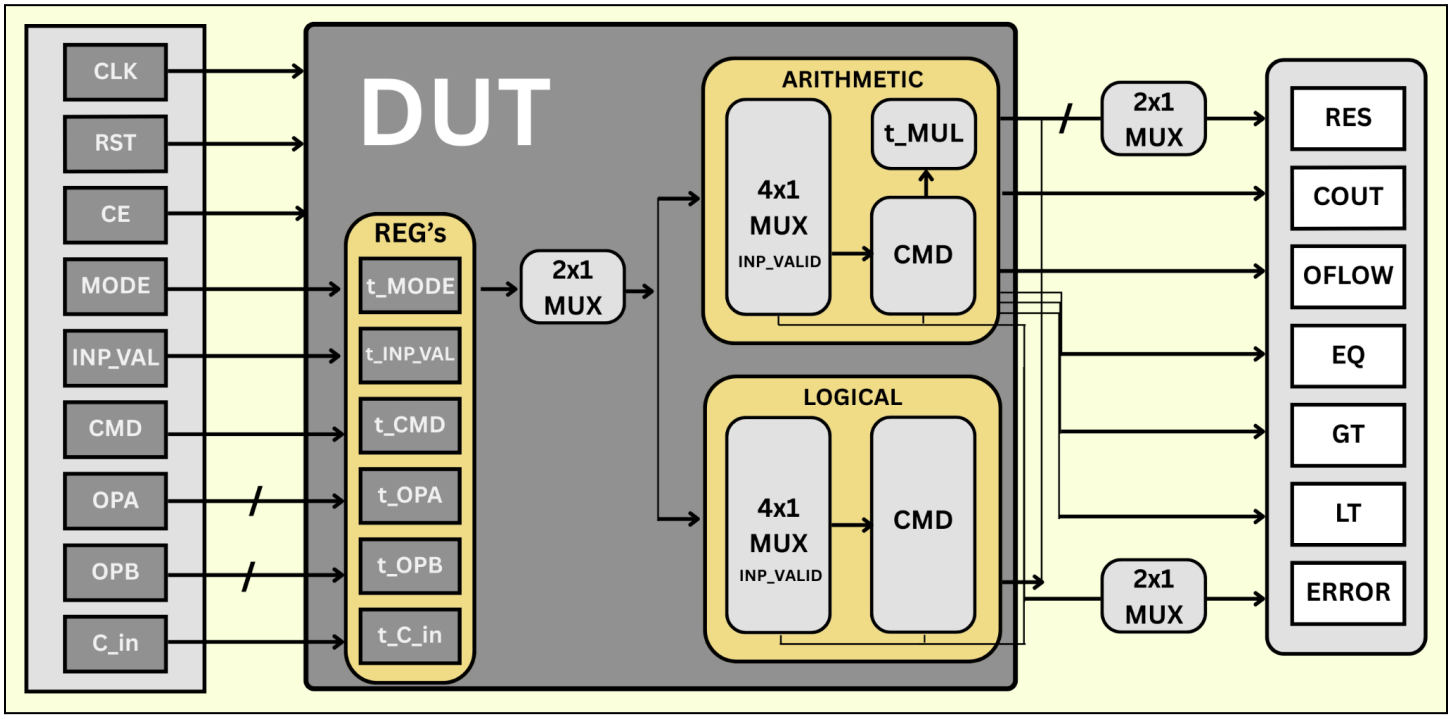
When CLK_EN = 0 - output comes in the same posedge:



Result.txt file

```
[[raghulganesan@FEserver ALU_PROJECT]$ cat results.txt
Feature ID   1  : PASS
Feature ID   2  : PASS
Feature ID   3  : PASS
Feature ID   4  : PASS
Feature ID   5  : PASS
Feature ID   6  : PASS
Feature ID   7  : PASS
Feature ID   8  : PASS
Feature ID   9  : PASS
Feature ID  10  : PASS
Feature ID  11  : PASS
Feature ID  12  : PASS
Feature ID  13  : PASS
Feature ID  14  : PASS
Feature ID  15  : PASS
Feature ID  16  : PASS
Feature ID  17  : PASS
Feature ID  18  : PASS
Feature ID  19  : PASS
Feature ID  20  : PASS
Feature ID  21  : PASS
Feature ID  22  : PASS
Feature ID  23  : PASS
Feature ID  24  : PASS
Feature ID  25  : PASS
Feature ID  26  : PASS
Feature ID  27  : PASS
Feature ID  28  : PASS
Feature ID  29  : PASS
Feature ID  30  : PASS
Feature ID  31  : PASS
Feature ID  32  : PASS
Feature ID  33  : PASS
Feature ID  34  : PASS
Feature ID  35  : PASS
Feature ID  36  : PASS
Feature ID  37  : PASS
Feature ID  38  : PASS
Feature ID  39  : PASS
Feature ID  40  : PASS
Feature ID  41  : PASS
```

DESIGN ARCHITECTURE:



DUT CODE COVERAGE:

Instance Path:
/alu_tb/uut
Design Unit Name:
work.alu_design
Language:
Verilog
Source File:
alu_tb.v

Local Instance Coverage Details:

Total Coverage: 100.00% **100.00%**

Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Statements	188	188	0	1	100.00%	**100.00%**
Branches	88	88	0	1	100.00%	**100.00%**
FEC Expressions	9	9	0	1	100.00%	**100.00%**
FEC Conditions	8	8	0	1	100.00%	**100.00%**
Toggles	222	222	0	1	100.00%	**100.00%**