Project ALU

Introduction

The project, arithmetic-logical unit (ALU) is a fundamental combinational unit in a digital circuits which performs different arithmetic and logical operations. The project is designed using Verilog focuses on parameterizing the inputs and outputs. The design various operations including addition, subtraction, multiplication, bitwise logics, shift of the input operands. The design ensures 2 clock cycle delay for the multiplication and 1 clock cycle delay for rest of the operations.

Objectives

- The design if flexible for the any data width, as it parameterized.
- ALU performs specific operations depending on the mode and commands which is selected.
- ALU is designed to generate the output with one clock cycle delay.
- Flags (OFLOW, COUT, ERR) are used to check the status of the obtained output.
- A testbench for the same is written in Verilog to check the proper function of the design.

Architecture

Design architecture

The below diagram represents the architecture of the ALU design.

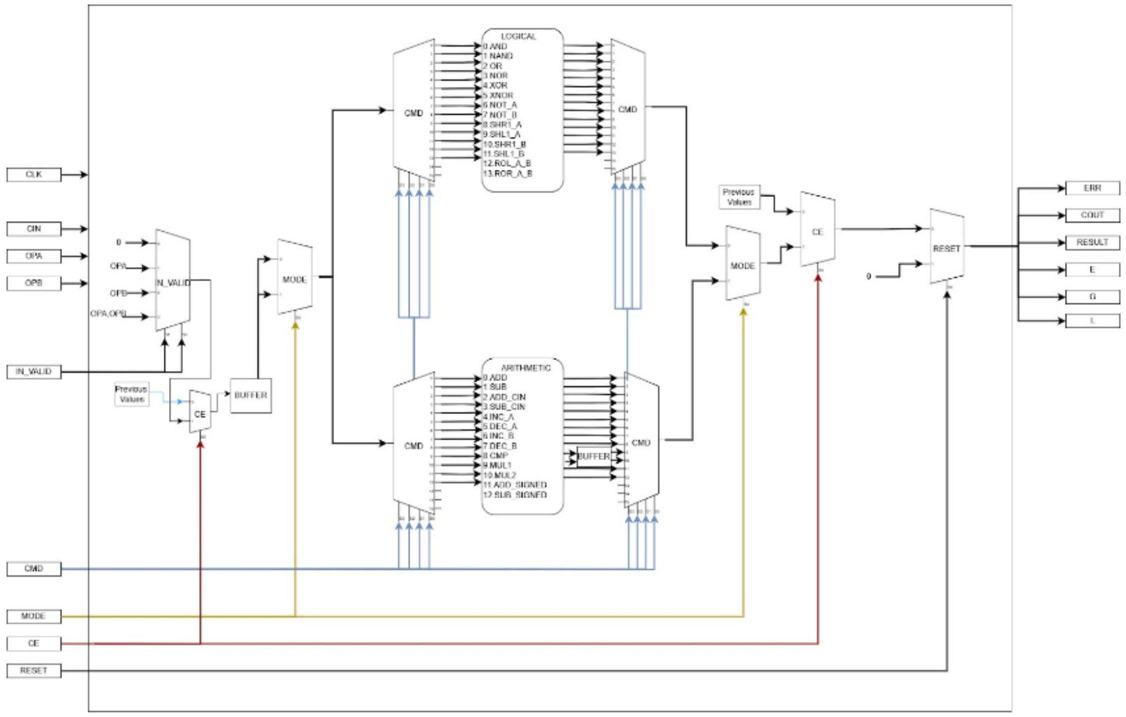


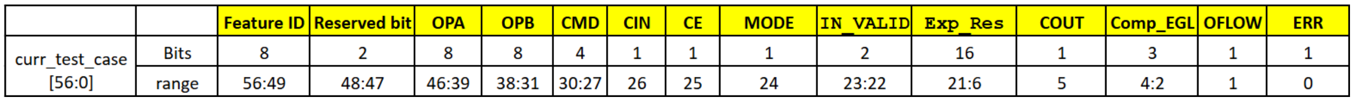
Fig.1: ALU Design Architecture

Pin description: The design has several input and output pins. Below table gives the description of the pins.

	Pin name	Description
INPUT PORTS	CLK	The clock signal on which the design perform function during positive edge.
	RESET	ALU is designed with the asynchronous active high reset signal
	CE	Clock enable is active high signal
	MODE	It is 1 bit signal. Arithmetic (MODE=1) or Logical (MODE=0) operations are performed, based on this signal
	IN_VALID	It is a 2-bit input valid signal to check the validity of the input operands
	CMD	CMD is the command input, which tells which operation has to be performed, depending on, in which MODE it is present
	OPA	Parameterized input operand
	OPB	Parameterized input operand
	CIN	1 bit input signal
OUTPUT PORTS	RESULT	Parameterized output
	COUT	1 bit carry-out signal used in addition/subtraction
	OFLOW	1 bit overflow signal used in addition/subtraction
	ERR	1 bit error signal, flag is raised to indicate errors
	G	1 bit output raised if OPA is greater than OPB
	L	1 bit output raised if OPA is lesser than OPB
	E	1 bit output raised if OPA and OPB are equal

Packet description

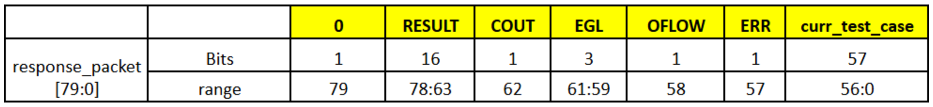
A text file, stimulus.txt contain the data, which includes expected outputs along with the feature_id and inputs, in the format given below.



		Feature ID	Reserved bit	OPA	OPB	CMD	CIN	CE	MODE	IN_VALID	Exp_Res	COUT	Comp_EGL	OFLOW	ERR
curr_test_case [56:0]	Bits	8	2	8	8	4	1	1	1	2	16	1	3	1	1
	range	56:49	48:47	46:39	38:31	30:27	26	25	24	23:22	21:6	5	4:2	1	0

Fig.2: Packet format of stimulus

Response packet contains the exact result obtained from DUT, along with the packet of stimulus.



		0	RESULT	COUT	EGL	OFLOW	ERR	curr_test_case
response_packet [79:0]	Bits	1	16	1	3	1	1	57
	range	79	78:63	62	61:59	58	57	56:0

Fig.3: Packet format of response packet

Testbench architecture

The below diagram represents the architecture of the testbench.

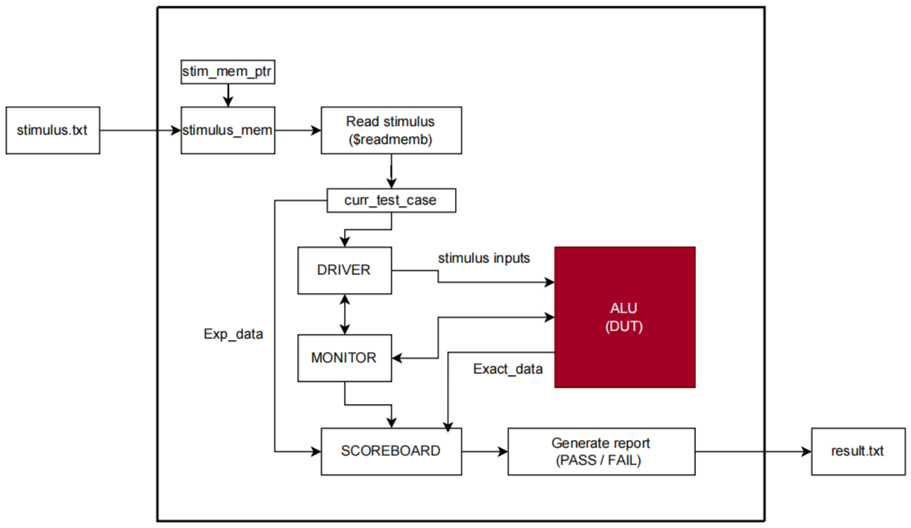


Fig.4: Testbench architecture

The testbench architecture contains blocks such as driver, monitor, scoreboard to check display the data.

- stimulus.txt is a file containing the test case data in the format as mentioned in Fig.2.
- Each test cases is stored in stimulus_mem, and it is fetched accordingly.
- Single test case (curr_test_case) is driven by the driver and inputs are fed into the DUT.
- The obtained output from DUT is sent to scoreboard.
- Based on the expected data (Exp_data) and obtained data (Exact_data), the report is generated, and for each feature id, the report is updated in result.txt file.
- Monitor monitors the data and prints it.

Working

Designed ALU will perform the operations and produce the outputs at the positive edge of the clock. ALU is designed with the asynchronous active high RESET and an active high clock enable. Whenever the RESET signal is HIGH(1), all the outputs are reset to 0. When the RESET signal is LOW(0), the preference is given to CE. When the CE signal is HIGH, the ALU performs the specified operation, and when it is LOW, ALU is latched to previous values.

MODE is a single bit signal, used to select in which mode the ALU has to perform the operation. When MODE is HIGH, then ALU is in arithmetic mode, when MODE is LOW, ALU is in logical mode. Then in particular mode, the operations are performed depending on the opcode (CMD).

IN_VALID is a 2-bit input valid signal which says which input operand (OPA and OPB) is valid. If IN_VALID is 0 (2’b00), no operand is valid. Input both OPA and OPB are not valid. If IN_VALID is 1 (2’b01), then OPA is valid. If IN_VALID is 2 (2’b10) only OPB is valid. If IN_VALID is 3 (2’b11).

CMD is the 4-bit opcode, which tell the ALU to perform the commands.

In MODE 1, i.e., arithmetic operation,

CMD	Operation	Description
0000	ADD	Addition of OPA and OPB
0001	SUB	Subtraction of OPB from OPA
0010	ADD_CIN	Addition of OPA and OPB with CIN
0011	SUB_CIN	Subtraction of OPB and CIN from OPA
0100	INC_A	Increment OPA by 1
0101	DEC_A	Decrement OPA by 1
0110	INC_B	Increment OPB by 1
0111	DEC_B	Decrement OPB by 1
1000	CMP	Compare OPA and OPB
1001	MUL1	Increment OPA and OPB and then multiply
1010	MUL2	Shift left OPA by 1 and multiply with OPB
1011	ADD_SIGNED	Signed addition of OPA and OPB
1100	SUB_SIGNED	Signed subtraction of OPA and OPB

In MODE 0, i.e., logical operation,

CMD	Operation	Description
0000	AND	Bitwise AND of OPA and OPB
0001	NAND	Bitwise NAND of OPA and OPB
0010	OR	Bitwise OR of OPA and OPB
0011	NOR	Bitwise NOR of OPA and OPB
0100	XOR	Bitwise XOR of OPA and OPB
0101	XNOR	Bitwise XNOR of OPA and OPB
0110	NOT_A	Bitwise NOT of OPA
0111	NOT_B	Bitwise NOT of OPB
1000	SHR1_A	Right shift OPA by 1
1001	SHL1_A	Left shift OPA by 1
1010	SHR1_B	Right shift OPB by 1
1011	SHL1_B	Left shift OPB by 1
1100	ROL_A_B	Rotate left OPA by OPB
1101	ROR_A_B	Rotate right OPA by OPB

Result

Conclusion

Future work