CMPE 260 Laboratory Exercise 1
Introduction to Vivado & Simple ALU

Andrei Tumbar
Performed: February 1st
Submitted: February 9th

Lab Section: 1
Instructor: Moskal
TA: Jacob Meyerson

Lecture Section: 2
Professor: Cliver

Abstract

In this laboratory exercise, the basics of VHDL were revisited and applied by implementing a simple multi-function ALU. The ALU has the ability to apply one of six operations given an input vector. `OR`, `AND`, `XOR`, `SLL` (Shift-Left-Logical), `SRL` (Shift-Right-Logical), and `SRA` (Shift-Right-Arithmetic). These operations are binary operations of N-bit width and could be controlled by the test-bench running the simulation. Two sets of test-benches were written to test the ALU in both 4-bit and 32-bit modes.

Design Methodology

Implementing the 32-bit version of the ALU operations involves providing generic parameters to the ALU chip entity. The generic parameter can be passed down to child ALU functors such as the SLL, SRL etc circuits.
A block diagram was created to illustrate the functionality of the ALU operations given a 4-bit operation select input.

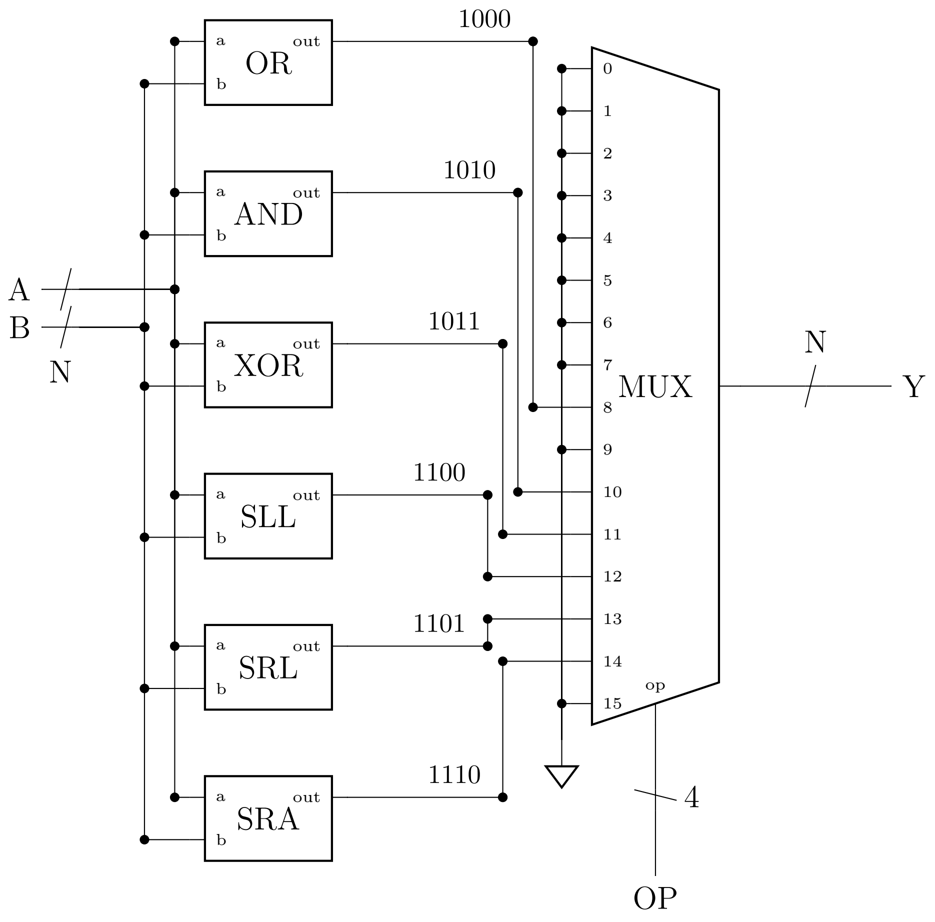


Figure 1: Layout of the N-bit ALU with six operations

Figure 1 shows a block diagram of the generic-sized input ALU `N`. The select signal (`OP`) is

4 bits long and therefore there are 16 different inputs this signal can take. The multiplexer in the diagram will handle a subset of the inputs to bind the outputs of valid `OP` values. Figure 1 shows the six valid operations and their corresponding binary values labeled. Each operation will take two N-wire signals `A` and `B`.

Right-shift design

To design the right-shift operations `SRL` (logical) and `SRA` (arithmetic), a similar approach was taken when compared to the `SLL` implementation. A table of every shift up to `N` bits was computed. Based on the shift amount `B`, a different vector at index `B` in the shift table was chosen. Shifts above `N` will output a `0` signal. The difference between the arithmetic and logical shifts is the value that the left bits are filled with. For a logical shift, the are always 0, for an arithmetic shift, they are equal to the value of the most significant bit in the input `A`.

Results & Analysis

To test the VHDL source code, a test-bench was written to analyse the behaviour of the circuit and all of its operations.
A behavioural waveform was generated.

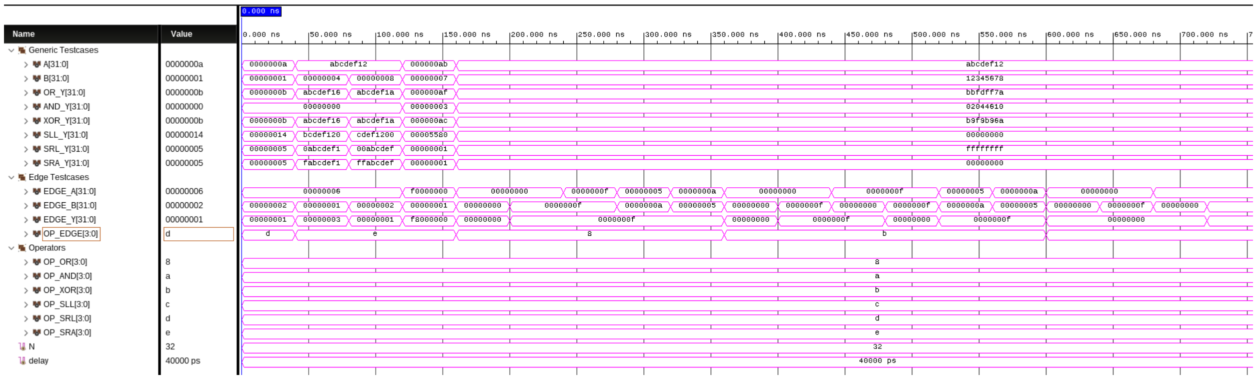
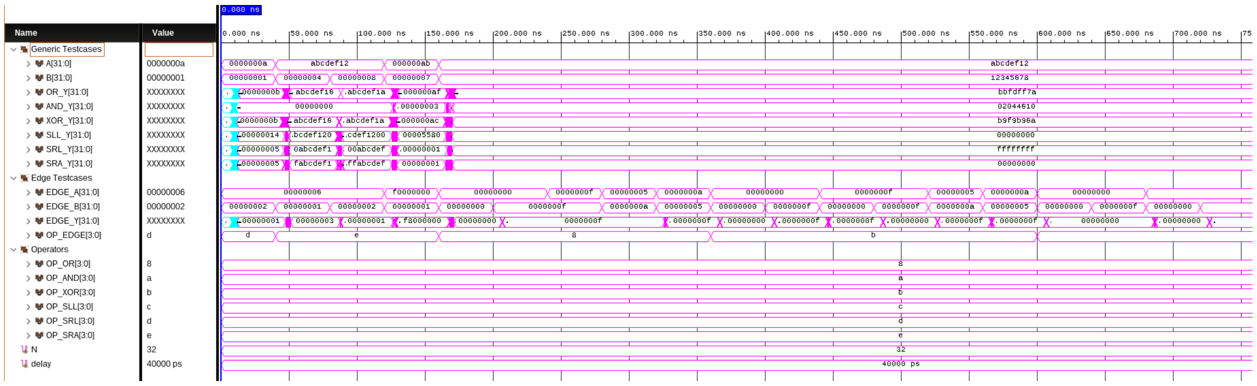


Figure 2: Screen capture of behavioural simulation in 32-bits

Figure 2 shows the waveforms generated from running a behavioural simulation of the VHDL test-bench. All values are denoted in hexadecimal representation for easy readability. The signals are split into three sections, Generic Testcases, Edge Testcases, and Operators. The operators section are simply constants signal values for each type of operation. These are used to understand the `OP_EDGE` signal values.
The Generic Testcases will test 4 different combinations of inputs `A` and `B` on all 6 different ALU operations. The output signals simulated in parallel and are named `[OP]_Y` where `[OP]` is the ALU operation being performed.

Following the behavioural simulation, a Post-Implementation Timing simulation was performed. A corresponding waveform was generated.



Name	Value
Generic Testcases	
A[31:0]	0000000a
B[31:0]	00000001
OR_Y[31:0]	XXXXXXXX
AND_Y[31:0]	XXXXXXXX
XOR_Y[31:0]	XXXXXXXX
SLL_Y[31:0]	XXXXXXXX
SRL_Y[31:0]	XXXXXXXX
SRA_Y[31:0]	XXXXXXXX
Edge Testcases	
EDGE_A[31:0]	00000006
EDGE_B[31:0]	00000002
EDGE_Y[31:0]	XXXXXXXX
OP_EDGE[3:0]	d
Operators	
OP_OR[3:0]	8
OP_AND[3:0]	a
OP_XOR[3:0]	b
OP_SLL[3:0]	c
OP_SRL[3:0]	d
OP_SRA[3:0]	e
N	32
delay	40000 ps

Figure 3: Screen capture of timing simulation in 32-bits