Lab Assignment 2

Implementing Combinational and Sequential Logic in VHDL

Task 1

Draw a detailed block diagram of the ALU (Arithmetic Logic Unit), specified using Fig. 1 and Tables 1 and 2.

Then develop a dataflow VHDL description of the ALU. Also, debug and verify your code using the respective testbench developed in Lab 1.

The ALU performs 16 different operations, including

- Logic Operations
- Arithmetic Operations
- Shifts and Rotations
- BCD (Binary Coded Decimal) to binary conversion.

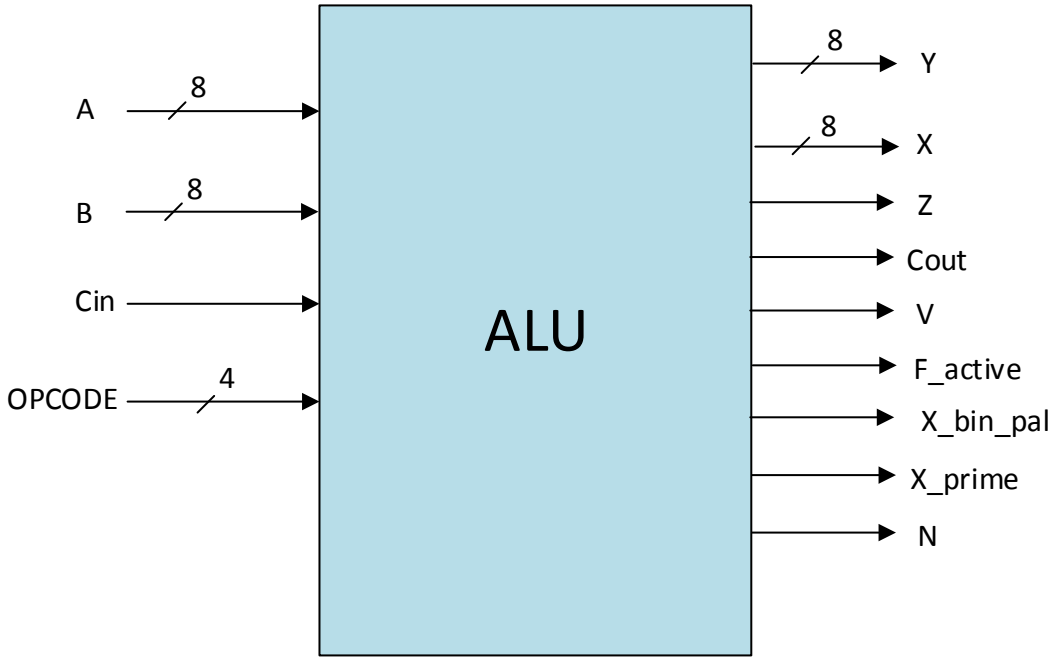

Fig. 1 Interface of ALU.

Table 1. Meaning and width of inputs and outputs

Name	Mode	Width	Meaning
A	IN	8	Input A
B	IN	8	Input B
Cin	IN	1	Carry In
OPCODE	IN	4	Operation Code
X	OUT	8	Output or Least Significant Byte of Output
Y	OUT	8	Most Significant Byte of Output or Zero
Z	OUT	1	Zero Flag
Cout	OUT	1	Carry out Flag
V	OUT	1	Overflow Flag
F_active	OUT	1	Logical OR of Z, Cout and V
X_bin_pal	OUT	1	Flag set to high when the output X is a binary palindromic number (numbers that remain the same when binary digits are reversed)
X_prime	OUT	1	Flag set to high when the output X is a prime number
N	OUT	1	Flag set to high when the output X is a negative number

Each operation performed by ALU has a specific OPCODE as shown in the table below. Z, Cout, V, N are one-bit flags. They are either affected by a given operation (which means that they are set to 1 or 0 depending on the result of a given operation) or unaffected, as summarized in Table 2. The affected flag is denoted by '↕', and unaffected by '–'.

In the combinational version of the circuit (ALU in Task 1), '–' represents 0, as all outputs must be fully determined for each set of inputs.

In the sequential version of the circuit (ALU_SEQ in Task 2), '–' represents no change compared to the value of a given flag determined as a result the previous operation.

The general rules are summarized below:

- All operations affect the zero flag, Z. This flag is set if the result of the operation is zero, and is cleared otherwise.

On top of that:

- Unsigned addition and subtraction affect the carry out flag, Cout. Cout is set when the result is out of range for 8-bit unsigned numbers, i.e., either smaller than 0, or greater than 255, and is cleared otherwise.
- Signed addition and subtraction affect the overflow flag, V.
- V is set when the result is out of range for 8-bit signed numbers (in the two's complement representation), i.e., either smaller than -128, or greater than 127, and is cleared otherwise.
- Signed and unsigned multiplications don't affect either Cout or V flags. The result can always be represented using 16 bits stored in registers Y:X (with Y representing 8 most significant bits, and X representing 8 least significant bits of the result).
- Rotation with carry affects Cout. The carry flag, Cout, is changed to the value of the bit rotated to Cout.
- Shift operations affect Cout. The carry flag, Cout, is changed to the value of the bit shifted out of the input.
- BCD to binary conversion affects Cout. Cout is set if the most significant byte of the result, Y, is different than zero, and is cleared otherwise.
- X_prime and X_bin_pal are affected by every operation, and depend only on the value of X. X_prime indicates whether X is prime when treated as an 8-bit unsigned number, X_bin_pal indicates whether X is a binary palindromic number.
- N is affected only by the signed addition/subtraction/multiplication and by arithmetic and logic shifts. For these operations, it should be set to the MSB of the result, either MSB of X or MSB of Y:X, depending on the operation. For the logic shift right, N=0, based on the definition of the operation and the definition of the flag. N should be cleared for all the remaining operations.
- If the output Y is not affected by a given operation, it is set to 0 for the combinational version of the circuit (ALU in Task 1),, and remains unchanged for the sequential version of the circuit (ALU_SEQ in Task 2).

Table 2. The meaning of operations performed by ALU for each OPCODE.

OPCODE	OPERATION	FORMULA	Z	Cout	V	X_bin_pal	X_prime	N
0000	AND	X = A AND B	↕	–	–	↕	↕	0
0001	OR	X = A OR B	↕	–	–	↕	↕	0
0010	XOR	X = A XOR B	↕	–	–	↕	↕	0
0011	XNOR	X = A XNOR B	↕	–	–	↕	↕	0
0100	Unsigned Addition	(Cout:X) = A + B	↕	↕	–	↕	↕	0
0101	Signed Addition	X = A + B	↕	–	↕	↕	↕	↕
0110	Unsigned Addition with Carry	(Cout:X) = A + B + Cin	↕	↕	–	↕	↕	0
0111	Signed Multiplication	(Y:X) = A * B	↕	–	–	↕	↕	↕
1000	Unsigned Multiplication	(Y:X) = A * B	↕	–	–	↕	↕	0
1001	Unsigned Subtraction	X = A - B	↕	↕	–	↕	↕	0
1010	Rotation Left	X = A <<< 1	↕	–	–	↕	↕	0
1011	Rotation Left with Carry	(Cout:X) = (Cin:A) <<< 1	↕	↕	–	↕	↕	0
1100	Logic Shift Right	X = A >> 1	↕	↕	–	↕	↕	0
1101	Arithmetic Shift Right	X = A >> 1	↕	↕	–	↕	↕	↕
1110	Logic Shift Left	X = A << 1	↕	↕	↕	↕	↕	↕
1111	BCD to Binary Conversion	(Y:X) = BCD2BIN(B:A)	↕	↕	–	↕	↕	–

The notation used in the table is as follows:

- (Y:X) denotes the concatenation of Y and X, with the most significant part stored in Y, and the least significant part stored in X.
- ↕ means that a given flag is affected, – means that a given flag is unaffected (cleared for ALU, and left unchanged for ALU_SEQ).
- A <<< 1 = rotation of A left by one position.
- A << 1 = shift of A left by one position.
- A >> 1 = shift of A right by one position.

- BCD2BIN(B:A) = conversion of a 4-digit decimal value encoded in BCD (binary-coded decimal) to a binary value. The input may vary between 0000 and 9999 (decimal). 4 bits are used to represent each decimal digit in BCD.

Task 2

Develop a mixed-style VHDL RTL description of a sequential ALU (Arithmetic Logic Unit), specified using the interface in Fig. 2 , the internal block diagram in Fig. 3 and the operation of the circuit explained in Tables 3. In order to do so, you will have to update the block diagram in Fig. 3 and add the controller logic (used to generate all enable signals and select signals).

Then, debug and verify your code using the respective testbench developed in Lab 1.

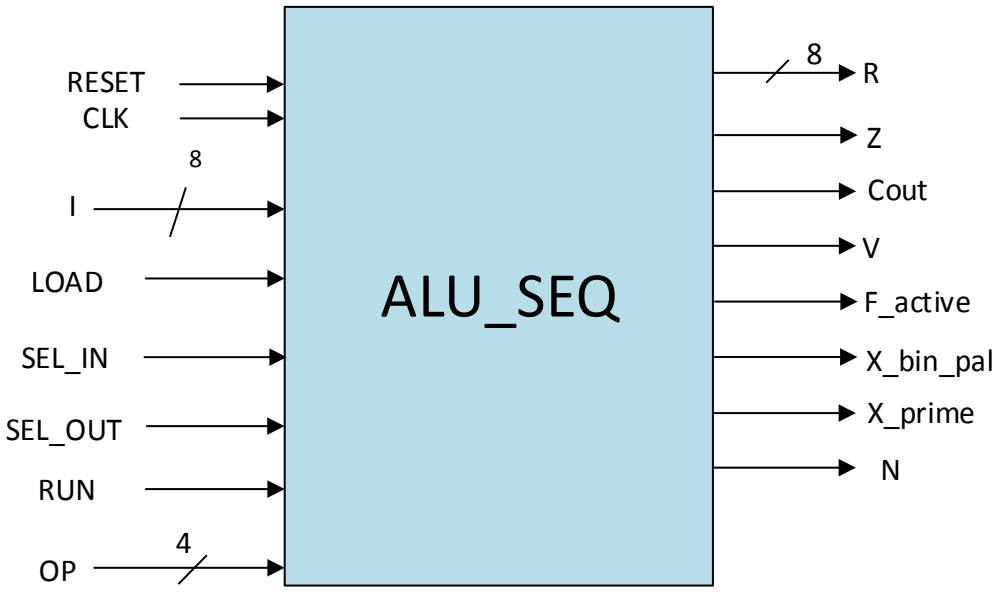

Fig. 2 Interface of ALU_SEQ.

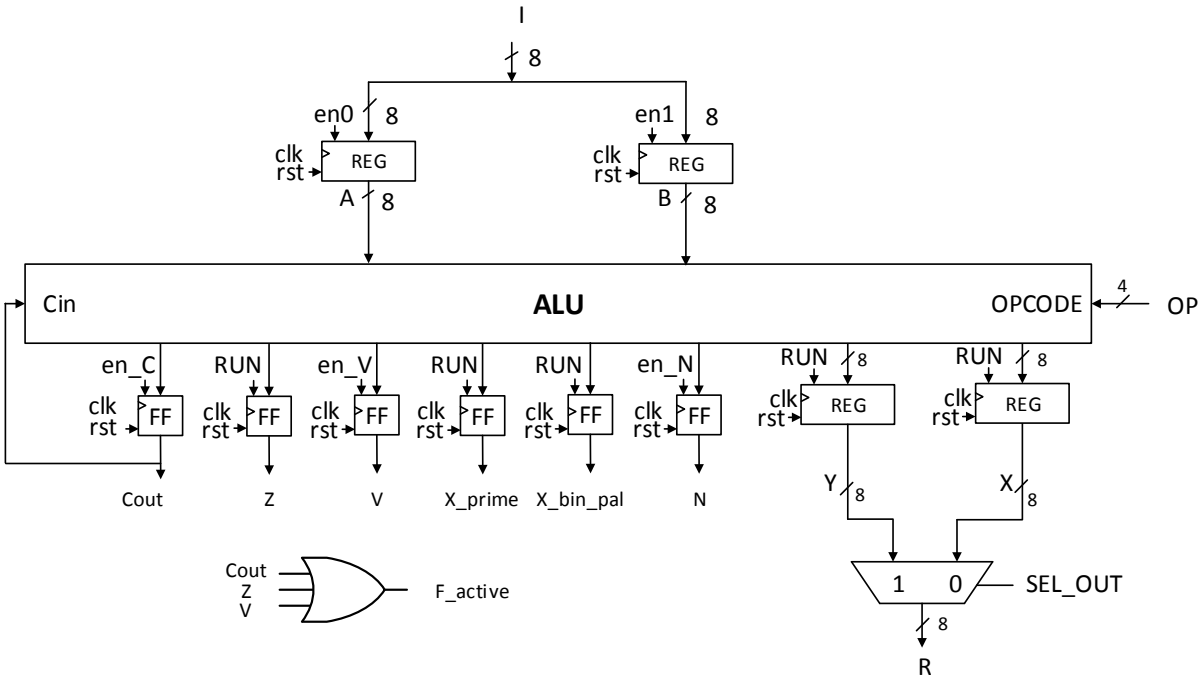

Fig. 3 Block Diagram of ALU_SEQ.

Table 3. Meaning and width of inputs and outputs

Name	Mode	Width	Meaning
CLK	IN	1	System clock
RESET	IN	1	Asynchronous reset active high
I	IN	8	Value of an operand A or B
LOAD	IN	1	Loading value of input I to one of the internal registers holding A or B (control signal active for one clock period; the action takes place at the rising edge of the clock)
SEL_IN	IN	1	0: loading register A 1: loading register B
OP	IN	4	Operation mode
RUN	IN	1	Writing the result to registers holding X and Y (control signal active for one clock period; the action takes place at the rising edge of the clock)
SEL_OUT	IN	1	0: R = X 1: R = Y

R	OUT	8	Byte of a result
Z	OUT	1	Zero flag
Cout	OUT	1	Carry out flag
V	OUT	1	Overflow flag
F_active	OUT	1	Logical OR of Z, Cout and V
X_bin_pal	OUT	1	Flag set to high when the output X is a binary palindromic number (numbers that remain the same when binary digits are reversed)
X_prime	OUT	1	Flag set to high when the output X is a prime number
N	OUT	1	Flag set to high when the result is negative

Enable signals en_C, en_V, en_Z, en_N, en_Xp, en_Xbp can be calculated based on OPCODE, RUN and Table 2 of the specification.

Task 3 (tested during demonstration):

Be prepared to demonstrate the operation of all your testbenches using Aldec Active-HDL and Xilinx ISim.

You may be tested on your level of understanding of this simulators, based at least on the following features.

- Adding signals to the Waveform window.
- Including signals from lower levels of hierarchy.
- All options to run Simulation.
- Introducing Breakpoints and showing the execution of logic before and after Breakpoints in the Waveform window.
- Measuring time intervals.
- Dealing with Buses (Expanding and viewing all bits of a signal).
- Taking a signal and changing Radix to Decimal, Binary and Hexadecimal.
- Saving the waveforms in Native format of the simulators.
- Printing output of the simulators to PDF files.
- Clearing waveforms.

In your report, please describe any problems you have encountered while using Aldec Active-HDL and Xilinx ISim. Please also grade each simulator on the scale from 0 to 10, based on your experiences with using it. Shortly, justify your rating.

Bonus Task:

Modify your code for the ALU in such a way that it supports variable shifts and rotations, instead of fixed shifts and rotations.

Namely, assume that the meaning of the following opcodes has changed compared to Task 1.

OPCODE	OPERATION	FORMULA	Z	Cout	V	X_bin_pal	X_prime	N
1010	Variable Rotation Left	X = A <<< B	↕	–	–	↕	↕	0
1011	Variable Rotation Left with Carry	(Cout:X) = (Cin:A) <<< B	↕	↕	–	↕	↕	0
1100	Variable Logic Shift Right	X = A >> B	↕	↕	–	↕	↕	0
1101	Variable Arithmetic Shift Right with Rounding	X = (A >> B) + A(B-1)	↕	↕	–	↕	↕	↕
1110	Variable Logic Shift Left	X = A << B	↕	↕	↕	↕	↕	↕

Notation used in this table is as follows:

A <<< B = rotation of A left by the number of positions given by a value of B.

A << B = shift of A left by the number of positions given by a value of B.

A >> B = shift of A right by the number of positions given by a value of B.

X = (A >> B) + A(B-1) = An arithmetic shift by the number of positions given by B followed by a rounding operation, which consists of adding the value of the B-1st bit of A to the result of the shift.

Modify your testbench to fully debug and verify your code for the Bonus Task.

Deliverables:

1. Complete block diagram of the ALU from Task 1 and updated block diagram with control logic from Task 2.
2. Dataflow and Mixed-style RTL VHDL code for Task 1 and Task 2 and possibly Bonus Task.
3. Outputs from all simulations proving the correct operation of codes developed in Task 1, Task 2 and possibly bonus task in the form of PDF files.
4. Short report describing the status of your codes for Task 1 and Task 2. Additionally, please describe any problems you have encountered while coding in VHDL or using Active-HDL and ISim. Please grade each simulator on the scale from 0 to 10, based on your experiences with using it. Shortly justify your rating.

Important Dates

	Monday Section	**Wednesday Section**	**Thursday Section**
Hands-on Session and Introduction to the Experiment	02/02/2015	02/04/2015	02/05/2015
Deliverables Due	**02/11/2015, 4:00pm**	**02/11/2015, 7:00pm**	**02/12/2015, 7:00pm**
Demonstration Due	**02/11/2015, 10:00pm**	**02/11/2015, 10:00pm**	**02/12/2015, 10:00pm**