VLSI Digital Design

LAB ASSIGNMENT 2

32-bit ALU Design

0. Objectives

The objective of this laboratory assignment consists in the design of the execution unit of the microprocessor that will be developed along the course. This execution unit is structured as a 32-bit Arithmetic-Logic Unit (ALU). As specified in the following section, besides executing basic arithmetic and logic operations, this unit will contain a 32-bit pseudo-random number generator.

1. Specification

The system to be designed has 5 inputs, ***clk***, ***reset***, ***opcode(4 downto 0)***, ***a(31 downto 0)*** and ***b(31 downto 0)*** and two outputs, ***result(31 downto 0)*** and ***nzco(3 downto 0)***.

Input ***clk*** is the system global clock signal, and its rising edge has to trigger all the memory elements. The ***reset*** input signal controls a synchronous system reset, and it is active at level high ('1'). The operation to be carried out by the ALU is specified by the ***opcode*** input signal. Input signals ***a*** and ***b*** store the two input operands for the arithmetic or logic operations.

The ***result*** output signal stores the operation result specified by the ***opcode*** input. The ***nzco*** output signal contains the flag values resulting from the ALU instruction execution. The interpretation of each one of the bits that compose the ***nzco*** output is the following:

- **nzco(3):** "Negative" flag. It indicates whether the resulting value of the selected operation is negative. To generate this flag, it has to be taken into account that both input operands and the output result are interpreted in two's complement. This flag has to be verified for any operation.
- **nzco(2):** “Zero” flag. It indicates that the result of a given operation is zero. This flag has to be verified for any operation.
- **nzco(1):** “Carry” flag. With this propagation flag arithmetic and shift instructions can be chained. It has to be verified only for arithmetic and logic shift operations. For any other an operation its value has to be '0'.
- **nzco(0):** “Overflow” flag. It indicates that an overflow condition has appeared when executing a 2's complement arithmetic operation. This implies that the result is not correctly represented with the selected representation format. This flag has to be verified only when arithmetic operations are executed. For any other operation its value has to be '0'.

Concerning the ALU functions to be implemented, Table 1 shows the mnemonics, the value of each input opcode and the function associated with each of the instructions that have to be interpreted by the ALU.

MNEMONIC	OPCODE	OPERATION
NOP	00000	No operation is done. El result has to be equal to the ALU input operand ***a***. Flags “overflow”, “carry” and "negative" have to be forced to ‘0’. Flag “zero” has to be forced to ‘1’.
OR	00001	The result signal returns the OR logic function of the ALU two input operands.
XOR	00010	The result signal returns the XOR logic function of the ALU two input operands.
NOR	00011	The result signal returns the NOR logic function of the ALU two input operands.
AND	00100	The result signal returns the AND logic function of the ALU two input operands.
ADD	00101	The result signal returns the arithmetic sum of the ALU two input operands.
SUBB	00110	The result signal returns the ***a*** input value minus ***b*** input value**.**
NOT	00111	The result signal is obtained inverting all the bits of the ***b*** input value.
ROR	01000	Rotate right. The result is obtained by rotating the input ***a*** value to the right the number of positions indicated by the 3 less-significant bits of input ***b.***
SLL	01001	Shift-left logical. The result is obtained by shifting the input ***a*** value to the left the number of positions indicated by the 3 less-significant bits of input ***b.*** The less significant part of the result is filled with '0' for each encoded shift.
SLR	01010	Shift-right logical. The result is obtained by shifting the input ***a*** value to the right the number of positions indicated by the 3 less-significant bits of input ***b.*** The most significant part of the result is filled with '0' for each encoded shift.
SRA	01011	Shift-right arithmetic. The result is obtained by shifting the input ***a*** value to the right the number of positions indicated by the 3 less-significant bits of input ***b.*** The most significant part of the result is filled with the value of input ***a*** most significant bit for each encoded shift.
JUMP	01111	Jump instruction. The result signal returns the arithmetic sum of the ALU two input operands.
JUMPI	10010	Immediate jump instruction. The result signal returns the arithmetic sum of the ALU two input operands.
LOAD	10011	Load instruction. The result signal returns the arithmetic sum of the ALU two input operands.
MOVE	11011	Data move instruction. The result signal returns the value of input ***b***.
BITSET	11100	This instruction sets (forces to '1') the bit of input ***a*** that is located at the position encoded by the 5 less-significant bits of input ***b***.
BITCLEAR	11101	This instruction resets (forces to '0') the bit of input ***a*** that is located at the position encoded by the 5 less-significant bits of input ***b***.
RANDOMSET	11110	Loads the value of input ***b*** to the pseudo-random number generator.
RANDOMIZE	11111	The result signal returns the current content of the pseudo-random number generator.

Table 1. ALU instruction set.

Regarding the subsystem in charge of generating the pseudo-random numbers, the proposed structure, shown in Fig. 1, corresponds to the so-called Linear Feedback Shift Register (LFSR).

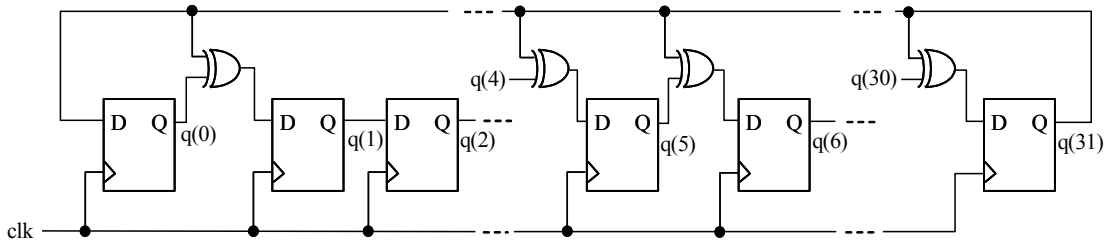

Figure 1. Structure of the pseudo-random number generation subsystem to be included into the ALU.

To facilitate the structure understanding, the elements required to load the value corresponding to input ***b*** when an instruction RANDOMSET is executed are not shown in Figure 1. Also the register reset input has been omitted in the figure. This input has to be controlled by the ALU reset input signal.

The following considerations have to be taken into account for the ALU proper implementation:

- The only sequential element of the design is the pseudo-random number generator.
- Inputs ***a***, ***b*** and ***opcode*** are externally registered, thus **they do not have to be loaded into any register**. These inputs are stable during a ***clk*** clock period. For simulation purposes, input changes have to be produced with a 2 ns delay after the clock rising edge.
- All instructions are performed in a single clock cycle.
- System clock frequency is 25 MHz.
- Concerning arithmetic operations, functional subsystems can be implemented with any architectural option. However, the selected option should have the minimum area occupancy. Moreover, these subsystems must operate properly at the specified clock frequency.
- The ALU entity name has to be ***execution_unit***.
- The entity input and output ports have to be named as specified.

The tasks to be done in this laboratory assignment are:

- Describe the ALU in VHDL.
- Simulate functionally with Modelsim (no delays), verifying the correct syustem operation.
- Synthesize with Precision. Verify that the resulting flip-flop number from synthesis is exactly 32, that is, the corresponding to the LFSR, and make sure that no additional sequential elements (latchs) appear. Obtain the critical path area and delay results.
- Compile with ISE (placement and routing). Use the device specified in the initial tutorial. Verify the device resource occupancy.
- Simulate with Modelsim including the backannotation delays. Verify that the ALU still operates properly at the specified clock frequency.