Indian Institute of Technology, Indore
Department of Computer Science and Engineering

Project: Simple Processor Design

Objective

This objective of this project is to design and construct a simple Processor in VHDL

Upon completion of this project, the student should be able to:

• Design and build all functions of a simple Processor.

• Design, simulate and program a simple Processor using VHDL (Based on Functional Simulation in Xilinx Simulator).

Part I: Description of a simple processor:

The processor of a computer is usually divided into at least four components, the *control* unit which performs the fetching the signals, the *bus* which controls access to the data and address bus, *latches* for temporary storage, and the *arithmetic/logic unit (ALU)* that performs arithmetic and logical operations. In this experiment we will focus on developing all the four components working concurrently. The processor will be tested by performing the functions of all. The heart of every computer is an Arithmetic Logic Unit (ALU). This is the part of the computer which performs arithmetic operations on numbers, *e.g.* addition, subtraction, *etc*. In this lab you will use VHDL to implement an ALU with eight functions.

• A general block diagram of a simple processor is shown in Figure 1.

1) Synthesize, simulate and verify the functionality.
2) Create symbol from schematic for being used as a functional block for use later in the schematic design of your simple processor.

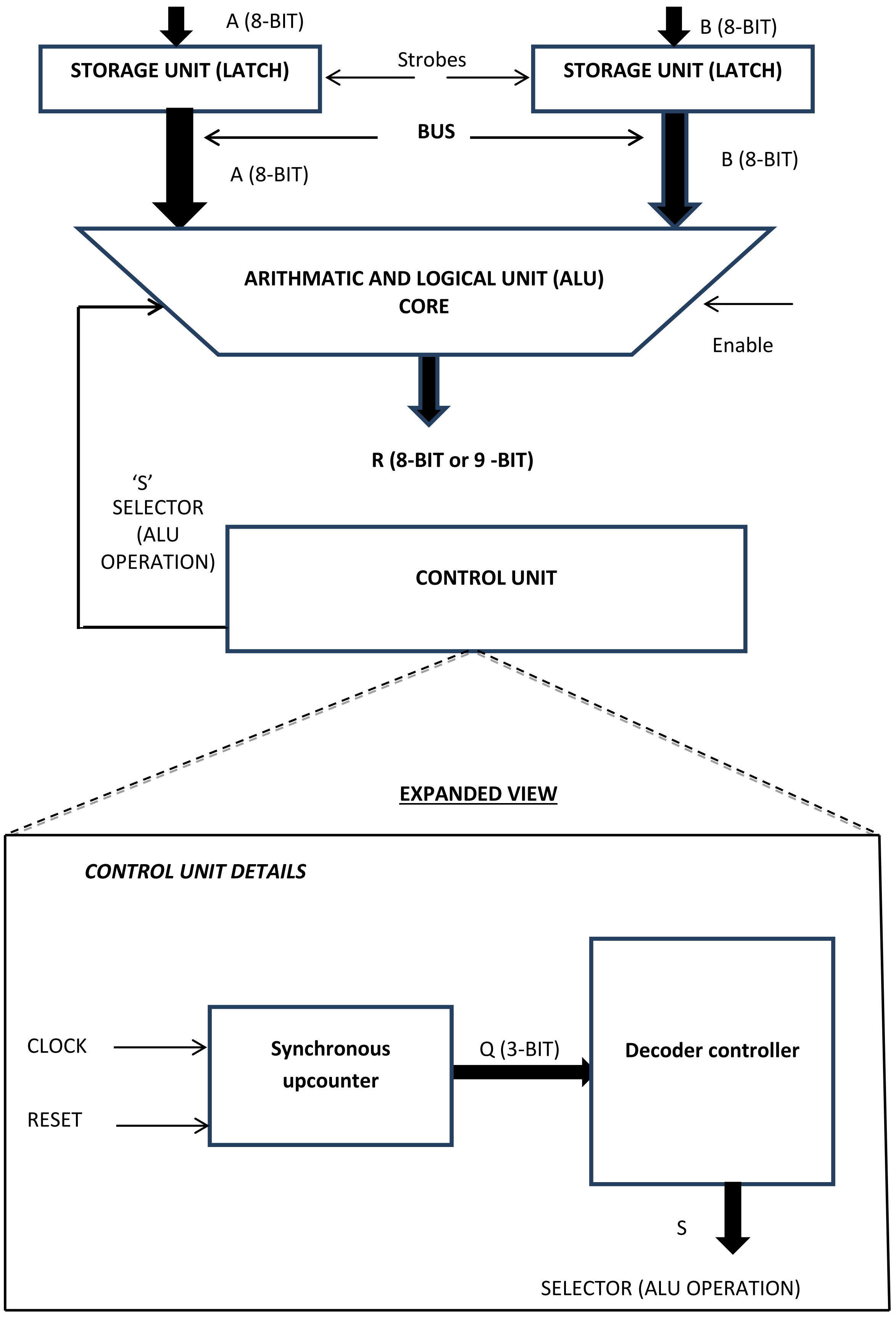

Figure1. The Block Diagram of a simple Processor

Problem sets for the Project:

1. ***The outputs should be generated in the sequence mentioned in the question.***

   a) Take input two 8 bit numbers (A and B) to produce their difference, keep the output constant for the next 3 clock cycles/periods and then alternate the output to produce sum for last 4 cycles.

   b) Take input two 8 bit numbers (A and B) to alternate between inverting output of A and B in consecutive clock cycle/periods.

   c) Take input two 8 bit numbers (A and B) to alternate between NAND for 1st 4 clock periods and OR outputs in remaining 4 clock periods.

   d) Take input two 8 bit numbers (A and B) to produce the OR output for 2 consecutive clock periods, XOR for next 2 consecutive clock periods, SUM for next 2 clock periods and INV for next 2 clock periods.

   e) Take input two 8 bit numbers (A and B) to produce no result for consecutive 7 clock cycles/clock periods and generate a SUM in the next clock period. The output pattern should repeat following the SUM.

   f) Take input two 8 bit numbers (A and B) to produce the OR output for 3 consecutive clock periods, NAND for next 3 consecutive clock periods and 'no result' for the remaining clock periods. *(Note: NULL keyword does not produce no result. Use of some numeric value should be made to drive the output to no result i.e. LEDs should be turned off)*

   g) Take input two 8 bit numbers (A and B) to generate the XOR result in the first clock period. Hold the result in the remaining 7 clock periods.

   h) Take input two 8 bit numbers (A and B) to generate the OR result followed by AND result in first 2 clock periods. Hold the result in the next 6 clock periods. Repeat the OR followed by AND result pattern.

2. ***The outputs should be generated in the sequence mentioned in the question.***

   a) Take input two 8 bit numbers (A and B) to produce an alternate output of the single incremented values of A and single decremented value of A in consecutive clock periods.

   b) Take input two 8 bit numbers (A and B) to produce the sum in first clock period, followed by increment of A by 1, increment by 2, increment by 3, increment by 4, increment by 5, increment by 6 and increment by 7 in consecutive clock periods.

c) Take input two 8 bit numbers (A and B) to produce the sum, difference, decrement of A by 2, logical shift right (***srl***) A by 3, rotate left A by 2 (***rol***), rotate right A by 2 (***ror***), logical shift left (***sll***) A by 2 and increment B by 1 in consecutive clock periods.

d) Take input two 8 bit numbers (A and B) to generate the rotate left A by 2 (***rol***). Hold the output for the next 3 clock cycles before changing the output to produce the logical shift left (***sll***) A by 2 for the remaining 4 clock periods.

e) Take input two 8 bit numbers (A and B) to generate the rotate right A by 2 (***ror***). Hold the output for the next 3 clock periods before changing the output to produce the logical shift right (***srl***) A by 2 for the remaining clock periods.

f) Take input two 8 bit numbers (A and B) to generate the decrement A by 2. Hold the output for the next 4 clock periods before changing the output to produce the increment A by 2 for the remaining clock periods.

g) Take input two 8 bit numbers (A and B) to generate the rotate left A by 3 (***rol***) followed by rotate right A by 3 (***ror***). Alternate this output in the remaining clock periods.

h) Take input two 8 bit numbers (A and B) to generate the logical shift left (***sll***) A by 3 (***sll***) followed by logical shift right by 3 (***srl***). Alternate this output in the remaining 6 clock periods.