Royal Institute of Technology (KTH)
School of Electrical Engineering and
Computer Science (EECS)
Department of Electronics (ELE)

Version 1.2
Last revision 2019-09-04

Laboration 2

IL1331 VHDL Design

IL2203 Digital Design and Validation using HDLs

Register File and Datapath

Student Name:……..

Personal Number:……

Date of Approval:……

Assistant:……

Laboration 2

Register File and Datapath

In the laboration, we first complete the ALU from lab1 and make it clocked by by adding registers to it. We then go on to create a Register File for the CPU/microcontroller we are building, and round it off by connecting the components together into a Datapath, i.e., the Execution pipeline of the processor, and build a test bench to test it. Finally, we download the Datapath component on the prototype FPGA board.

Preparations

1) Complete the ALU you designed in Lab1 clocked by adding registers on all outputs.
2) Create a register file according to the specification in the Appendix A.
3) Create a datapath according to the specification in the Appendix B.
4) Think through carefully which test patterns you need to toggle all wires in the final datapath when all components are connected, and write them down in a table. This table will serve as your test protocol.

Tasks

Task 1. RTL Modelling of the ALU

1) Complete the RTL model of the ALU from Lab1 by adding registers to all outputs. The register should have one asynchronous reset signal, and one enable signal (active high). The functionality/specification should be the same as outlined in the Appendix of Lab1, but outputs should come one clock cycle later. Also, if the Enable signal is not active (=low signal) the register should keep the previous value. The specification of the new signals are found in Appendix A.
2) Modify the automatic test bench for the ALU, and test all possible combinations of the registered ALU. Use one process for generating the signals (the signal generator), and a second process for checking the signals (the monitor).

Task 2. RTL Modelling of the Register File (RF)

1) Create a RTL model of the Register File. The functionality/specification is found in Appendix B.
2) Write a testbench that test the RF. A good way to test registers/memory components is writing alternative patterns of ‘0’ and ‘1’ to the bits, and see if reading the register returns the same sequence (patterns like “1010101” followed by “0101010” also checks if the bits are leaking to their neighbors).

Task 3. The Datapath

1) Create a structural model that connect the Register file with the ALU. The specification of the datapath is shown in Appendix C.
2) Test the Datapath. The testbench should apply an Opcode the ALU, set the appropriate enabling signals and write the result back to the RF. A good example would be to consecutively add 1 a number of times, and check the result stored in the RF.
3) Create a clock divider component. It should divide the 100 MHz clock on the board to an internal clock of approximately 1 Hz. The easiest way to achieve that is to make an incrementer (add 1), and use the MSB of it as the Clk output. Write a testbench and test the clock divider.

Task 4. Prototyping the ALU on an FPGA board

1) Add the clock divider to the datpath component. If the frequency is faster than ~1 Hz, you don't see what is happening on the LEDs.
2) Import and synthesize the design in Quartus II
3) Download the design onto the FPGA. For prototyping, connect the 100 MHz clock signal to the input of the clock divider and the OpCode signal to manual switches. Perform the add-1 test on the prototype and display the result on the LEDs.

Passing Requirements

To pass the lab, the student should be able to show the assistants:

- Well-documented, well-commented and proper indented VHDL-models.
- Simulation results and waveforms in Modelsim.
- A functioning prototype on the FPGA board.
- Answer any questions that the assistants may have during the lab examination.
- When the lab assistant has approved the lab, he/she will sign the front page of it. Keep that document until the course has ended and your grade has been registered in Ladok. It is the only proof you have that you have made the lab.

Appendix: Registered ALU

The Arithmetic-Logic Units has as its task to perform all arithmetic and logic operations in the computer. Its external pins are shown in the block diagram below.

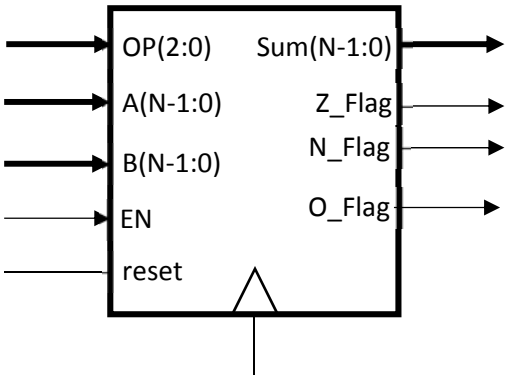


Figure 1. Block diagram registered ALU

Signal(s)	Function	Output
OP(2:0)	ALU operation according to OP(2:0). Functionality of OP Codes is outlined in the Appendix of Laboration 1.	Sum gets the value of the operation indicated by OP(2:0), after a posititve edge on Clk
EN	Enable signal for the outputs. Active high.	If EN='1' update the outputs according to the function indicated on input OP(2:0). If EN='0', keep old value.
Clk	Clock signal. Positive edge.	-
Reset	Asynchronous reset. Active high.	If Reset='1', set all output registers to '0'.

Appendix B. The Register File

The Register File has as its task to store all register values in the computer, and provide them to functional units upon request. The unit should have two parameters (M,N), used as 1) the number of registers (2^M registers) and the width of each register (N bits wide). It can read two registers every clock cycle and store one value every clock cycle. A reset should set all bits to '0'. The external pins of the RF are shown in the block diagram below.

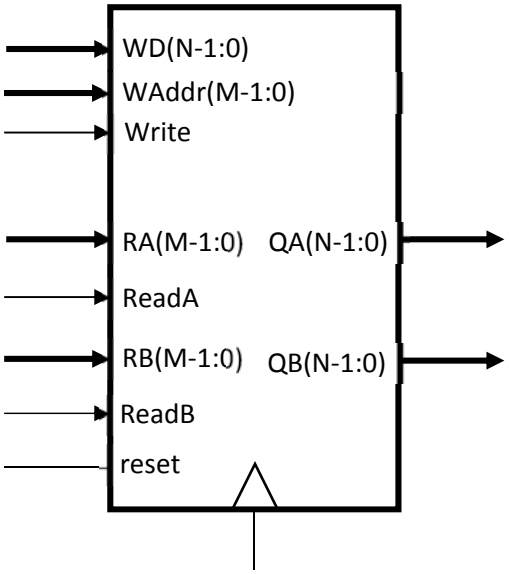
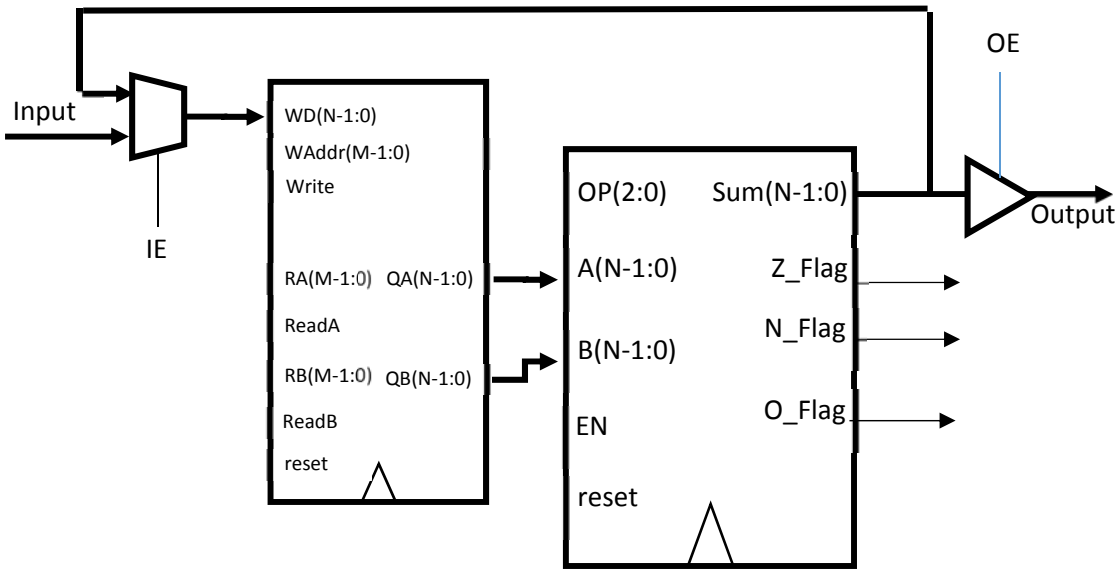


Figure 2. Block diagram – Register File

Signal	Function	Output
WD(N-1:0)	Write Data (N bits wide)	-
WAddr(M-1:0)	Write Address (M bits wide)	-
Write	Write='1' writes register pointed to by Write Address. Register should be written on positive flank.	-
RA(M-1)	Read Address A (M bits wide)	-
ReadA	ReadA='1' reads register pointed to by Read Address A. Output should be updated immediately. ReadA='0' should output a 0.	Register output QA updated accordingly.
QA	QA output.	Outputs a value from the register pointed to by RA.
RB(M-1)	Read Address B (M bits wide)	-
ReadB	ReadB='1' reads register pointed to by Read Address B. Output should be updated immediately. ReadB='0' should output a 0.	Register output QB updated accordingly.
QB	QB output.	Outputs a value from the register pointed to by RB.

Appendix C. The Datapath

The Datapath has as its task to perform all operations in the computer. It contains the RF, the ALU, and input Mux, and an output register. The external pins, and the functionality of each pin has, are shown in the block diagram and the table below, respectively. Pins to the RF and ALU that has not been connected in the figure, should be connected directly to an output of the datapath component.



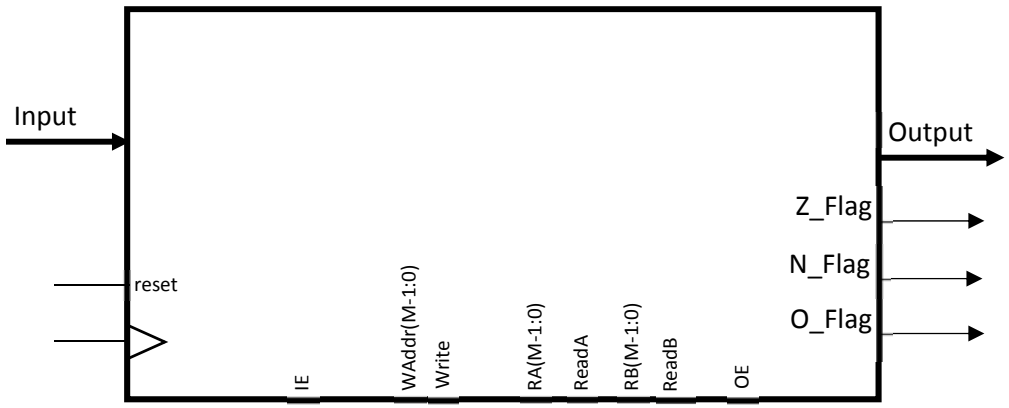


Figure 3. Block diagram – Datapath

Signal	Function	Direction
IE	Input Enable. If it is ‘1’, the Mux should select data from external pin Input. If it is ‘0’, the Mux should select data from the ALU output SUM.	Input
WAddr(M-1:0)	Write Address (M-bit). Points to which Address the Register file should store data.	Input
Write	Write signal. If it is ‘1’, the RF file should store the output of the MUX into the register pointed to by WAddr.	Input
RA(M-1:0)	Read Address A(M-bit). Points to which Address the Register file should read data A.	Input
ReadA	ReadA. If it is ‘1’, the input of the ALU input A should get the value from the register pointed to by RA	Input
RB(M-1:0)	Read Address B(M-bit). Points to which Address the Register file should read data A.	Input
ReadB	ReadB. If it is ‘1’, the input of the ALU input B should get the value from the register pointed to by RA	Input
OE	Output Enable. If it is ‘1’, the output tri-state buffer should output the sum output of the ALU. If it is ‘0’, the output of the tri-state buffer should be ‘Z’	Input
Output	Outputs the result of the tri-state buffer.	Output
Z_Flag	Zero flag	Output
N_Flag	Negative flag	Output
O_Flag	Overflow	Output