FH University of Applied Sciences TECHNIKUM WIEN

Chip Design 1

Specification of Calculator Project
Part 1: Overview of Calculator Project

Version: 1.0
Author: R. Höller

Introduction

As a final project, a simple calculator will be implemented during both the attendance phase and the distance learning phase of the course “Digital System Design 1”. Details of the project are described in six distance learning letters: This document as well as five learning letters named “IO Control Unit”, “Calculator Control Unit”, “Arithmetic Logic Unit”, “Top-Level Design” and “Synthesis & Implementation”, see CIS website. Please contact the supervisors if any problems or questions arise, so that they can be resolved during the attendance phase of the course.

Learning Outcomes of the Project

- How to implement a given specification for a digital system
- VHDL for design (entity, architecture, components and component instantiations, signals, variables, processes, combinatorial and sequential logic, …)
- VHDL for verification (testbench, clock and reset generation, generation of stimuli, …)
- Application of a VHDL coding style
- Application of a structured and clean design flow, starting from an idea to implementation and documentation

Functional Specification

- A simple calculator shall be implemented using the Digilent Basys3 FPGA development board. The FPGA contained on the board is a Xilinx Artix-7 FPGA (XC7A35T-1CPG236C) device.
- Two unsigned 12-bit integer values OP1 and OP2 as well as an arithmetic/logic operation (sum, multiplication, logical AND, square root …) are defined by the user via the switches and push buttons of the Basys3 board. The calculator performs the selected operation and displays the result on the 7-segment displays (DISP1). Details about the user interface as well as on the arithmetic/logic operations are described below.
- The design shall operate fully synchronous using with the 100 MHz clock from the board’s oscillator.
- An asynchronous high-active reset shall be used to initialize the design (BTNU button on the Basys3 board).
- Note, that there are individual project specifications for each student which depend on the number in the attendance list of the course (Table 2).

User Interface of the Calculator

There are two variants for the user interface of the calculator (called “Variant A” and “Variant B”) which are described in the following. Each student has to implement only ONE of these variants! Please have a look at Table 2 to see which variant you have to implement.

User Interface, Variant A

- After a reset, the leftmost digit of DISP1 shall display “1.” to indicate, that the unsigned 12-bit integer operand OP1 can be entered via the twelve switches SW0-SW11. Any changes on SW0-SW11 are immediately shown on the three rightmost digits of DISP1 in this state (i.e., the selected 12-bit value as a hexadecimal number) and all 16 LEDs should be dark, see left hand side of Figure 1.
- Once button BTNL is pressed by the user, the leftmost digit of DISP1 shall change to “2.”. Now, the twelve switches SW0-SW11 are used to define the unsigned 12-bit integer operand OP2. Again, any changes on SW0-SW11 should be immediately shown on the three rightmost digits of DISP1 in this state (the selected 12-bit value as a hexadecimal number) and all 16 LEDs should be dark, see right hand side of Figure 1.
- If button BTNL is pressed again, the leftmost digit shall display “o.”. At this point of time, the four switches SW12-SW15 are used to define the arithmetic/logic operation that has to be performed by the calculator on the previously entered operands OP1 and OP2 (according to Table 1). Any changes on SW12-SW15 are immediately shown on the three rightmost digits of DISP1 in this state (the abbreviation shown in column “DISP1” of Table 1) and all 16 LEDs should be dark, see left hand side of Figure 2.
- Finally, once button BTNL is pressed again, the calculator shows the signed result of the selected arithmetic/logic operation on the four digits of DISP1 (or indicates an error/overflow). Moreover, LED15 should be switched on to indicate, that the result is displayed on DISP1 in this state, see right hand side of Figure 2.
- If BTNL is pressed another time, the calculator jumps back to the reset state where operand OP1 can be reentered … and so on.

User Interface, Variant B

- After a reset, both operands OP1 are OP2 are initialized to zero and “Add” shall be the selected arithmetic/logic operation (see Table 1) which is immediately performed. Thus, the four digits of DISP1 show the signed result of the operation as a 16-bit hexadecimal number (“0000”). Moreover, LED15 should be switched on to indicate, that the result of the operation is displayed on DISP1 in this state, see right hand side of Figure 2.
- When button BTNL is pressed by the user, the leftmost digit of DISP1 shall change to “1.” to indicate, that the unsigned 12-bit integer operand OP1 can be entered via the twelve switches

SW0-SW11. Any changes on SW0-SW11 are immediately shown on the three rightmost digits of DISP1 in this state (i.e., the selected 12-bit value as a hexadecimal number) and all 16 LEDs should be dark, see left hand side of Figure 1.

- Once button BTNC is pressed, the leftmost digit of DISP1 shall change to “2.”. Now, the twelve switches SW0-SW11 are used to define the unsigned 12-bit integer operand OP2. Again, any changes on SW0-SW11 should be immediately shown on the three rightmost digits of DISP1 in this state (the selected 12-bit value as a hexadecimal number) and all 16 LEDs should be dark, see right hand side of Figure 1.
- If button BTNR is pressed, the leftmost digit shall display “o.”. At this point in time, the four switches SW12-SW15 are used to define the arithmetic/logic operation that has to be performed by the calculator, according to Table 1. Any changes on SW12-SW15 are immediately shown on the three rightmost digits of DISP1 in this state (the abbreviation shown in column “DISP1” of Table 1) and all 16 LEDs should be dark, see left hand side of Figure 2.
- Finally, once button BTND is pressed, the calculator jumps to the reset state described previously and performs the selected arithmetic/logic operation.
- A change of the operands OP1 and OP2 (via BTNL and BTNC) or the type of operation (via BTNR) as well as starting a new calculation (button BTND) should be possible in any order.

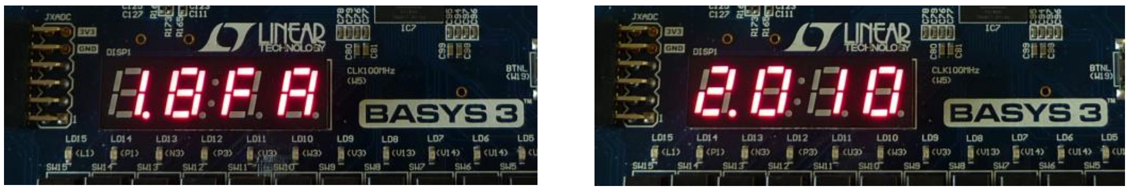

Figure 1: Entering Operand OP1 (left hand side) and OP2 (right hand side)

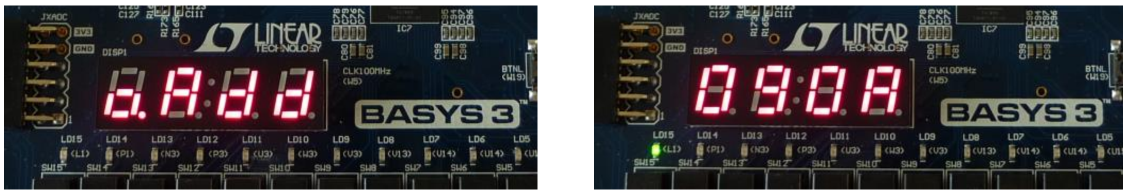

Figure 2: Entering the Type of Operation (left hand side) and Output of the Result (right hand side)

Arithmetic/Logic Operations

Table 1 shows all the arithmetic/logic operations that are supported by the calculator. Both operands OP1 and OP2 are treated as unsigned integer values with a data width of 12 bits for all operations. However, you do not have to implement all operations but only the operations that are given by Table 2! Have a look at the distance learning letter “Arithmetic Logic Unit” which describes how you should implement the operations.

SW12 to SW15	**DISP1**	**Operation**	**Result of Operation**
0000	`'Add'`	Add	OP1 + OP2
0001	`'Sub'`	Sub	OP1 - OP2
0010	`'X  '`	Multiply	OP1 * OP2
0011	`'di '`	Divide	Integer fraction of OP1 / OP2
0100	`'rE '`	Remainder of Division	Remainder of OP1 / OP2
0101	`'Sqr'`	Square	$OP1^2$ (value of OP2 will be ignored)
0110	`'Sro'`	Square Root	Integer fraction of $\sqrt{OP1}$ (value of OP2 will be ignored)
0111	`'Lb '`	Binary Logarithm	Integer fraction of $\log_2$ OP1 (value of OP2 will be ignored)
1000	`'no '`	Logical NOT	Performs bit-wise NOT of OP1 (value of OP2 will be ignored)
1001	`'And'`	Logical AND	Performs bit-wise AND of OP1 with OP2
1010	`'or '`	Logical OR	Performs bit-wise OR of OP1 with OP2
1011	`'Eor'`	Logical Ex-OR	Performs bit-wise "Exclusive-OR" of OP1 with OP2
1100	`'roL'`	Rotate Left	OP1 will be shifted left by one bit position and the most significant bit of OP1 becomes the least significant bit (value of OP2 will be ignored)
1101	`'ror'`	Rotate Right	OP1 will be shifted right by one bit position and the least significant bit of OP1 becomes the most significant bit (value of OP2 will be ignored)
111X	(dark)	none	reserved for future use

Table 1: Arithmetic/Logic Operations Supported by Calculator

Individual Project Specification

Number in the attendance list	User Interface Variant	Logic/Arithmetic Operations to Implement
1	A	Add, Multiply, Logical NOT, Logical Ex-OR
2	B	Sub, Divide, Logical AND, Rotate Left
3	A	Remainder of Division, Binary Logarithm, Logical OR, Rotate Right
4	B	Add, Square, Logical NOT, Logical Ex-OR
5	A	Sub, Square Root, Logical AND, Rotate Left
6	B	Multiply, Binary Logarithm, Logical OR, Rotate Right
7	A	Add, Divide, Logical NOT, Logical Ex-OR
8	B	Sub, Remainder of Division, Logical AND, Rotate Left
9	A	Square, Binary Logarithm, Logical OR, Rotate Right
10	B	Add, Square Root, Logical NOT, Logical Ex-OR
11	A	Sub, Multiply, Logical AND, Rotate Left
12	B	Divide, Binary Logarithm, Logical OR, Rotate Right
13	A	Add, Remainder of Division, Logical NOT, Logical Ex-OR
14	B	Sub, Square, Logical AND, Rotate Left
15	A	Square Root, Binary Logarithm, Logical OR, Rotate Right
16	B	Add, Multiply, Logical NOT, Logical Ex-OR
17	A	Sub, Divide, Logical AND, Rotate Left
18	B	Remainder of Division, Binary Logarithm, Logical OR, Rotate Right
19	A	Add, Square, Logical NOT, Logical Ex-OR
20	B	Sub, Square Root, Logical AND, Rotate Left
21	A	Multiply, Binary Logarithm, Logical OR, Rotate Right
22	B	Add, Divide, Logical NOT, Logical Ex-OR
23	A	Sub, Remainder of Division, Logical AND, Rotate Left
24	B	Square, Binary Logarithm, Logical OR, Rotate Right
25	A	Add, Square Root, Logical NOT, Logical Ex-OR
26	B	Sub, Multiply, Logical AND, Rotate Left
27	A	Divide, Rotate Right, Binary Logarithm, Logical OR
28	B	Add, Remainder of Division, Logical NOT, Logical Ex-OR
29	A	Sub, Square, Logical AND, Rotate Left
30	B	Square Root, Binary Logarithm, Logical OR, Rotate Right

Table 2: Individual Project Specification

Design Specification

Figure 3 shows a block diagram of the calculator.

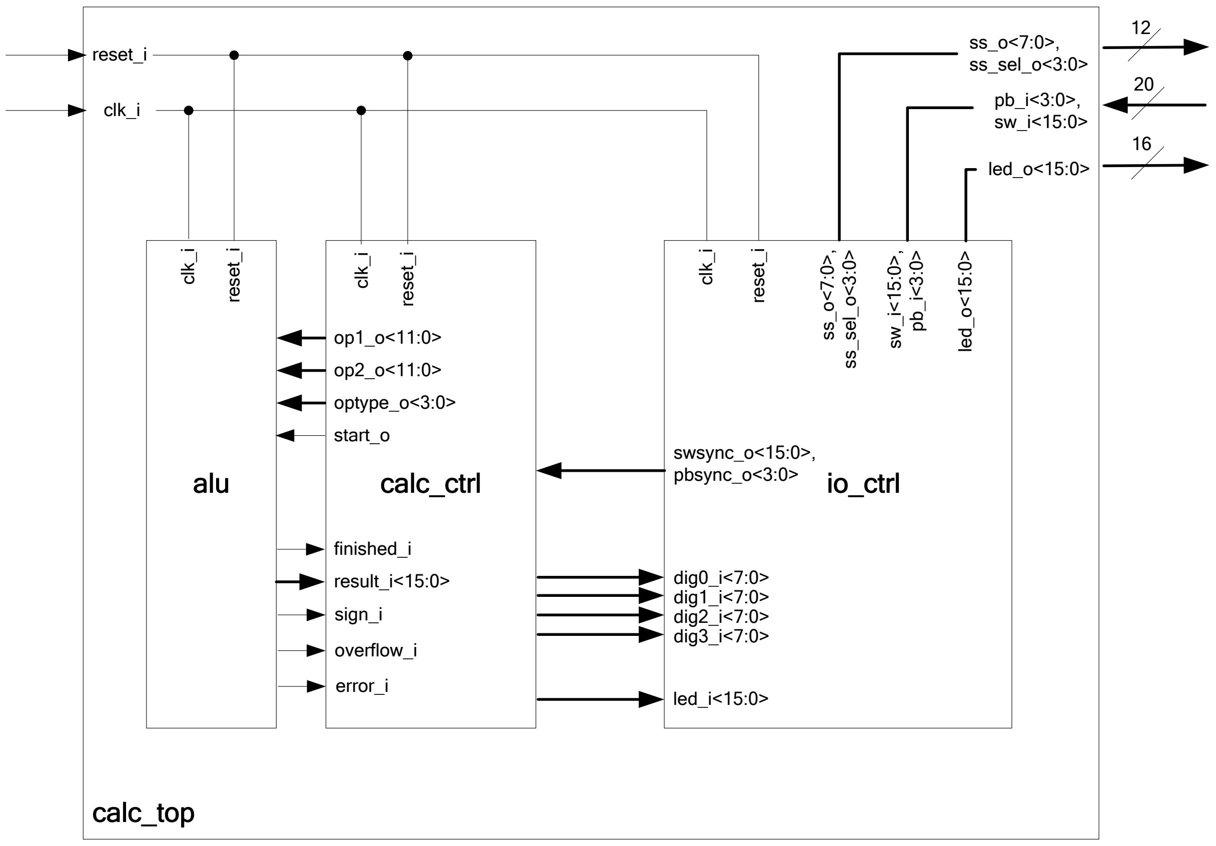

Figure 3: Block Diagram of the Calculator

The top-level module "calc_top" of the calculator consists of three sub-units named "io_ctrl", "calc_ctrl" and "alu" which are described in the following. Table 3 shows all I/Os of the top-level module "calc_top".

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset (connected to push button BTNU)
sw_i(15:0)	In	Connected to 16 switches SW0-SW15
pb_i(3:0)	In	Connected to the 4 push buttons BTNL, BTNC, BNTR and BTND
ss_o(7:0)	Out	Contain the value for all four 7-segment digits (including the decimal point)
ss_sel_o(3:0)	Out	Select one out of four 7-segment digits
led_o(15:0)	Out	Connected to 16 LEDs

Table 3: Top-level I/Os of the Calculator

The IO control unit “io_ctrl” controls all I/O ports (except the clock and the reset signal). It includes the multiplexer needed for the 7-segment digits, debounces the switches and push buttons and makes the debounced signals available for FPGA-internal logic on swsync_o(15:0) and pbsync_o(3:0). The IO control unit implements a generic interface for the I/O hardware of the Basys3 board. This means, that not all I/Os are used for the calculator project. For example, some of the LEDs are unused (see the distance learning letter “IO Control Unit” for details). Table *4* shows all ports of the IO control unit.

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
dig0_i(7:0)	In	State of 7 segments and decimal point of Digit 0 (from FPGA-internal logic)
dig1_i(7:0)	In	State of 7 segments and decimal point of Digit 1 (from FPGA-internal logic)
dig2_i(7:0)	In	State of 7 segments and decimal point of Digit 2 (from FPGA-internal logic)
dig3_i(7:0)	In	State of 7 segments and decimal point of Digit 3 (from FPGA-internal logic)
led_i(15:0)	In	State of 16 LEDs (from FPGA-internal logic)
sw_i(15:0)	In	State of 16 switches (from FPGA board)
pb_i(3:0)	In	State of 4 push buttons (from FPGA board)
ss_o(7:0)	Out	To 7-segment display of the FPGA board
ss_sel_o(3:0)	Out	Selection of a 7-segment digit on the FPGA board
led_o(15:0)	Out	Connected to 16 LEDs of the FPGA board
swsync_o(15:0)	Out	State of 16 debounced switches (to FPGA-internal logic)
pbsync_o(3:0)	Out	State of 4 debounced push buttons (to FPGA-internal logic)

Table 4: Ports of the IO Control Unit

The calculator control sub-unit “calc_ctrl” is the main control unit of the calculator. It provides the two operands OP1 and OP2 as well as the type of arithmetic/logic operation to the ALU (Arithmetic Logic Unit) and starts processing of an operation via signal “start_o”. Once the calculation is finished (indicated by the ALU via signal “finished_i”) the result of the calculation and the sign bit as well as special conditions like error or overflow will be evaluated and forwarded to the IO control unit. Table 5 shows the ports of the calculator control unit while implementation details can be found in the distance learning letter “Calculator Control Unit”.

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
swsync_i(15:0)	In	State of 16 debounced switches (from IO control unit)
pbsync_i(3:0)	In	State of 4 debounced push buttons (from IO control unit)
finished_i	In	ALU indicates that calculation of an arithmetic/logic operation is finished
result_i(15:0)	In	16-bit result of an arithmetic/logic operation coming from the ALU
sign_i	In	Sign bit of the result (0 = positive, 1 = negative)
overflow_i	In	Indicates that the result of an operation exceeds 16 bits
error_i	In	Indicates that an error occurred during processing of the operation
op1_o(11:0)	Out	Operand OP1 for the ALU
op2_o(11:0)	Out	Operand OP2 for the ALU
optype_o(3:0)	Out	Defines the type of arithmetic/logic operation for the ALU (see Table 1)
start_o	Out	Instructs the ALU to start a new calculation
dig0_o(7:0)	Out	State of 7 segments and decimal point of Digit 0 (to IO control unit)
dig1_o(7:0)	Out	State of 7 segments and decimal point of Digit 1 (to IO control unit)
dig2_o(7:0)	Out	State of 7 segments and decimal point of Digit 2 (to IO control unit)
dig3_o(7:0)	Out	State of 7 segments and decimal point of Digit 3 (to IO control unit)
led_o(15:0)	Out	State of 16 LEDs (to IO control unit)

Table 5: Ports of the Calculator Control Unit

The arithmetic logic unit “alu” processes the selected arithmetic/logic operation. Table 6 shows the ports of the ALU, implementation details can be found in the distance learning letter “Arithmetic Logic Unit”.

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
op1_i(11:0)	In	Operand OP1 from the Calculator Control Unit
op2_i(11:0)	In	Operand OP2 from the Calculator Control Unit
optype_i(3:0)	In	Type of arithmetic/logic operation (see Table 1) from the Calculator Control Unit
start_i	In	The Calculator Control Unit instructs the ALU to start a new calculation
finished_o	Out	ALU indicates that calculation of an arithmetic/logic operation has finished
result_o(15:0)	Out	16-bit result of an arithmetic/logic operation for the Calculator Control Unit
sign_o	Out	Sign bit of the result (0 = positive, 1 = negative)
overflow_o	Out	Indicates that the result of an operation exceeds 16 bits
error_o	Out	Indicates that an error occurred during processing of the operation

Table 6: Ports of the ALU

The pin-out for the FPGA is shown in Table 7.

Port Name	Pin	Feature
clk_i	W5	System Clock
reset_i	T18	Button BTNU
ss_sel_o(0) … ss_sel_o(3)		
ss_o(0) … ss_o(7)		
sw_i(0) … sw_i(15)		
pb_i(0) … pb_i(3)		
led_o(0) … led_o(15)		

Table 7: Pinout of the FPGA

The missing information in Table 7 can be found in the schematics of the FPGA board (see CIS website of the lecture, link “FPGA Board Documentation”). The pin identifiers (W5, T18 …) are also printed on the Basys3 board.

How to Proceed?

The next steps in the project are as follows:

- Read the available documentation (the remaining five distance learning letters). Walk through the documentation in the following order: Distance learning letter “Part 2: IO Control Unit”, “Part 3: Calculator Control Unit”, “Part 4: Arithmetic Logic Unit”, “Part 5: Top-level Design” and “Part 6: Synthesis & Implementation”.
- Create a directory structure for the project, similar to the structure of the “Full Adder Example” (see distance learning letter “Introduction to ModelSim-Intel FPGA Starter Edition”) which includes folders for the VHDL code of the design (“vhdl”), the testbenches (“tb”), the simulation scripts (“sim”) and the Xilinx Vivado project (“impl”).
- Break down each unit into smaller sub-blocks and decide which of these blocks need to be coded as combinatorial logic and which of them need storage elements (registers).
- Write VHDL code for each sub-unit and verify all sub-units stand-alone in a simulation (using testbenches). Moreover, verfiy the whole design in a top-level simulation.
- After you have verified your design in a simulation, start with synthesis and implementation using Xilinx Vivado. Have a look at all warnings and fix them in the VHDL code! The goal is a “first-time-right design” which means that your design runs on the first attempt without the need for any bugfixing.

Project Presentation

Please prepare the following things for the presentation of your project in the lab:

- Demonstration of the functionality of the calculator project on the FPGA board
  - The design must implement 100% of the specified functionality. Otherwise, you will obtain zero points!
- Commented and compilable VHDL testbenches for your FPGA design:
  - If you cannot present a simulation at least at the top-level of your design, you will obtain zero points since this (i) reflects a poor design style and (ii) indicates that you have not really developed the design by yourself!
- Commented and synthesizable VHDL code of your FPGA design:
  - If your design is not compilable/synthesizable, you will obtain zero points.
  - Note that the number of points that can be obtained depends heavily on the quality of your code and project structure, e.g., the application of a clean and structured coding style, naming conventions for signals and files, quality of testbenches and simulation setup, number of warnings reported by Xilinx Vivado as well as the quality of comments in the source code (you will lose points for both undercommented and overcommented code!).
  - Finally, the grading of your project depends on how well you can answer the questions of the instructor during the presentation. It is assumed that every student understands 100% of the source code and the design flow, even If the project was developed by a group of students (this changes from year to year).

Please upload a ZIP file of your project (which should contain VHDL files only!) after the presentation via the CIS website of this course. Moreover, return the Basys3 board to the course supervisors. Note, that you cannot finish the course before you did not upload the design and return the board!

Version

Version 1.0	Update to entire document for CHIP1

If you find mistakes or inconsistencies, please report them to the course supervisors via email. Thank you!