FH University of Applied Sciences TECHNIKUM WIEN

Lecture Notes

Specification of Counter Project
Part 1: Overview

Version: BELN1.0
Author: R. Höller, P. Rössler

Copyright Notice

Introduction

As a final project, a simple up/down-counter will be implemented during the last distance learning phase of this course. Details of the project are described in five specification documents: This document as well as another four lecture notes named “Counter Unit”, “IO Control Unit”, “Top-Level Design” and “Synthesis & Implementation”, see course website for details.

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 counter using four 7-segement digits shall be implemented for the Digilent Basys3 FPGA development board. The used FPGA is an AMD/Xilinx Artix-7 (XC7A35) device.
- Four of the 16 switches located on the Basys3 board will control the counter as shown in Table 1. The state of the four switches is reflected on four LEDs.
- The counting mode (decimal, hexadecimal, or octal) and the counting frequency of the least significant digit (0.5 Hz, 1 Hz, 10 Hz or 100 Hz) depends on your matriculation number as shown in Table 2 (if you work in groups, use the number of the group member with the lowest number).

SW0 (Run/Stop)	SW1 (Up)	SW2 (Down)	SW3 (Clear)	Funktion
X	X	X	1	Counter shall be cleared to `0000` (“soft reset”)
1	1	X	0	Counter counts up. If the counter reaches the end value (`9999` in case of a decimal counter, `7777` in case of an octal counter, and `FFFF` in case of a hexadecimal counter), the digits shall switch to `0000` and count up from the beginning (`0001`, `0002`, `0003`, …) again.
1	0	1	0	Counter counts down. If the counter reaches `0000`, the digits shall switch to the end value (`9999` in case of a decimal counter, `7777` in case of an octal counter and `FFFF` in case of a hexadecimal counter) and count down again (e.g. `9998`, `9997`, `9996`, … in case of a decimal counter).
others				Counter holds the actual counting value

Table 1: Functional description of the switches

- The design shall operate synchronously with the on-board 100 MHz clock.
- An asynchronous high-active reset shall be used to initialize the design (BTNC button on the Basys3 board).

Individual Counter Specification

The counting mode (decimal, hexadecimal, or octal) and the counting frequency of the least significant digit (0.5 Hz, 1 Hz, 10 Hz or 100 Hz) depend on the two least significant digits of your matriculation number, as shown in Table 2 (if you work in groups, use the number of the group member with the lowest number).

Example: If your matriculation number is 2146776**7**, then you have to implement a decimal counter with a frequency of the least significant digit which is 10 Hz.

Note, that the supervisors will check if your implemented counter matches to your matriculation number according to Table 2. If you implement a counter with a mode/frequency that is different to the definition shown in Table 2, your project **cannot be accepted by the supervisors!**

Two least significant digits of your matriculation number	Counter mode	Counter frequency of the least significant digit [Hz]
01, 13, 25, 37, 49, 61, 73, 85, 97	Decimal	0.5
02, 14, 26, 38, 50, 62, 74, 86, 98	Octal	0.5
03, 15, 27, 39, 51, 63, 75, 87, 99	Hexadecimal	0.5
04, 16, 28, 40, 52, 64, 76, 88	Decimal	1
05, 17, 29, 41, 53, 65, 77, 89	Octal	1
06, 18, 30, 42, 54, 66, 78, 90	Hexadecimal	1
07, 19, 31, 43, 55, 67, 79, 91	Decimal	10
08, 20, 32, 44, 56, 68, 80, 92	Octal	10
09, 21, 33, 45, 57, 69, 81, 93	Hexadecimal	10
10, 22, 34, 46, 58, 70, 82, 94	Decimal	100
11, 23, 35, 47, 59, 71, 83, 95	Octal	100
12, 24, 36, 48, 60, 72, 84, 96	Hexadecimal	100

Table 2: Individual counter specification

Design Specification

The top-level VHDL entity `cntr_top` contains all the I/Os as specified in Table 3.

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
sw_i(15:0)	In	16 switches
pb_i(3:0)	In	4 push buttons
ss_o(7:0)	Out	Contains the value for all four 7-segment digits
ss_sel_o(3:0)	Out	Selects one out of the four 7-segment digits
led_o(15:0)	Out	16 LEDs

Table 3: Top level I/O ports

Description: The top-level entity `cntr_top` contains of two sub-units: `io_ctrl` and `cntr`. Sub-unit "IO control" `io_ctrl` handles the I/O ports (with exception of the clock signal and the reset signal), and the second sub-unit `cntr` contains the implementation of the four decimal/hexadecimal/octal counters. The 7-segment decoder/multiplexer (used to control and multiplex the four 7-segment digits) is located in the `io_ctrl` unit.

The IO control sub-unit has the following I/O ports:

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
cntr0_i(n:0)	In	Digit 0 (from FPGA-internal logic)
cntr1_i(n:0)	In	Digit 1 (from FPGA-internal logic)
cntr2_i(n:0)	In	Digit 2 (from FPGA-internal logic)
cntr3_i(n:0)	In	Digit 3 (from FPGA-internal logic)
led_i(15:0)	In	State of 16 LEDs (from FPGA-internal logic)
sw_i(15:0)	In	16 switches (from FPGA board)
pb_i(3:0)	In	4 push buttons (from FPGA board)
ss_o(7:0)	Out	to 7-segment digits of the FPGA board
ss_sel_o(3:0)	Out	Selection of a 7-segment digit
swsync_o(15:0)	Out	16 switches (to FPGA-internal logic)
pbsync_o(3:0)	Out	4 push buttons (to FPGA-internal logic)
led_o(15:0)	Out	to 16 LEDs of the FPGA board

Table 4: I/O ports of the IO control unit

Description: The `io_ctrl` unit implements a generic interface for the I/O hardware contained on the Basys3 board. This means, that not all I/Os are used for the counter project (in detail, only

4 switches, 4 LEDs and none of the four push buttons will be used here). The unit debounces the signals coming from the switches and push buttons of the board and makes them available for FPGA-internal logic on the signals swsync_o(15:0) and pbsync_o(3:0). It also decodes the values cntr0_i(n:0), cntr1_i(n:0), cntr2_i(n:0) and cntr3_i(n:0) coming from the counter sub-unit and maintains the 7-segement digits via signals ss_o(7:0) and ss_sel(7:0).

The counter sub-unit has the following I/O ports:

Port Name	Direction	Description
clk_i	In	System clock (100 MHz)
reset_i	In	Asynchronous high active reset
cntrup_i	In	Counts up if signal is '1'
cntrdown_i	In	Counts down if signal is '1'
cntrclear_i	In	Sets counter to 0x0 if signal is '1'
cntrhold_i	In	Holds count value if signal is '1'
cntr0_o(n:0)	Out	Digit 0 (from FPGA-internal logic)
cntr1_o(n:0)	Out	Digit 1 (from FPGA-internal logic)
cntr2_o(n:0)	Out	Digit 2 (from FPGA-internal logic)
cntr3_o(n:0)	Out	Digit 3 (from FPGA-internal logic)

Table 5: I/O ports of the counter unit

Description: The `cntr` sub-unit contains implementation of the decimal/hexadecimal/octal counter. By using the *cntrclear_i* signal, the counter can be synchronously set to `0000`. The signal *cntrhold_i* makes it possible to hold the current counting value. The signals *cntrup_i* and *cntrdown_i* control whether the counter counts up or down. In order to define the behavior when multiple control inputs are logic high, a priority scheme shall be implemented (see Table 1). The signals cntr0_o(n:0), cntr1_o(n:0), cntr2_o(n:0) and cntr3_o(n:0) are connected to the `io_ctrl` sub-unit which controls the 7-segment displays.

The pinout for the FPGA is as follows (the missing pin numbers can be found in the board schematics and are also printed on the board).

Port Name	Pin	Feature
clk_i	W5	System Clock
reset_i	U18	Button BTNC
ss_sel_o(0) … ss_sel_o(3)		
ss_o(0) … ss_o(7)		
led_o(0) … led_o(15)		
sw_i(0) … sw_i(15)		
pb_i(0) … pb_i(3)		

Table 6: Pinout of the FPGA

How to Proceed?

The next steps in the project are as follows:

- Read the available documentation (the remaining four distance learning letters). Walk through the documentation in the following order: Distance learning letter “Part 2: IO Control Unit”, “Part 3: Counter Unit”, “Part 4: Top-level Design” and “Part 5: 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 (“msim”) and the AMD/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 the correct behavior of all sub-units, stand-alone in a simulation (using testbenches). Moreover, test the whole design in a top-level simulation.
- After you have verified your design in a simulation, start with synthesis and implementation using AMD/Xilinx Vivado (see distance learning letter “Introduction to AMD/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 works at 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 counter 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 AMD/Xilinx Vivado as well as the quality of comments in the source code (you will lose points for both undercommented and overcommented code – simply stick to what you have already learned in the previous units of the course).

- Finally, the grading of your project depends on how well you can answer the questions of the supervisors 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.
- Note, that the supervisors will check if your implemented counter matches to the specification according to Table 2. If you have implemented a counter with a mode/frequency that is different to the definition shown in Table 2, your project **cannot be accepted by the supervisors!**
- Also note that, if two or more groups submit the same VHDL code, all of these students will be graded with **zero points** for the final project!
- There is, however, no need to create any PowerPoint slides for the project presentation.

Please upload a ZIP file of your project (which should contain VHDL design and testbench files as well as ModelSim do-files only!) via the Moodle course website by the given deadline. 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! **Also note, that non-timely returning the board as well as not responding to emails or phone calls concerning board return can lead to legal actions and being excluded from the study program due to termination of your contract with the FH Technikum Wien!**

Version

Version 1.0	Minor modifications of document CHIP1 1.0

If you find bugs or inconsistencies in this document, please report them to the course supervisors via email. Thank you!