Grading criteria of System Modelling and Synthesis Exercises

(UPDATED 18.10.2023)

- E1, E2, and E3 are basic experiments exercising fundamental modelling tasks. You should finish E1, E2, and E3 with correct results to pass the course.
  a. E1 is pass/fail.
  b. E2 and E3 grading scheme can be found at the end of this document.
  c. Note, If you only finish the exercises E1, E2, and E3, your final grade of the course will be 1.
- E3 has two tasks. Only finishing Task 1 will be at maximum graded 3. To get the highest grade of 5, you should finish the whole experiment with a correct result.
- E4 has 7 tasks. By finishing the first 5 Tasks you will get a grade of 3 (If your submission has only the behavioural simulation with the LED Driver, then your grade should be below 3). To get a grade of 4 or 5, you should finish Tasks 6 and 7 with correct results.
- E5 is a “step-by-step" experiment. You will get 0 or 5 on this assignment. Without answering the questions, your grade will be 3.
- Except for E5, to demonstrate an exercise, you need to:
  a. explain your code and simulations (either to TAs in person or as a video recording included in the .zip file submission),

b. demonstrate board results to TAs in person
c. submit the source codes (including all relevant files in your Vivado project) to the return folder in Moodle **on time**.

*Late submission is allowed for the first Exercise. However, in the later exercises, late submission will be penalized with **-1** for the calculation of your grade. Re-submission is also allowed after the deadline, but for each attempt, your re-calculated grade will have **-0.5** penalty.*

*Contribution to the final grade:

Exercise	1	2	3	4	5
Weight	10%	20%	25%	35%	10%

Grading scheme of Exercise 2

Specifications	**points**
If reset = '1' then count <= '0'	**+0.5**
If enable = '1' and reset = load = '0' then count <= count ∓ '1' elsif reset = load = '0' then count <= count	**+0.5**
If up/down = '0' then count <= count + '1' else count <= count – '1'	**+0.5**
If (up/down = '0' and count = '0xf') or (up/down = '1' and count = '0x0') then over <= '1' else over <= '0'	**+0.5**
If load = '1' then count = data	**+0.5**
2nd counter with correct behaviours	**+ 1**
Testbench	**+ 1**
Zedboard implementation	**+ 1**
Violations	
Late submission after the deadline	**- 1**
Re-submission after the deadline (*N), where N is the number of re-submissions	**- 0.5 * N**

Latches	**- 0.5**
Counter's sequential logic is not correct: e.g., update outputs value every 2nd rising_edge; update outputs on both clock edges, counter skips, counting in the wrong direction, counting up/down by 2 units	**-0.5**
Grade	Minimum(round_up(**points**),5)

Grading scheme of Exercise 3

Specifications	**points**
Task 1 (architecture in two processes)	

Combinatorial Process, similar to Ex2, with a few modifications:
count = [count_msb] & [count_lsb] (two outputs);
data = 8-bit (one input array);
if (up/down = 0 and count = 0xff) then
over = '1'; count = 0x00;
if (up/down = 1 and count = 0x00) then
over <= '1'; count = 0xff;
	+ 1

Clock Divider Process: Outputs a divided clock signal using a integer counter with a comparator, either implemented as the entity's generic list or as input logic vector.	**+ 1**
Testbench covers all important behaviours	**+ 1**
Task 2 (PWM output)	
PWM module maps the counter's outputs to the duty-cycle of its output signal (by comparing counter output to a hard-coded immediate value or PWM's generated sequential counter)	**+1**
demonstrate PWM duty cycle control with Zedboard LED brightness	**+1**

Penalties	
Late submission after the deadline	**- 1**
Re-submission after the deadline (*N), N is the number of **re-submissions**	**- 0.5 * N**
Inferred Latches	**- 0.5**
Counter's sequential logic is not correct: e.g., update outputs value every 2nd rising_edge; update outputs on both (rising and falling) clock edges, counting in the wrong direction, counting up/down by 2 every rising clock event	**-0.5**
Grade	Minimum(round_up (**points**),5)

Grading scheme of Exercise 4

Specifications	**points**
Task 1 (FSM diagram)	
Shows all states names and their transitions	**+ 1**
Optional bonus*: Shows inputs and outputs from each state (Moore) or state-transition (Mealy) (e.g., using a K-map state table)	**+ 0.25**
Task 2 (controller and timer)	
Timer entity has std_logic_vector (1 downto 0) as input to control timer duration	**+ 0.5**
State iteration control with std_logic input: Iteration_mode = '0': 1 → 6 → 2 → 3 → 7 → 1 Iteration_mode = '1': 7 → 3 → 2 → 6 → 7	**+ 0.5**
Reset switch to trigger STANDBY (state 4)	
Testbench (up to Task 2)	**+ 0.5**
Task 3 (generic timer)	
The timer uses comparator that it receives from entity's generic list (e.g., millisec,1/10 sec)	**+ 0.5**
Task 4 (Alarm substate)	
An additional generic/clocked timer for 7 seconds	**+ 0.25**
Alarm state as substate to the controller, triggered by a switch's falling-edge	**+ 0.25**

Task 5 (Implement up to Task 4 on Zedboard)	**+ 0.25**
Task 6 & 7 (Replace input switches with buttons debounce)	
Button debounce handling	**+ 1**
Replace input switches with buttons push	**+ 0.5**
Penalties	
Late submission after the deadline	**- 1**
Re-submission after the deadline (*N), N is the number of **re-submissions**	**- 0.5 * N**
Inferred Latches	**- 0.5**
FSM is sensitive to both rising and falling clk event	**- 0.5**
Grade	Minimum(round_up (**points**),5)

Exercise 5 Grading scheme

Specifications	**points**
Task 1 (Testbench code)	
Initialize new value for matrix A and matrix B to and give correct computation of their product	**+ 1**
Verify with correct result (including screenshot of debug program pointer at the Error captured)	**+ 0.5**
Verify with wrong result	**+ 0.5**
Task 2 (controller and timer)	
Report Estimated clock period	**+ 0.5**
Report Worst case latency	
Report Number of DSPs, FFs, LUTs	**+ 0.5**
Task 3 (short essay 4-10 lines)	
Discuss benefits, drawbacks and challenges of programming FPGA with VHDL, compared with High-Level Synthesis programming with C++.	**+ 1**
Find parts in generated VHDL code that are readable, and unreadable.	**+ 1**
Penalties	
Late submission after the deadline	**- 1**
Re-submission after the deadline (*N)	**- 0.5 * N**
Grade	Min(round (**points**),5)