2-point Homework 3

Use the last 3 digits of your personal number (PN) to select which question you have to answer. Modulo divide your 3 digits with the number of questions, in this case, 2, plus 1. The resulting number is the question that you should answer.

!!! (PN mod 2) + 1 !!!

For example, if your PN ends in 730, then (730 mod 2) + 1 = 1, meaning you must answer question 1.

Make sure you comment on the top of each file you submit, your PN, name and the question you are answering.

!!! IMPORTANT !!!

If you answer the wrong question, your submission will be invalid.

We can call you to explain your solution. If you cannot explain your answer, the homework will be invalid!

KTH has a zero-tolerance policy against cheating.

https://www.kth.se/en/student/stod/studier/fusk-1.997287

If you have questions or need clarifications, you can ask in the discussion forum in Canvas.

QUESTIONS

1.1 QUESTION 1

Design the constrains that will generate objects of the class student defined below.

```
class Student;
    string studentId;
    int unsigned year;
    string class;
    string sport;
    string listOfCourses[$];
    bit leftHanded;
endclass;
```

The ID of the student should be a string of 10 random characters.

The year of the student should be between 1 and 12.

If the year is between 1 and 6, the student can be in class A, B, C or D. If the year is over 6, only in A, B or C.

If the student is between year 1 and 6 the sports they can chose are football or basketball. If the student is above 6th year they can choose football, basketball, tennis, and chess.

The list of courses should be an array of 5 courses if the student is between year 1 and 6, and 7 courses otherwise. The possible courses (regardless of the year), with equal probablilty are: maths, science, english, history, phyisical education, language, computer science, arts, music, economics.

1.2 QUESTION 2

Consider a digital circuit that implements a simple communication protocol. The protocol is defined as follows:

1. Two devices are present, master and slave
2. The master initiates all transactions by setting the signal “active” to 1
3. The slave confirms that it is ready to transmit/receive by setting the signal “ready” to 1
4. If the slave does not become ready in 5 clock, cycles the master should go idle and request the transaction again.
5. The direction of the communication is decided by the master (**before** setting the active flag) by setting the “send” signal to 1 if the data will flow master->slave or 0 for opposite.
6. The transaction ends when the master sets “active” back to 0, then the slave should immediately set “ready” to 0 in the next cycle

Implement a set of assertions that will check the following:

1. When active signal **or** ready signal are 0, data should be high-z
2. If the active signal becomes 1 for 5 cycles without ready becoming 1, active should be 0 by the 6^{th} cycle.
3. Ready signal should never become 1 if active is 0.
4. When active signal becomes 0 (after a transaction), ready should become 0 **exactly** in the next cycle

Note: you don’t need to implement the master, slave, or bus. Only the assertions.