|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Document no. |  | | | | | | | TFE4171 – Design of Digital Systems 2 |
| TFE4171-Ex4-01 |
| Revision | | Date | | | | |
| 1.0 | | 13.03.2015 | | | | |
| Revision By | |  | | | | |
| Hope / Grindheim | |  | | | | |
| ­ | | | | | | | |
| Laboratory Report | | | | | |  | |
| Exercise IV | | | | | |  |  |
|  | | | | | | |
| ­ | | | | | | | | |
| Author(s):  B.Hope  S.Grindheim | |  | |  |  | | | |
| Revision Notes | * 1.0: Initial release | | | |
|  |  | | | |
|  |  | | | |
|  |  | | | |
|  |  | | | |
| ***Bengt Erik Hope / Svein Grindheim © 2015 Bergen, NORWAY*** | | | | | | | | |

**🗎Page intentionally left blank**

* Abstract

*This document attempts to answer the questions as listed in the fourth exercise in the NTNU course TFE4171 – Design of Digital Systems 2.*

Contents

[Assignment 5 1](#_Toc414584824)

[**Task 5.1** 1](#_Toc414584825)

[**Task 5.2** 1](#_Toc414584826)

[**Task 5.3** 1](#_Toc414584827)

[**Task 5.4** 2](#_Toc414584828)

[**Task 5.5** 2](#_Toc414584829)

[**Task 5.6** 4](#_Toc414584830)

[**Task 5.7** 5](#_Toc414584831)

[Assignment 6 6](#_Toc414584832)

[**Task 6.1** 6](#_Toc414584833)

[Appendix A 8](#_Toc414584834)

# 

## 

Below follows a state diagram for the main controller in the serial receiver:

Figure 1

## 

Below is the property reset – with its corresponding assertion and sequence, which proves that right after applying the reset sequence the controller is in state IDLE and the counter is reset to 0.



Code Excerpt 1

## 

Below is the property called stay\_in\_idle – with its corresponding assertion, which proves that the controller stays in state IDLE if no start bit arrives.



Code Excerpt 2

## 

Below is the property read\_byte – with its corresponding assertion, which proves that a byte is transmitted when applicable and that the controller returns to IDLE after transmission.



Code Excerpt 3

## 

We now restrict the OneSpin tool to use only the basic IPC solver. This will make OneSpin generate a counterexample[[1]](#footnote-1) to the read\_byte property assertion. This is a false negative – the counterexample is “spurious”. The debugger shows us a situation which could never happen. In this case we see *cnt\_s* starting at 1 and *cnt\_en* asserted high when the controller is IDLE. In order to fix this we must add some reachability constraints.

1. By writing the following sequence and adding it to the cause/assumption part in our property we try to prove the property again:

 Code Excerpt 4

Obviously, the property cannot be proven because the counter is enabled when the controller is idling in the counterexample.

1. We now write an additional sequence called\_in\_idle\_counter\_not\_enabled:

 Code Excerpt 5

We can now prove the read\_byte property!

1. We already prove the correctness of the output behavior!

## [[2]](#footnote-2)

In this task we will prove the reachability constraint in\_idle\_counter\_not\_enabled, which was introduced in the previous task, is invariant. We utilize two inductive proofs – called in\_idle\_counter\_not\_enabled\_\_step and in\_idle\_counter\_not\_enabled\_\_base, to show that the reachability constraint is fulfilled in all states reachable from the initial state. These proofs are defined by the following properties:



Code Excerpt 6

## [[3]](#footnote-3)

In this task we continue the work in the previous task and introduce two new properties, as shown below. We then assert the properties we have written as illustrated from line 108.



Code Excerpt

As one can see from the code above: In order to prove the induction step we strengthened the inductive proof by considering more than one time frame – where n>=1, in order to prove the properties.

# 

In this assignment we will check the completeness of our property set written in the last assignment. We expand the properties as necessary if any shortcomings are detected. The code listings referred to can be found in **Appendix A**.

## 

Before running tests we had give the completeness checker information about the sequencing of operations, signals central to the operations and the inputs the operations depend on. We did this as follows in the file **readserial.gfv:**



When running all tests as specified in the assignment text we discovered several issues which we had to fix. This led us to add the following too our property definitions (additions in bold font):







Contents of readserial.gfv



 Contents of readserial.tda

**Specific determination test fixes:**

* + explicitly check that *valid* is low in the assumption part of read\_byte (readserial.tda; line 66) and stay\_in\_idle (readserial.tda; line 52)

**Specific reset test fixes:**

* + explicitly define input and register states when entering reset (readserial.tda; line 38 through 42)

**Specific Case Split test fixes:**

* + define follow

**Successor test fixes:**

* + explicitly define input and register states when entering reset (readserial.tda; line 38 through 42)

1. …had to struggle a bit to find this error as I would have put […] t**##**0 cnt\_s **==** 3'b0 **and t##**0 cnt\_en **==** 1'b0 **implies** […] directly in the cause part – which would make the property pass when asserting it in both configurations [↑](#footnote-ref-1)
2. There is a typo in the exercise text where Task 5.6 is number as Task 5.5 [↑](#footnote-ref-2)
3. There is a typo in the exercise text where Task 5.7 is number as Task 5.6 [↑](#footnote-ref-3)