

**Faculty of Engineering & Technology**

**Electrical & Computer Engineering Department**

**Verification and Validation of Hardware - ENCS5399 -**

**Assignment #1**

**Counter-Controlled Loop Unit (CCLU) Verification Plan**

**Prepared by:**

Zainab Shaabneh\_1182820

**Instructor**:

Dr. Ayman Hroub.

**Date:**

11 \ 1 \ 2023

**Table of** **Contents**

[**Create Verification Plan** 3](#_Toc124372167)

[**1)** **Description of the verification levels** 3](#_Toc124372168)

[**2)** **Functions to be verified** 3](#_Toc124372169)

[**3)** **Resources required (people) and schedule details** 3](#_Toc124372170)

[**4)** **Required tools** 3](#_Toc124372171)

[**5)** **Specific tests and methods** 3](#_Toc124372172)

[**6)** **Completion criteria** 3](#_Toc124372173)

[**7)** **Test scenarios** 4](#_Toc124372174)

# **Create Verification Plan**

## **Description of the verification levels**

* Since we have only one unit we can start from block level.

## **Functions to be verified**

* Number in counter greater than zero.
* Check if the address exist in the top of the stack or not.
* Check the stack is full or not.
* Check if the input target address equal the output target address [if there is no error].
* Check operation to deal with the stack (push, pop and extend in memory).

## **Resources required (people) and schedule details**

* One to two engineers.

## **Required tools**

* Simulation engines
* Debuggers
* Verification environment authoring tools

## **Specific tests and methods**

* Level of Observability 🡪 Black box.
* Verification strategy 🡪 Random & Deterministic.
* Abstraction level 🡪 Block level.
* Checking strategy 🡪 Input and output pins.

## **Completion criteria**

* Coverage targets.

## **Test scenarios**

|  |  |
| --- | --- |
| Test # | Description |
| 1.1 | Check the operation of each command |
| 1.2 | Check the stack operation |
| 1.3 | Try to induce errors in the input signals |
| 1.4 | Check the reset pin |
| 1.5 | Ensure the timing issues |

|  |  |  |
| --- | --- | --- |
| Topic | Test # | Description |
| Operation  of each command | 1.1.1 | Check that input opcode is 32 bit |
| 1.1.2 | Check if the address is valid and the counter is valid ( less than 216) |
| 1.1.3 | Check if the command was 00 then it will ignore the input opcode |
| 1.1.4 | Check if the command was 00 and the stack has 15 element then isFull signal must be low. |
| 1.1.5 | Check if the command was 01 and the stack is empty then it will push on the stack at the top of the stack and edit the counter |
| 1.1.6 | Check that the output target address will be as the input target address when there is no errors |
| 1.1.7 | Check if the command was 01 and the stack has 15 element to see if it will make isFull output equal 1 in the next clock cycle |
| 1.1.8 | Check if the command was 01 and the stack has 16 element it should not accept this data and assert the error signal |
| 1.1.9 | Check if the command was 10 and the stack is empty then there is an error |
| 1.1.10 | Check if the command was 11 then assert the error output |
| Check  stack is full  or not | 1.2.1 | Check if the command was 01 and the stack has some element then it will check if the stack is full or not before doing any operation |
| 1.2.2 | Check if the top element of the stack is popped when the counter equal 1 |
| 1.2.3 | Check if the command was 01 and address input is the same as the address at the top element of the stack then it will decrement the counter of the top element of the stack |
| 1.2.4 | Check if the command was 10 then it will pop the top element of the stack |
| 1.2.5 | Check if the command was 00 then it should not push the data into the stack even if the stack was {empty, has some data or full) |
| Check different errors | 1.3.1 | Check the valid output signal |
| 1.3.2 | Check that when stack is full then the error signal and the isFull will be high |
| 1.3.3 | Check that valid output should not be high when the reset equal 1 (the error then must equal 1) |
| Reset pin | 1.4.1 | Test the reset signal when the stack is empty |
| 1.4.2 | Test the reset signal when the data is halfway in |
| 1.4.3 | Test the reset signal when the stack is full |
| Timing | 1.5.1 | Check the time that is needed to get the output when there is no error occurred |
| 1.5.2 | Check the time that is needed to get valid signal when there is no error occurred |
| 1.5.3 | Check the time that is needed to get isFull signal {when the stack has 16 elements it should be high in the next clock cycle} |
| 1.5.4 | Check the time that is needed to get error signal {when error occurred then error signal should be high in the same clock cycle} |
| 1.5.5 | Check the time that is needed to get the stack empty when the reset pin equal 1 {reset pin need 1 clock cycle to make the stack empty} |
| Corner cases | 2.1 | Check if the input counter equals 0 |
| 2.2 | Check if the input counter equals 1 |
| 2.3 | Put the command 01 and the counter 2 to see if it will pop this element in the next clock cycle {since it will be 1} |