|  |
| --- |
|  |
| CO-DMA PROJECT  Meng Final Report - In Partnership with Infineon |
| |  |  |  | | --- | --- | --- | | Oliver Anderson | Academic Year: | Electronic Systems Engineering Meng | | Student No. 189215639 | 2022-2023 | Supervisor: Peter Wilson | |

Text

Description automatically generated

# Executive summary

Summary and project recap

# acknoledgements

Todo: Thank you to Infineon and Peter Wilson

Table of Contents

[Executive summary 1](#_Toc130371685)

[acknoledgements 2](#_Toc130371686)

[I. Introduction 4](#_Toc130371687)

[II. Methods and Results 5](#_Toc130371688)

[II.1 Definition of Modules & Packages 5](#_Toc130371689)

[II.2 READ & WRITE MACHINES 5](#_Toc130371690)

[III. Discussion Conclusions & Project Review 6](#_Toc130371691)

[References 7](#_Toc130371692)

[Appendices 8](#_Toc130371693)

# Introduction

* You should set out the context of the work with a clear statement of its aims.
* You may also introduce the layout of the report.
* You should concisely review the relevant literature and any past work and link this to your project aims.
* You should describe any background theory, analysis techniques, design tools that you will rely on.

# Methods and Results

* You should describe how you achieved the project objectives in sufficient detail that the work could be repeated by someone else, perhaps working in a different University.
* You should include photographs of any equipment used to provide evidence of use.
* You should present your results in graphical and tabular form providing appropriate numerical and statistical analysis where appropriate.

This section of the report will break the codma down into the main building blocks before explaining how they connect together to achieve the goals of the module. To improve readability and accelerate the development of the hardware, the codma is broken down into modules and packages.

The codma uses finite state machines to determine the actions that are happening to the internal register. This is most efficient way of developing hardware of this nature.

## II.1 Definition of Modules & Packages

The codma is comprised of three modules, connected by a fourth top-level module and a package containing the definitions for the states of the finite state machines.

**NOTE ON NEED READ AND NEED WRITE SIGNALS**

## II.2 READ & WRITE MACHINES

The read and write machines handle the internal registers that are then used to drive the bus interface at the top level. The two machines are separate to allow for queuing a write whilst a read is being performed and vice-versa. This is especially useful for updating the status of the codma in the memory location specified by the status pointer. As the codma is usually required to perform a read before a write, the read-machine takes priority should both a read and write be requested on the same clock cycle.

Though the read and write machines are contained within one SystemVerilog file, the bus interface is only driven in the top-level module. This is to ensure there are no contentions which could cause problems for full synthesis and is a design rule violation.

Both machines are broken down into three operation states and an un-used state so that should an error happen in hardware causing the hardware to enter an otherwise un-defined state it can resolve itself preventing a lock-up. The three used states are described below:

#### Idle

This state is the usual state for the read and write machines, where it will wait for an internal flag to be asserted before moving to the ask state.

#### Ask

This state is entered once the internal flag is asserted designating the need for a read or write. In this state, the bus interface will be connected to either the read or write address, size and data registers depending on the demand. The bus interface signal requesting a read or write will also be asserted.

#### Granted

Once the grant signal has been received from the bus interface, the read or write machine will move to the granted state, de-asserting the read or write request and the internal flag.

This phase is considered complete once the word count has reached the specified limit. The word count is a fixed number, depending on the size set for the bus interface size wire. This counter in incremented for each rising clock edge where the data valid signal is asserted, or each new data packet written whilst the write valid is being asserted. As there is no confirmation from the bus interface confirming receipt of the data packet, the data is written sequentially and each clock pulse. However, the specification explicitly said this assumption could not be made for the data read and there may be a delay between every data-word.

#### unused

The unused state is defined so should there be a hardware fault that causes the machine to enter an otherwise undefined state, it will return to IDLE and not lockup. This could be furthered through the use of Assertions, however the simulation tool in use did not support this.  **Maybe it should make the dma go to the error state ?**

## II.3 CODMA Machine

The codma machine module is the main module that describes the operation of the codma, using the read write and CRC machines to complete tasks. The states of the codma machine can be broken down as follows:

#### DMA IDLE

The DMA IDLE state is the state in which the codma will perform a final write to the status pointer and wait for the next start signal from the CPU. Once the start signal has been received, the dma will assert the need read signal to launch the read machine and move to the dma pending state.

#### DMA PENDING

The codma enters the dma pending state when the start signal has been received and it is performing an initial read of the memory address given by the status pointer. Once this read is complete, the codma will have all the required information to perform the task and therefore will move to the dma data read state, or the dma CRC state.

#### DMA TASK READ

This state is reserved for the task type 2. In this task type, the dma will perform a first task and then read the memory address above the status pointer form the first task. This is the move data and perform linked task operation of the codma as per the specification document.

#### DMA DATA READ

In this state, the dma will perform a read at the source address and store the data, ready to write it to the destination address. This state is complete once the read machine returns to Idle.

#### DMA WRITING

In this state, the dma will assert a need write internal flag, forcing the write machine into the write ask state. This will allow the codma to write the stored data from the source address to the destination address. Once the write machine returns to Idle, the byte length of the task is reduced by the number of bytes that have been copied to the new address. If this number is now zero, the codma will move to the idle state and update the value at the status pointer address or the dma task read state if the task was a type 2. If this number is not zero, the codma will return to the dma data read state.

#### dma crc

The dma CRC state is called when the task type 3 is selected. This will cause the compute CRC module to be used.

#### dma error

#### dma unused

Akin to the read and write state machines, the dma machine had an undefined state. It was therefore defined as an unused state so the codma would not lockup if this state was entered. If this state is entered, the dma will enter the dma error state to report a hardware error.

## II. 4 Compute CRC Module

As computing the CRC code was a complex task, it was developed as a standalone module. **It can then be connected at the top level and use the state of the DMA and internal flags to operate when required.**

## II.5 Evaluation & Testing

### Proof of Operation

The testbench […]

#### Task 0

#### Task 1

#### Task 2

#### Task 3

The CRC module required a different approach to develop.

### Bug Fixing

Write machine and changing the way the data words are counted. This improved performance from 2288ps to 2012ps the tb to run. This is a 12% improvement over the whole testbench and general codma operation. For a single word write this can improve things to 50% and a four-word cycle 20%.

The initial setup counted the number of different data packets on a read, however this caused problems where two identical words one after the other would not be counted as new data packets **[…]**.

## II.6 Pipelining

## II.7 Advanced Features

## II. 8 SYNTHESES

# Discussion Conclusions & Project Review

* You should draw the threads of your project work together to determine the outcomes of your project in relation to the literature and past work.
* You should include a discussion of the uncertainty or incompleteness of any information or results used or gained during the project and how they affect the project outcomes.
* You should draw conclusions describing the overall engineering achievement and evaluating the project’s outcome.
* You should indicate where future project work could develop and enhance your results.
* You should review the operation of your project, identifying what went well and describe the setbacks, and discuss how the project lifecycle could be improved should the project or a similar one be repeated.

# References

# Appendices