ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and
Electrical Engineering

Spring Semester 2025

Ensuring Fault Tolerance in the CROC SoC: A Hardware-Centric Approach to Watchdog Timer Implementation

VLSI 2 Course Project

Miguel Correa and Elio Wanner
cream@ethz.ch, ewanner@ethz.ch

May 20, 2025

Professor: Frank Kagan Gürkaynak, kgf@iis.ee.ethz.ch

Abstract

This report summarizes the work done in the VLSI 2 course project. The project consisted of extending the Croc SoC with a Watchdog Timer.

Contents

1 Introduction **1**
- 1.1 Overview of the Croc SoC and its open-source nature . . . 1
- 1.2 Why a watchdog timer (WDT) was needed . . . 1
- 1.3 Goals and expected improvements . . . 1

2 Related Works **2**
- 2.1 Overview of existing watchdog timer implementation . . . 2
- 2.2 Justification for our approach . . . 2

3 Methodology **3**
- 3.1 Design choices for WDT (e.g., countdown vs. count-up, reset duration) . . . 3
- 3.2 Testing strategy (testbenches, simulation tools) . . . 3
- 3.3 Debugging and iteration process . . . 3

4 Hardware Architecture **4**
- 4.1 Simple Watchdog Timer . . . 4
  - 4.1.1 Hardware implementation . . . 4
  - 4.1.2 Tests and Results . . . 4
  - 4.1.3 Conclusion . . . 4
- 4.2 Watchdog Wrapper . . . 4
  - 4.2.1 Hardware implementation . . . 4
  - 4.2.2 Tests and Results . . . 5

5 Evaluation **6**
- 5.1 Simulation results . . . 6
- 5.2 Performance comparison (goal achieved?) . . . 6
- 5.3 Possible improvements . . . 6

6 Conclusion **7**
- 6.1 Summary of key findings . . . 7
- 6.2 Future work and next steps . . . 7

Chapter 1

Introduction

1.1 Overview of the Croc SoC and its open-source nature

- Picture of Croc SoC
- Quick explanation of its main parts and mostly what we are using (userdomain)
- Mention that it is open-source and the importance of this, but also why it has to be robust for it to be competitive with closed source in real life scenarios

1.2 Why a watchdog timer (WDT) was needed

- Real life scenario as an example of anoying stuck in a loop chip (if possible, true historic reference)
- Explain what is a watchdog timer and how it would prevent this
- From this go on and conclude it was needed on the Croc SoC

1.3 Goals and expected improvements

- Emphasize on the main goals of the watchdog, with bullet points if possible to emphasize even more. THe main goals being: improve robustness while keeping area and throughput as low as possible. (should we add more goals?)

Chapter 2

Related Works

2.1 Overview of existing watchdog timer implementation

- State several possible watchdog implementations, with sources.
- Explain that it is possible to keep it really simple or to complicate it by adding some features, like having a log to know why the watchdog triggered.

2.2 Justification for our approach

Now, explain our choice to keep it small, simple, energy efficient easy to understand and easy to "take over" if it is needed for something else. Explain that the trade off gains in robustness of croc vs complication of watch dog is the best by keeping the watchdog simple.

Chapter 3

Methodology

3.1 Design choices for WDT (e.g., countdown vs. count-up, reset duration)

Explain that we decided to make a simple watchdog, independent from croc to keep it simple. Then, we added a wrapper to be able to connect it to OBI and croc. Our watchdog uses cout-up (was our own choice) and have a 1 cycle reset duration, wich we found enough.

3.2 Testing strategy (testbenches, simulation tools)

At each step we tested our watchdog, from the simple watchdog to the communication with obi and finally the whole connected croc. If we have time, we want to test a real life scenario (maybe a gpio that returns nothing so our program is stuck?).

3.3 Debugging and iteration process

By separating the work in easy, well separated steps it was easy to debbug.

Chapter 4

Hardware Architecture

This chapter describes the hardware architecture of the Watchdog Timer that we implemented in the Croc SoC. In the different sections we talk about different implementations with different features and complexities.

4.1 Simple Watchdog Timer

4.1.1 Hardware implementation

The first implementation of the Watchdog Timer is the simplest one we could imagine. As shown in Figure 4.1, the WDT is composed of very few components. The main component is the counter, which is incremented every clock cycle and implemented as a flip-flop. The counter is connected to a comparator that checks if the counter has reached a certain value. If the counter reaches the value, the comparator will trigger a reset signal that will reset the system. Otherwise, we keep incrementing the counter. As soon as we get a kick signal, the counter is reset to zero.

4.1.2 Tests and Results

Explain how we did the tests (python files to generate stimuli + expected then compare), show figure. Explain that the results were always good at the end.

4.1.3 Conclusion

Explain that thanks to our approach we found that the watchdog works fine, so if a future problem would come it would come from the wrapper.

4.2 Watchdog Wrapper

4.2.1 Hardware implementation

Add a graph of the watchdog wrapper, and maybe a figure of how it connects to the OBI protocol.

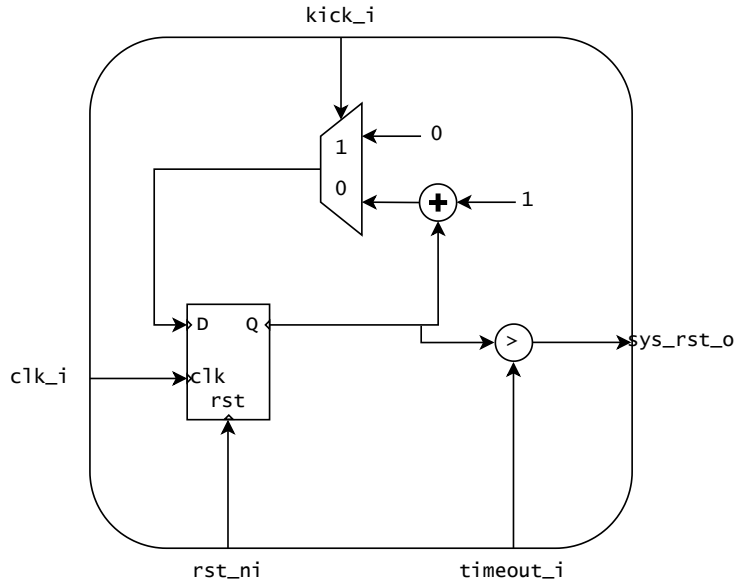

Figure 4.1: Simple Watchdog Timer Architecture

4.2.2 Tests and Results

We checked the functionality of our wrapper by running a helloword.c wich sets the timeout value and sends periodic kicks befor getting stuck in an impossible task. Our watchdog was successfully set thanks to the correctness of the wrapper and it allowed us to reset the program. We also checked the VCD to see if we could find any anormalities, but our wrapper respects OBI protocol :) .

Chapter 5

Evaluation

5.1 Simulation results

Our watchdog passed all our simulations: from the chip ggetting stuck in an infinit loop to TODO FIND OTHERS SIMULATION TESTS

5.2 Performance comparison (goal achieved?)

Our goal was awchieved: we cna successfully say that we managed to improve the reliability and robustness of croc by adding a small area and power. We also managed to store and print our names in a ROM stored in useer domain.

5.3 Possible improvements

A lot of improvements can be done, but all depend of the possible application and of the willingness to trade off simlicity, area, power for something else like debugging ability. TODO: explain more

Chapter 6

Conclusion

6.1 Summary of key findings

6.2 Future work and next steps