UTHM
Universiti Tun Hussein Onn Malaysia

Faculty of Electrical and Electronic Engineering
Semester II Session 2022/2023

Embedded Systems Design
BEJ42203

Section: 2
Group No.: 3
Lecturer: Dr. Mohamad Hairol Bin Jabbar

Report Title: Pollution based Traffic Control
Date: 28 June 2023

Name	Matric . No.	Contributions in the (1) Project Implementation and (2) Report	Digital Signature
Mattis Tom Ritter	JD222 0003	1. Select Components, Project Architecture, Security Implementation 2. Project Architecture, Results	[illegible]
Moritz Hoehnel	JD222 0004	1. Implement IP core, Optimize Performance, Analyze Performance 2. Project Introduction, Project Architecture, Discussion, Conclusion	[illegible]
Muhammad Nazmi Bin Ramli	DE190 204	1. Select Components, Project Architecture, Security Implementation 2. Project Architecture, Results	[illegible]
Hemaraj A/L Ramu	DE190 055	3. Implement IP core, Optimize Performance, Analyze Performance 4. Project Introduction, Project Architecture, Discussion, Conclusion	[illegible]

TABLE OF CONTENTS

1. Project Introduction

The European Environment Agency reports that “[c]ars, vans, trucks and buses produce more than 70 % of the overall greenhouse gas emissions from transport”[1]. Especially in cities, where many vehicles drive, health limits for fine particle matter get exceeded. Governments and cities are now curious to find fast solutions. The most recent is Abu Dhabi, they are starting a project where “AI-techniques will […] provide recommendations to improve traffic light efficiency, thus also reducing congestion and carbon emissions” [2]. Its goal is to reduce stop time at intersections, especially at those having high pollution levels.

This is also the goal of this project. To reach the environment protecting goal faster, a more simple system will be established. This system will be able to measure the fine dust level at a conjunction and act accordingly. A suiting hardware needs to be designed. Additional to the light signal and magnetic loop a standard traffic light already has, it needs sensors for air quality and a display to show the current pollution level. As the ZYNQ 7000 is a good architecture to read all this information, do complex processing and give output to the traffic light and display, it will be used.

The junction of this project has one main road, with one side road crossing it. This project will show how to establish such a traffic control system hardware. Therefore the report will discuss the project objectives, followed by an analysis of the hardware architecture. This is followed by a presentation of the results, a discussion and a conclusion.

2. Project Objectives

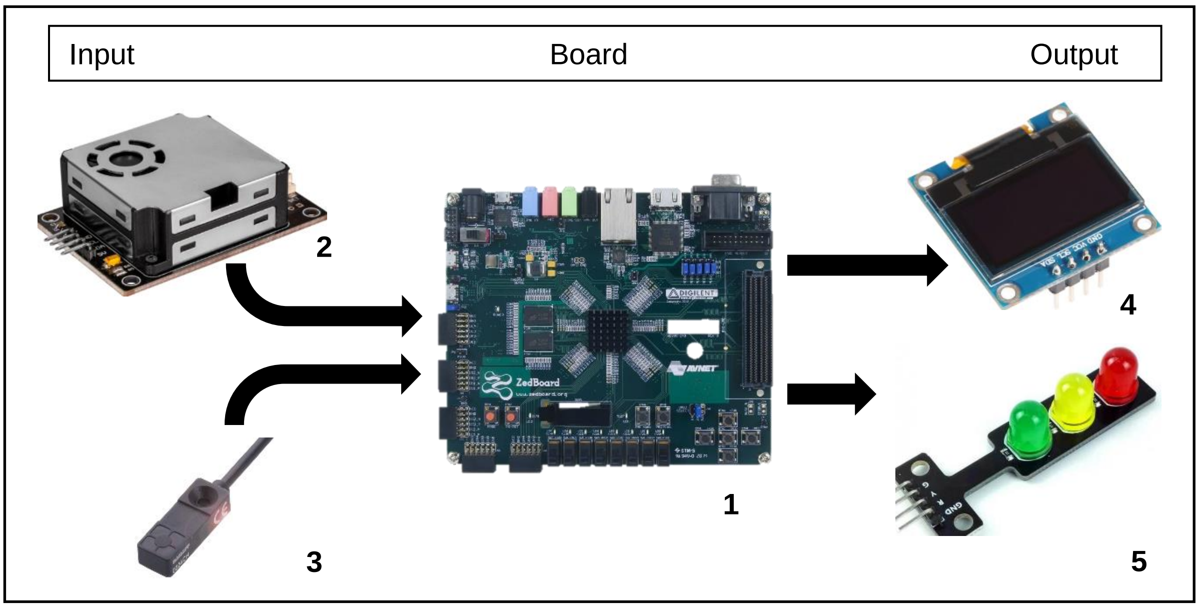
The project objective is to develop a traffic control system using the Zynq-7000 and open-source IP core. As development platform Vivado is chosen. Further specified objectives can be found in the following list:

No.	Description
1.	Analyze which hardware components will be used.
2.	Design the hardware architecture of the traffic control system using the Vivado software. Use open-source IP core. Analyze performance.
3.	Optimize the hardware and compare performance with the hardware state before optimizing.

3. Project Architecture

This chapter will explain the hardware and how it got developed. The project consists of five main components. The ZYNQ 7000, that takes the inputs, makes the calculations and create outputs. There are two types of input sensors. The first one is a fine dust sensor. The fine dust value will determine the pollution of the junction. The higher the fine dust level is, the higher is the pollution. The second sensor type is an inductive sensor. There will be two, each in the side road parts of the junction. As output there are the signals for the display and the traffic lights.

3.1 Architecture Overview



No.	Name	Description	Connection
1	Zynq 7000	Development board	-
2	HM3301	Fine dust sensor	IIC
3	BES R03KC-PSF30B-EP02	Inductive sensor	GPIO digital input (NO)
4	SBC - OLED01	OLED Display	IIC
5	BB-LED-TL-CC	LED traffic light	GPIO for all LEDs

Figure 3.1: Block Diagram with List

Figure 3.1 shows the components of the system, and how they are be connected. The list below gives their name, describe their function and how they are connected to the ZYNQ. The fine dust sensor and the OLED Display are both using the IIC protocol. The inductive sensor provides a digital value, that state if an object has been recognized. As each of the four sides of the junction has a traffic light, the GPIO needs to have suiting outputs for that.

3.2 Hardware blocks

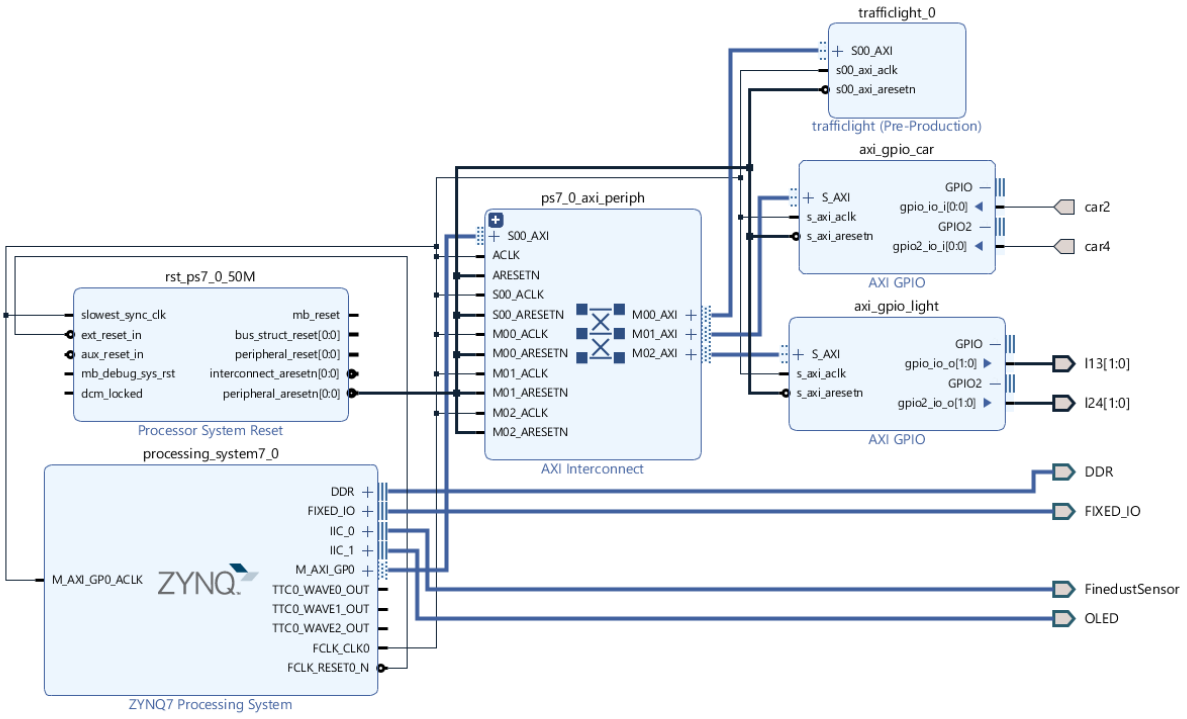
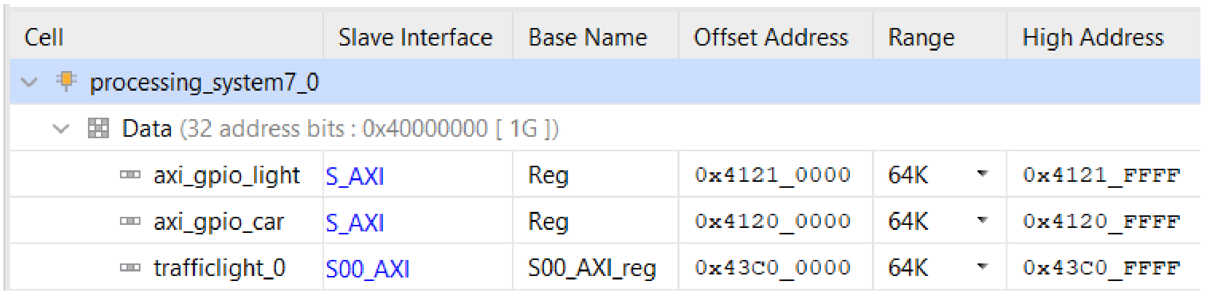


Figure 3.2 Top Level Block Diagram

The top-level architecture of the hardware design is displayed in Figure 3.2. The used blocks are the ZYNQ7 Processing System, a Processor System Reset block, a manager of onboard communication in form of the AXI-Interconnect block, an open-source block that manages the traffic lights and two AXI GPIO blocks, one for the inductive sensors and the other for traffic lights. The fine dust sensor and the OLED display are connected to the IIC 0 and IIC 1 ports of the processing system.

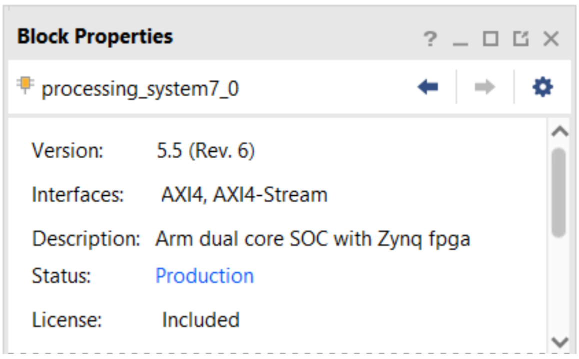


Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_gpio_light	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF
axi_gpio_car	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
trafficlight_0	S00_AXI	S00_AXI_reg	0x43C0_0000	64K	0x43C0_FFFF

Figure 3.3: Addresses

The figure above shows the memory address mapping of the AXI-Blocks. The Offset addresses are 0x4121_0000 for the AXI GPIO that connects the traffic lights, 0x4120_0000 for the AXI GPIO that link the inductive sensors and 0x43c0_0000 for the custom IP block. The high addresses are 0x4121_FFFF, 04120_FFFF and 0x43c0_FFFF respectively. All have a range of 64K.

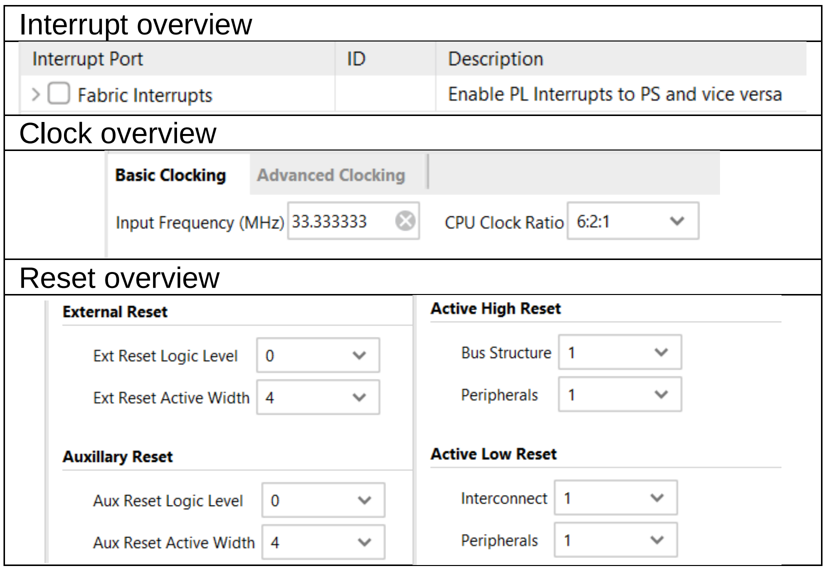
The following paragraphs will describe each of the blocks in detail. The ZYNQ7 Processing System is the first block.


Block Properties
processing_system7_0

Version:	5.5 (Rev. 6)
Interfaces:	AXI4, AXI4-Stream
Description:	Arm dual core SOC with Zynq fpga
Status:	Production
License:	Included

Figure 3.4: ZYNQ7 Processing System Details

The above figure shows that the version 5.5 (Rev. 6) has been used. There is no memory address mapping. The figure below lists the settings of the ZYNQ.


Interrupt overview

Interrupt Port	ID	Description
Fabric Interrupts		Enable PL Interrupts to PS and vice versa

Clock overview

Basic Clocking | Advanced Clocking

Input Frequency (MHz) 33.333333 CPU Clock Ratio 6:2:1

Reset overview

External Reset
Ext Reset Logic Level 0
Ext Reset Active Width 4

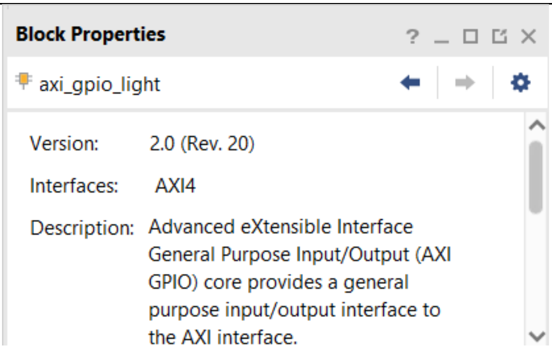
Auxillary Reset
Aux Reset Logic Level 0
Aux Reset Active Width 4

Active High Reset
Bus Structure 1
Peripherals 1

Active Low Reset
Interconnect 1
Peripherals 1

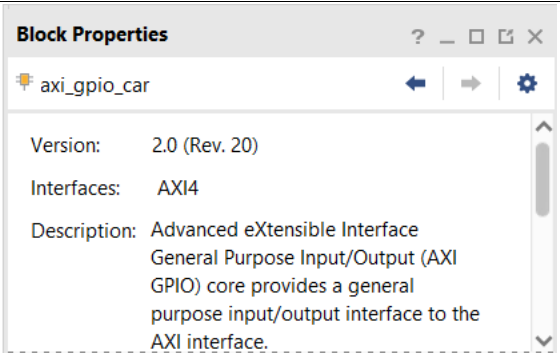
Figure 3.5: ZYNQ Settings

In the settings one can see that the Interrupt Port for fabric interrupts has not been ticked. For the clock an input frequency of 33.3 MHz is set. For the external reset and the auxiliary reset the logic level is “0”. For the active high reset the bus structure and peripherals “1” is selected. Same value is selected for the interconnect and peripheral settings of the active low reset.


Block Properties
axi_gpio_light

Version:	2.0 (Rev. 20)
Interfaces:	AXI4
Description:	Advanced eXtensible Interface General Purpose Input/Output (AXI GPIO) core provides a general purpose input/output interface to the AXI interface.

Figure 3.6: Details AXI GPIO Traffic Lights


Block Properties
axi_gpio_car

Version:	2.0 (Rev. 20)
Interfaces:	AXI4
Description:	Advanced eXtensible Interface General Purpose Input/Output (AXI GPIO) core provides a general purpose input/output interface to the AXI interface.

Figure 3.7: Details AXI GPIO Car Sensor

Figure 3.6 shows the details of the Traffic light interface. The used AXI GPIO has the Version 2.0 (Rev. 20). Figure 3.7 is showing the details of the AXI GPIO, that is used for the inductive sensors. The used IP Core version is 2.0 (Rev. 20).

To use the open-source hardware, a new IP Package needs to be created. The package is created as AXI4 peripheral. The logic is used from [5]. The Variables are as listed in the table below:

Name	Input/Output	Bit(s)
enable	In	1
clk	In	1
reset	In	1
car2	In	1
car4	In	1
state	In	2
nextstate	Out	2
l13	Out	2
l24	Out	2
d1	In	4
d2	In	4
d3	In	4
d4	In	4
setbit	Out	4
q	Out	4
delay	Out	4

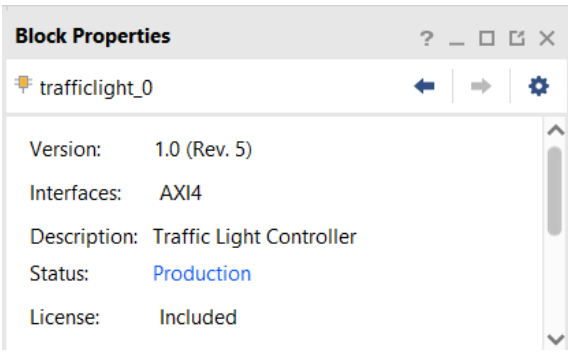
Table 3.1: Ports of the traffic light controller

For creating the registers of the new IP Block, the above listed variables are sorted by being an output or input. The registers and their variables can be seen in the table below:

Connection description	Register	Variables
Input	slv_reg0	enable, state, d1, d2, d3, d4, car2, car4
Output	trafficlight_out	nextstate, setbit, q, delay, l13, l24

Table 3.2: AXI-Peripheral register definition

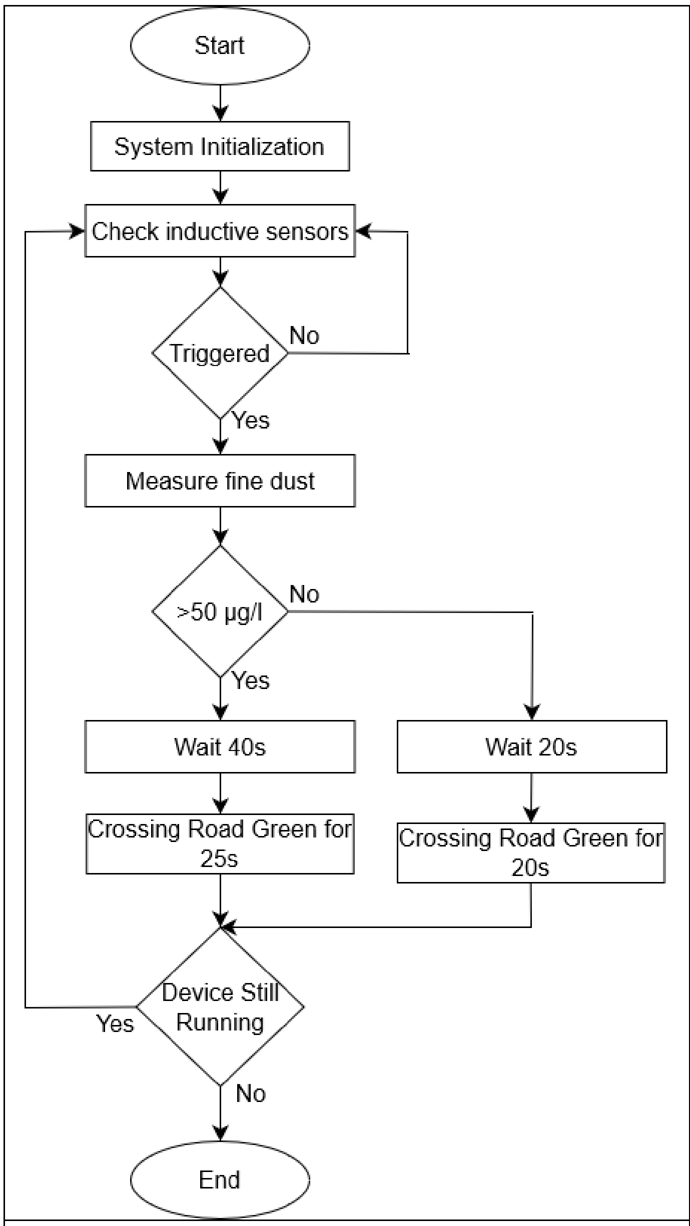
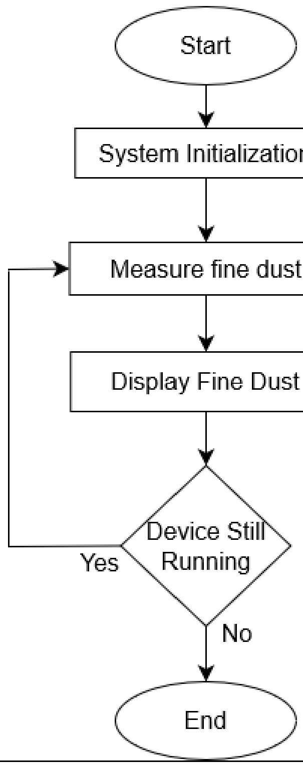
The following figure will explain further details of the block. The used AXI version is 1.0 (Rev.5).



Version: 1.0 (Rev. 5)
Interfaces: AXI4
Description: Traffic Light Controller
Status: Production
License: Included

Figure 3.8: Details AXI Traffic Light Controller

3.3 Operation

 Start System Initialization Check inductive sensors Triggered No Yes Measure fine dust >50 µg/l No Yes Wait 40s Wait 20s Crossing Road Green for 25s Crossing Road Green for 20s Device Still Running Yes No End	 Start System Initialization Measure fine dust Display Fine Dust Device Still Running Yes No End
Figure 3.9: Operation Flow Chart for Traffic Light	Figure 3.10: Operation Flow Chart for Display

The left flow chart shows the modus operandi for the traffic light. After the start the system get initialized. After that the inductive sensors in the side streets check if cars are waiting. If not, it will check again until a car is waiting. If a car is waiting the fine dust is measured. Depending on the fine dust value the switching time of the traffic light is changed. The value 50 µg/l is inspired by [6]. For the city of Stuttgart, a fine dust alert is triggered if this value is exceeded. The penultimate step is to check if the device is still running, if so, the operation checks the inductive sensors again, otherwise the end is reached.

The display is described in a separate flow. The idea is that after starting, the value gets measured, shown and measured again, so there is always the current value visible. The exact flow is like in the figure 3.10.

3.4 Design Flow

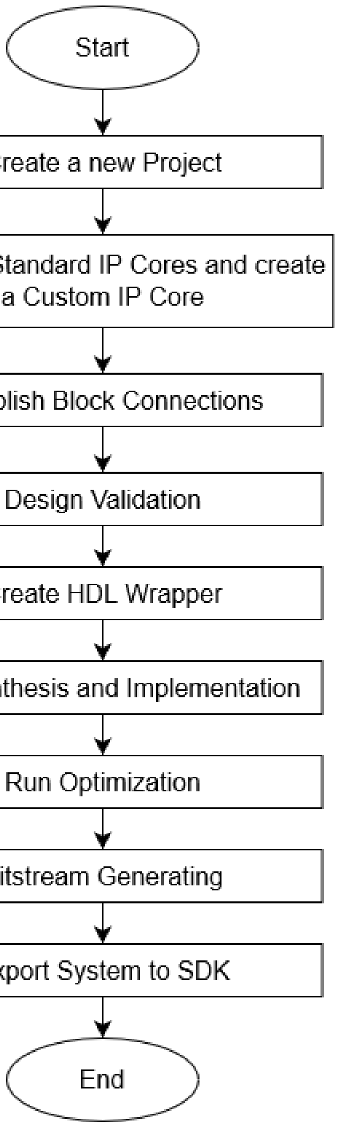
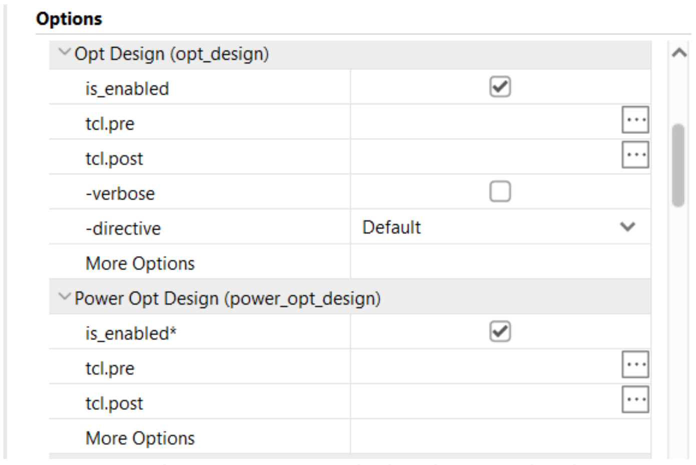


Figure 3.11: Design Flow

The above flow chart explains the design process step by step. After the project start a new project was created in Vivado. In Vivado a block design was created, IP blocks were added. After the open source IP block was downloaded from GitHub, it was integrated into the project by creating a custom IP block.

Then the connections between the blocks were established. As a first step the automation rooting was used. In the second phase connection were correct by hand, especially for custom IP block. Followed by saving and validating the block design. After all errors have been dealt with, the HDL Wrapper creation was performed. Followed by the synthesis and implementation to create the board. This was followed by an analyzation of the power consumption. To have a better performance, the optimization was run. Therefore, in options the power optimized design was enabled. That can be seen in figure 3.12 on the following page. Under “Power Opt Design (power_opt_design)” the box “is_enabled*” is ticked.



Options

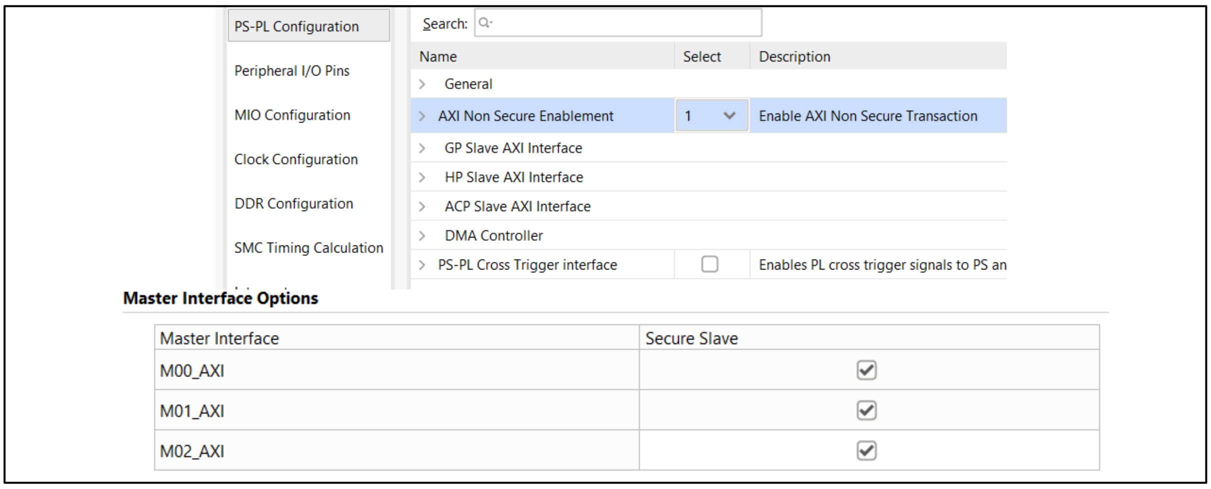
Setting	Value
Opt Design (opt_design)	
is_enabled	✓
tcl.pre	...
tcl.post	...
-verbose	☐
-directive	Default
More Options	
Power Opt Design (power_opt_design)	
is_enabled*	✓
tcl.pre	...
tcl.post	...
More Options	

Figure 3.12: Optimization Method

After ticking the box, the power consumption analysis was run again. The penultimate step was to generate the bitstream. Before the project was finished, it was exported to SDK.

3.5 Sustainability and Security

The sustainability aspect of this project is to optimize traffic control to reduce the pollution of cars at junctions. Therefore, the project is establishing a traffic control by traffic light. This should help to improve the traffic flow and reduce the standing time of cars. To analyze the impact of the standing cars at the junction, a fine-dust sensor is installed. The measured values get shown via a display. This is helping to raise awareness of the pollution. As a final step the traffic light switching time is manipulated by the fine-dust level. To see how it is manipulated exactly look at figure 3.9, which shows the operation flow. The idea is to have longer green periods to have more cars crossing and not interrupt the traffic flow. The green time for the main road is increased a lot more as a high number of cars is expected there. If more cars would need to wait, the pollution rate becomes bigger. Longer waiting times can also encourage drivers to turn off the engine.



Name	Select	Description
General		
AXI Non Secure Enablement	1	Enable AXI Non Secure Transaction
GP Slave AXI Interface		
HP Slave AXI Interface		
ACP Slave AXI Interface		
DMA Controller		
PS-PL Cross Trigger interface	☐	Enables PL cross trigger signals to PS an

Master Interface Options

Master Interface	Secure Slave
M00_AXI	✓
M01_AXI	✓
M02_AXI	✓

Figure 3.13: Security Options

Figure 3.13 is showing the security actions taken. In the upper part of the figure a snippet of ZYNQ7 Processing System settings for “PS-PL Configuration”, the AXI secure transaction, is displayed. The secure transaction has been enabled. This is visualized by showing in the select column “1”. For the second step the AXI Interconnect options are displayed. Under advanced options, one can find the shown table. To establish a secure connection, the boxes “Secure Slave” got ticked for all three AXI Cores.

4. RESULTS

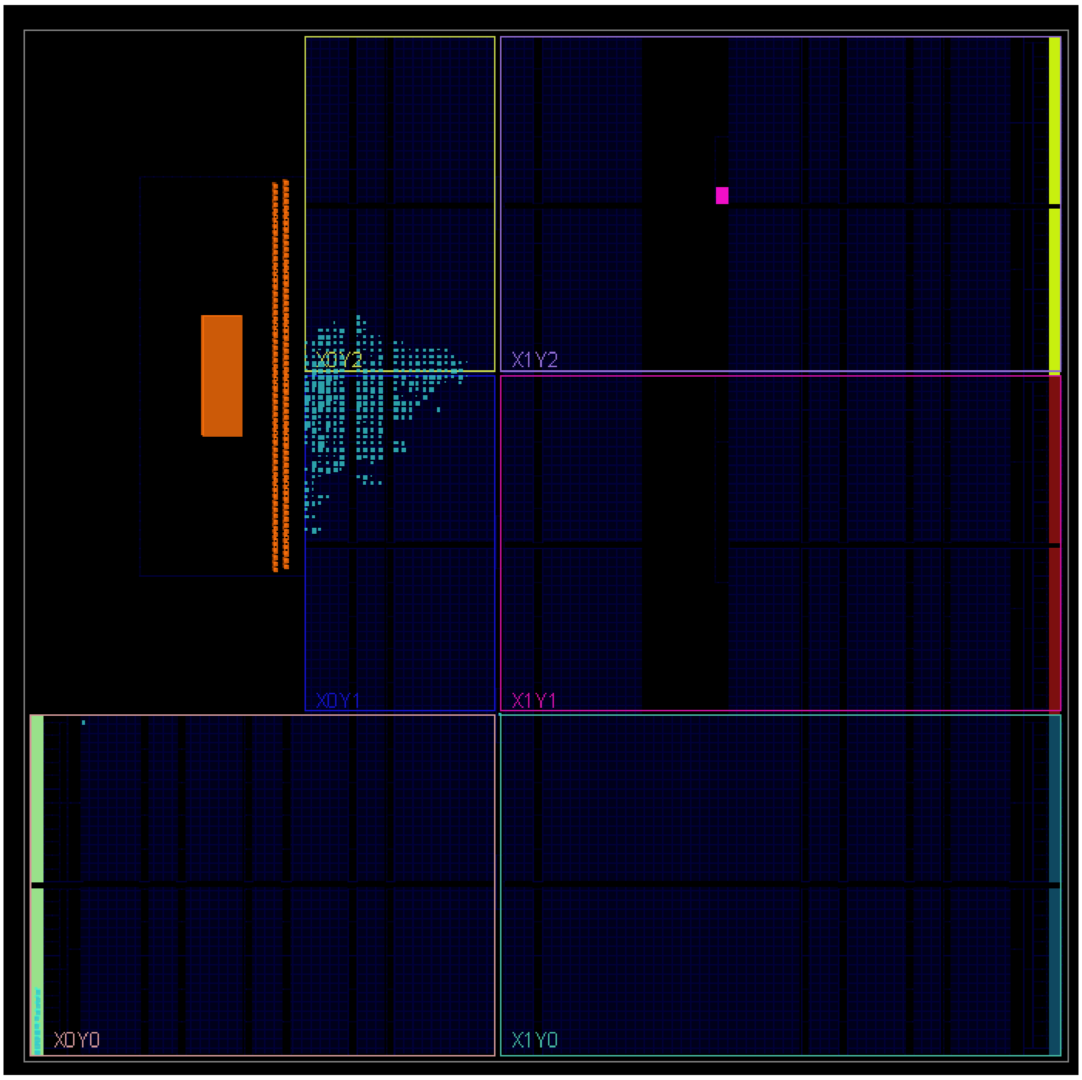


Figure 4.1: Device

Figure 4.1 shows the Layout of the PS and PL for the developed Logic. For PL only little connections got programmed, as the traffic light controller is a smaller project. The connections can be seen in sections X0Y1 and X0Y2.

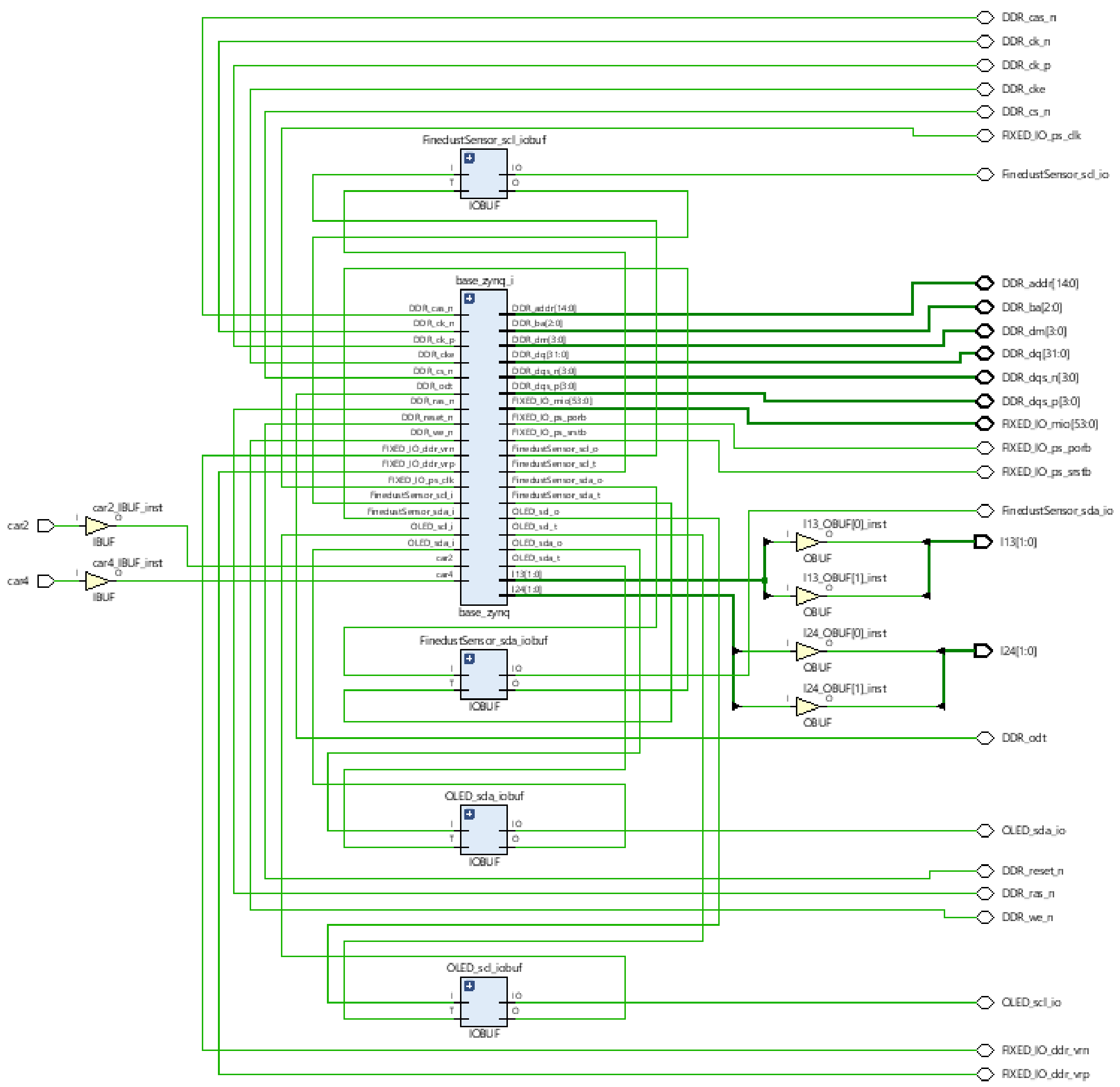
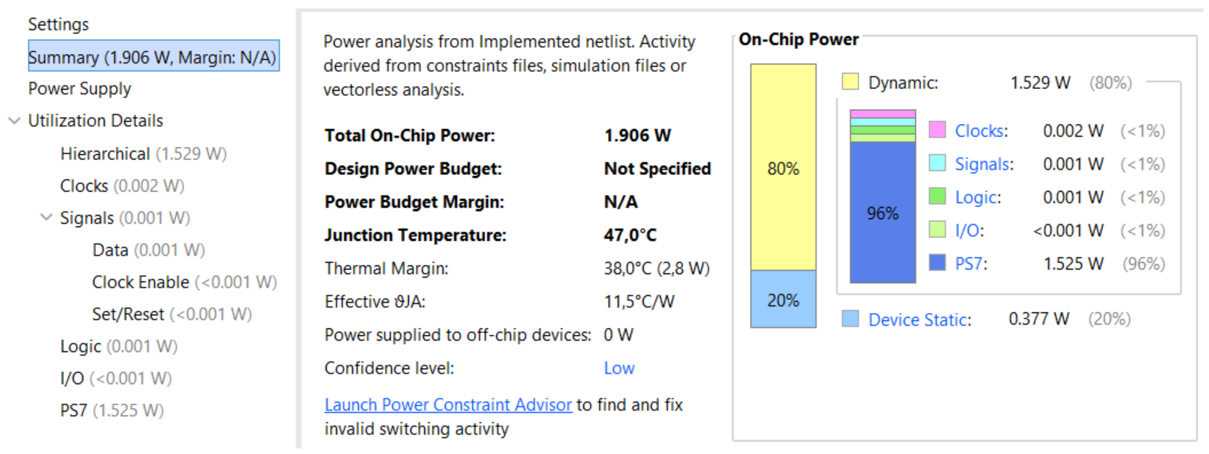


Figure 4.2: Schematic Layout

The above figure shows a top-level schematic layout of the device. One can see the blocks and the connections between them.



Settings
Summary (1.906 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (1.529 W)
Clocks (0.002 W)
Signals (0.001 W)
Data (0.001 W)
Clock Enable (<0.001 W)
Set/Reset (<0.001 W)
Logic (0.001 W)
I/O (<0.001 W)
PS7 (1.525 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.906 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**47,0°C**
Thermal Margin:	38,0°C (2,8 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

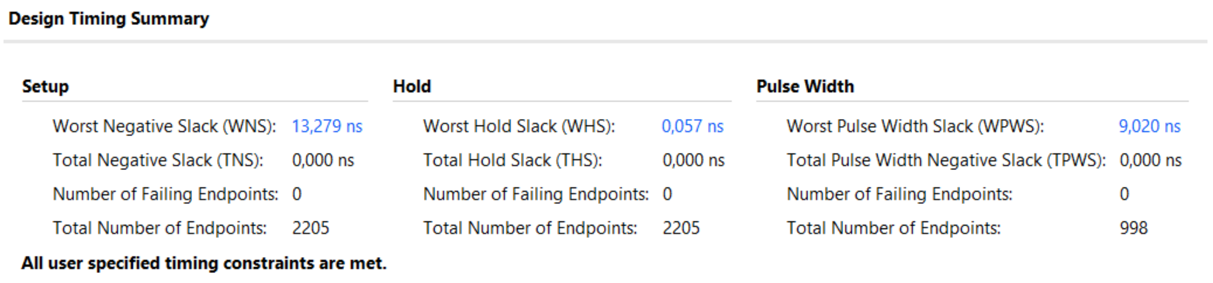
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	1.529 W	(80%)	
Clocks:	0.002 W	(<1%)	
Signals:	0.001 W	(<1%)	
Logic:	0.001 W	(<1%)	
I/O:	<0.001 W	(<1%)	
PS7:	1.525 W	(96%)	
Device Static:	0.377 W	(20%)	

Figure 4.3: Power Consumption Report before Optimization

The above figure shows a summary of the power consumption. The total used power is 1.906W. 80% of this is considered to be dynamic and the rest static.



Design Timing Summary

Setup

Worst Negative Slack (WNS):	13,279 ns
Total Negative Slack (TNS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	2205

Hold

Worst Hold Slack (WHS):	0,057 ns
Total Hold Slack (THS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	2205

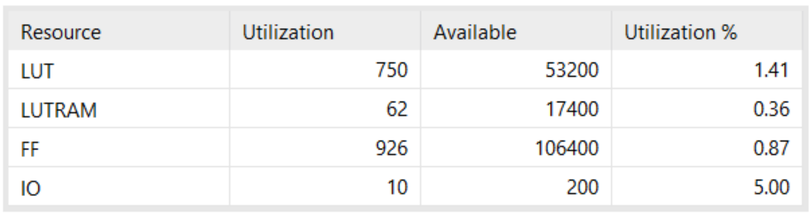
Pulse Width

Worst Pulse Width Slack (WPWS):	9,020 ns
Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	998

All user specified timing constraints are met.

Figure 4.4: Timing Report

The timing report is verifying that all timing constrains are fulfilled.



Resource	Utilization	Available	Utilization %
LUT	750	53200	1.41
LUTRAM	62	17400	0.36
FF	926	106400	0.87
IO	10	200	5.00

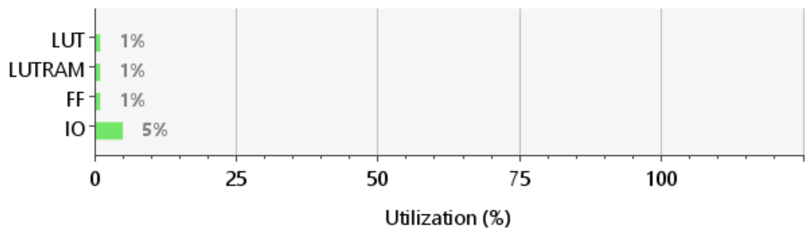
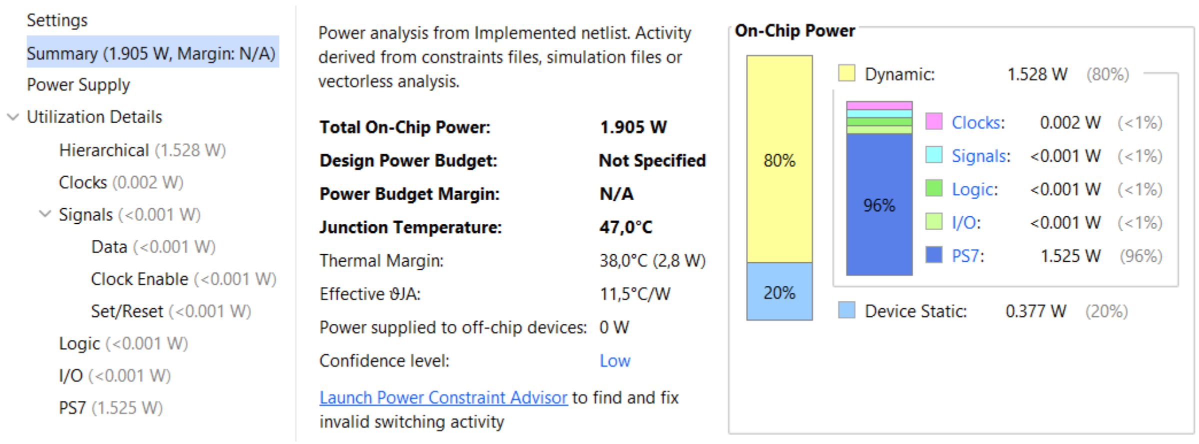


Figure 4.5: Logic Utilization

Figure 4.5 displays the logic utilization of the device. The report says that for LUTs, LUTRAM and flip-flops (FF) about one percent of the available capacity has been used. For the IO it is five percent. This means that there is still of capacity unused. But this fact contributes to a low power consumption.

5. Discussion



Settings
Summary (1.905 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (1.528 W)
Clocks (0.002 W)
Signals (<0.001 W)
Data (<0.001 W)
Clock Enable (<0.001 W)
Set/Reset (<0.001 W)
Logic (<0.001 W)
I/O (<0.001 W)
PS7 (1.525 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.905 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**47,0°C**
Thermal Margin:	38,0°C (2,8 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	1.528 W	(80%)	
	Clocks:	0.002 W	(<1%)
	Signals:	<0.001 W	(<1%)
	Logic:	<0.001 W	(<1%)
	I/O:	<0.001 W	(<1%)
	PS7:	1.525 W	(96%)
Device Static:	0.377 W	(20%)	

Figure 5.1: Power Consumption Report after Optimization

To analyze the power optimization result, values of the total power consumption are compared before and after the optimization as in [8]. Before 1.906 W are needed by the system, now it is down to 1.905 W. This means that the optimization improved consumption by 0.001 W. This results from improvements for Signals and Logic. But otherwise, the two reports look similar. This is mainly because of the fact that this system is not complex, which leaves not much room for improvement. To evaluate this result the power consumption is compared to a CASIO calculator [7]. It is a calculator for educational purposes. The power consumption is stated to be 0.35W. The power consumption of the traffic light controller is about five times greater. Considered that the calculator just performs mathematical task and the fact that the traffic light controller has a more complex logic, the power consumption is considered in order.

The timing summary has also been reviewed after the power optimization. As there are no changes one can refer to figure 4.3. This is good as the power optimization has no harming impact on the timing.

To validate if the project has been done successful, all objectives need to be fulfilled. The first objective was to find all needed components. As all sensors and actuators needed are found this goal is achieved. The second objective was to create the architecture. Furthermore, it was requested to analyze the performance. This was also dealt with and results of this can be looked up in chapter three and four. The last objective was to optimize the performance and compare the result to the state before. As a power optimization has been performed, the performance optimization is considered to be done. The comparison can be found earlier in this chapter. As all objectives are fulfilled, the project has been ended successfully.

6. Conclusion

This project successfully showed a fast way to implement an easy solution for traffic pollution problems. This solution is a low power consummating and secure effort to contribute to our planet’s health. During the development the team showed how to create a Vivado project, implement all IP cores including open-source code, validate, synthesize and optimize ZYNQ hardware.
On long term, the car industry needs to find more fundamental solutions to fight the pollution that comes from cars. The best approach would be to have cars that do not have high CO2 or fine-dust emissions. But also, alternative ways to organize traffic can help. Like establishing more convenient bus connections. An example would be to introduce many small buses, which can have a customized destination that a group of people can share the vehicle. So the total number of cars on the road is reduced.

7. References

[1] European Environment Agency, “Transport”, *eea.europa.eu*, para. 1, Mar. 15, 2023. [Online]. Available: https://www.eea.europa.eu/themes/transport/intro [Accessed: May 25, 2023]

[2] S. Zaman, “Abu Dhabi to use AI-data from Google to reduce traffic congestion”, *Gulf News*, May 26, 2023. [Online]. Available: https://gulfnews.com/uae/abu-dhabi-to-use-ai-data-from-google-to-reduce-traffic-congestion-1.96017551 [Accessed: May 26, 2023]

[3] P. García-Rubio, “Smart traffic lights to reduce air pollution”, *sacyr.com*, Nov. 17, 2022. [Online]. Available: https://www.sacyr.com/en/-/semaforos-inteligentes-para-reducir-la-contaminacion [Accessed: May 26, 2023]

[4] J. Johnson, “Creating a custom IP block in Vivado”, *FPGAdeveloper*, Nov. 11, 2017, [Online]. Available: https://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-in-vivado.html/ [Accessed: Jun. 07, 2023]

[5] R. Ansari, “smart-traffic-light-controller”, *GitHub*, Oct. 2, 2016, [Online]. Available: https://github.com/raash1d/smart-traffic-light-controller/commits/master/src/top.v [Accessed: Jun. 07, 2023]

[6] M. Cloud, “Fine dust: Better, not good”, *ILAWJOURNALS.COM*, Dec. 05, 2018, [Online]. Available: https://www.ilawjournals.com/fine-dust-better-not-good/# [Accessed: Jun. 20, 2023]

[7] Casio Computer Co, “fx-9860GIII, fx-9750GIII, fx-7400GIII Hardware User’s Guide”, *support.caso.com,* 2020. [Online]. Available: https://support.casio.com/storage/en/manual/pdf/EN/004/fx-9860GIII_9750GIII_7400GIII_Hard_EN.pdf [Accessed: Jun. 26, 2023]

[8] Xilinx, “Vivado Design Suite Tutorial Power Analysis and Optimization”, x*ilinx.com*, Dec. 05, 2018. [Online]. Available: https://docs.xilinx.com/v/u/2018.3-English/ug997-vivado-power-analysis-optimization-tutorial [Accessed: Jun. 07, 2023]

[9] Balluff, “Inductive Sensors”, BES R03KC-PSF30B-EP02 datasheet, Feb. 2023

[10] Joy-IT, “0.96 Inch I^{2}C-OLED-Display”, SBC-OLED01 datasheet, Sep. 2019

[11] Joy-IT, “Fine Dust Sensor”, HM3301 datasheet, Jul. 2022