Documentation of Student Project Work:

Recycle Management System

FPGA Programming SS 2021

Date of Submission: July 29th, 2021

Authors of Group A:

Cynthia Levina (621125)

Mohammadzaman Zamani (808247)

Amal Tomy (808241)

Jeet Bhadaniya (810391)

Vipin Koshy Thomas (812684)

Prof.:

Dr. Martin Schramm

Table of Contents

List of Figures

List of Tables

1. Abstract

In this student project work, we propose designing, implementing, and functional verifying a VHDL-based digital circuit for a simple recycle management system.

The recycle management system starts with sorting the materials that have been collected and taken to recycling facilities. Then, the recycling process of sorted waste (shown in Figure 1) was started and carried out using a combinational circuit with key module (push button) to select between different operations/processes based on material type. The materials for this project are categorized into four groups: bio, plastic, metal, and glass. Each material requires different processes or treatments and timing to produce the reusable product/raw material from the waste, for example, fertilizer, plastic granulates, metal bar, and glass cullet. By taking advantage of parallelism, low cost, and low power consumption offered by FPGAs [1], we can track and record the number of sorted input items and the final product or reusable raw material after going through the recycling process on each waste category.

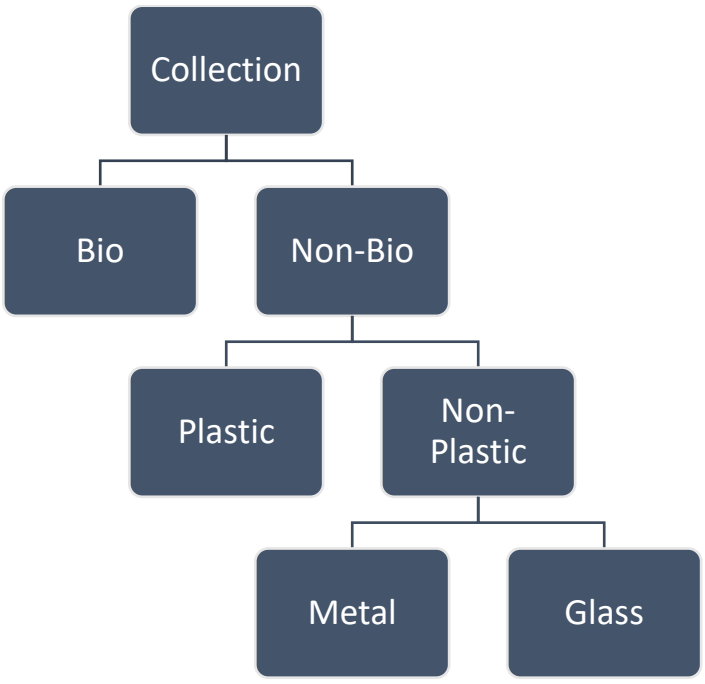


Figure 1. The Sorting process

2. Introduction

2.1. Background

Increasing volumes of waste are being generated as the global population, and living standards rise [2]. People are increasingly concerned about the negative impact of the waste on the environment, limited landfill space, and seeking ways to deal with the problem. The 3R (Reduce-Reuse-Recycle) concept is one of the ways to deal with the problem or at least minimize the impact of waste on the environment, which individuals or even significant communities can implement. It shows a sequence of steps on managing waste properly so that the waste material gets a second chance before disposing them to landfill [3]. The production of waste should be reduced from the beginning as much as possible. Then see if the waste of material can be reuse somehow, for example, making handicrafts. Finally, when materials can no longer be used again, the waste is recycled [3].

The recycling process is done differently depending on the category of the waste. It could be managed by the government or even a company that has enough budget to create and purchase the machines to support the recycling management system by themselves. In this project, we propose designing, implementing, and functional verifying a VHDL-based digital circuit for a simple recycle management system for a conceptional company. The company is assumed to produce four categories of waste: bio, glass, metal, and plastic, which should be recycled.

2.1.1. Biowaste recycling process

Bio-waste, also known as biodegradable waste, consists mainly of organic materials that can be recycled by converting them into compost through the operation shown in Figure 2 [4]. Compost is used as organic fertilizer or replaces peat in potting soil and plant substrates.

Figure 2. The recycling process for biowaste

2.1.2. Plastic-waste recycling process

There are many different steps to recycle plastic waste, one of them shown in Figure 3. The first step is to wash the plastic to remove some of the impurities that can impede the operation, for example, product labels and adhesives and dirt and food residue. The plastic is

then fed into shredders or grinders, which break it down into much smaller pieces, unlike formed plastic products. The plastic will then be heated to be formed with the intended plastic granulate shape, which will be then cooled down before it is ready to be reused for other applications. Additionally, the resized plastic pieces can be used for other applications without further processing, such as an additive within asphalt or sold as a raw material [5].

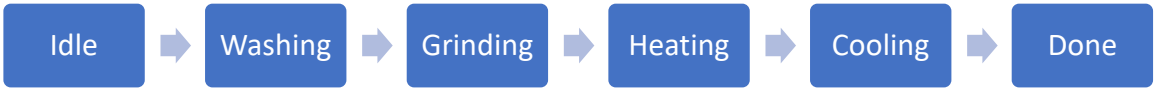


Figure 3. The recycling process for plastic material

2.1.3. Metal-waste recycling process

There are many types of metal with different physical and chemical characteristics, for example, melting point. The simple step to recycling metal waste is to convert it into the metal bar, for example, by following Figure 4. The metal should be first washed or clean to improve the value of their material by segregating clean metal from the dirty material. After that, the metal will be pressed or shredded into small metals, so it will have a large surface-to-volume ratio to promote the melting process. Then, through the heating process, the metal will be melted and then solidified through the cooling process. In this stage, scrap metals are formed into specific shapes such as bars that can be easily used as raw materials to produce various metal products [6].

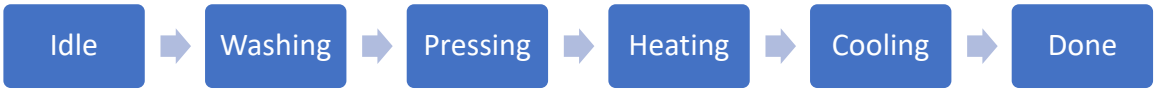


Figure 4. The simplified recycling process for metal

2.1.4. Glass-waste recycling process

Just like metal, glass-waste consist of a different type, size, and color of the glass. The recycling process starts with washing the glass to clean it from contaminants or impurities like dirt and food residue. The glass is then crushed into the coarse particle, which will be then heated to burn sugars and bacteria and loosen label glue. After that, the glass cullet will be cooled down [7]. The simple recycling process can be seen in Figure 5.

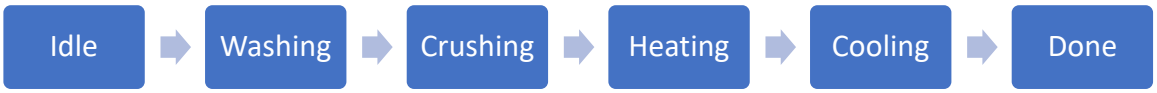


Figure 5. The recycling process for glass

2.2. Design and Functionality

As mentioned in chapter 2.1, in this project, an FPGA for the recycling management process is designed, implemented, and tested with VHDL. The concept is that the recycling process is managed internally by a particular manufacturer or company. The company is expected to have four waste categories: bio, plastic, metal, and glass. It will not produce any waste other than these categories. The whole scenario of the recycling process is described below:

1. The waste is pre-sorted before its being collected. Like in Germany, different rubbish bins with different colors are available around the company, which each symbolized a different waste category.
2. After the bin is full, they will be transported to the recycling processing station based on the type of waste.
3. The waste will go through the complete recycling process defined in 2.1 through the conveyor belt. The sequential recycling process is being designed with the state machine. The state will move from idle to the next state only if the combination of four pushbuttons explained in chapter 4.2.2 is pushed. For each waste category, the waste will be assumed to have the same process time regardless of the characteristic of the waste. For example, in the metal recycling process shown in Figure 4, the time to melt or heat the object will be assumed to be the same, 5 seconds, regardless if it is silver, gold, or aluminum. The predefined time is stated in Table 1 for each state and each waste category.
    - The number of the inputted waste object will be counted by using a counter and recorded in memory. For example, the counter will count if an object on the conveyor triggers the sensor. For biowaste, since it is uncountable, the weight will be considered the trigger to count up the counter. Thus, it can be every 100 grams or even 1 kg, and a number will be raised on the counter.
    - After going through the recycling process, the final product or reusable raw material will be counted using a counter and recorded in memory. How to measure the number of the final product may be different based on each

material type. For example, the metal bar, if it is big enough, a sensor can be used on the conveyor to count up as the trigger for the counter. Alternatively, if it is too small to be counted, which also can be uncountable, a scaler can be used to indicate the counter to count the number up. It can be every 100 grams or even 1 kg.

- The actual number of inputted waste objects and the final product will be displayed on a set of six 7-segment displays. Since there are only a limited number of 7-segment displays, the display can be switched to different types of waste material using the combination of 2 switches explained in 4.2.1. The first two 7-segment displays will show two characters of the waste category's abbreviation, which is explained in chapter 4.4.2—followed by the next two 7-segment displays show two digits integer of the number of actual inputted waste objects. Finally, the last two 7-segment displays show two digits integers of the final product's number based on the selected waste category.
- A switch will be used as well to reset the flipflop, register, and counter.

4. After the process is done, the state machine will go back to an idle state.

Table 1. Time required for each process state

Process state	Time required (in second)			
	Bio	Plastic	Metal	Glass
Washing		3	3	3
Grinding	3	5		
Pressing			5	
Crushing				3
Heating		3	5	5
Cooling		1	5	3
Mixing	3			
Drying	5			

3. Project Requirement

Software:

- IDE: Intel Quartus Prime Lite Edition (v18.1)
- Simulation: ModelSim

For Hardware:

- 5CSEMA5F31C6
- Terasic DE1-SoC Development kit
- FPGA: Altera/Intel Cyclone V SE

4. System Design and Methodology

4.1. Block Diagram

The block diagram of this project is shown in Appendix 1. It starts with the combination of pushbuttons (keys) as the input of selection logic to start the state machine of the selected waste category and the LED. Next, the number of collected products and produced products are saved in the memory of the selected waste category through the flipflop. Finally, these numbers are read from memory and displayed to the six 7-segment display. Switching the six 7-segment displays to show the number of collected/inputted- and produced data based on the selected waste category can be done through MUX 4x1 with two switches as the input: SW[1] and SW[2].

4.2. Input Module

4.2.1. Switches

There are ten switches available on the board. However, only three are used in this project, and the functionality is explained in table 2.

Table 2. The combination and functionality of the switch

Switch	Functionality
SW[0]	Reset the counter, flipflop, and register
SW[2] SW[1]	To switch the six 7-segment displays to show the number of collected/inputted- and produced data based on the selected waste category. [0,0]: Plastic [0,1]: Metal

	[1,0]: Glass [1,1]: Bio

4.2.2. Pushbuttons

The board has four pushbuttons connected to FPGA, and it is named KEY[0], KEY[1], KEY[2], and KEY[3]. The pushbutton generates a low logic level or high logic level when it is pressed or not, respectively [8]. In the project, the combination of this four pushbutton will start the recycling process, as stated in table 3.

Table 3. The combination and functionality of pushbutton

Key[3]	Key[2]	Key[1]	Key[0]	Functionality
1	1	1	0	To start plastic recycling process
1	1	0	1	To start metal recycling process
1	0	1	1	To start glass recycling process
0	1	1	1	To start bio recycling process

4.3. Memory Module

Four memories are used to store the details of each waste category. Each of them is 32 x 8 bits and has the functionality, as shown in table 4.

Table 4. The memory location and its functionality

Memory location	Functionality
0	Store the number of collected items
1	Store the number of produced items

4.3.1. Entities

Six entities are used, of which 5 of them are sub-level entities, and one is a top-level entity.

4.3.1.1. Sub-level entity

- The counter is implemented as the generic entity to produce 1 second, 3 seconds, and 5 seconds.
- Flipflop with asynchronous reset and use D-type flipflop.
- Register with asynchronous reset.
- 7-Segment-display
- Memory: 2 inputs to read/write with two outputs: output A and output B.

4.3.1.2. Top-level entity

It consists of 7 processes in the top-level entity:

- Selection of input material, if it is plastic, bio, metal, or glass
- Display the number of collected- and produced data based on the selected waste category on 7-segment displays
- Process plastic according to the predefined state
- Process glass according to the predefined state
- Process metal according to the predefined state
- Process bio according to the predefined state
- Proceed to next state

4.4. Processing and Output Module

4.4.1. The red LEDs

There are ten user-controllable red LEDs connected to the FPGA. Its associated pin to a high logic level or low level turns the LED on or off, respectively [8]. The functionality is described in table 5.

Table 5. The functionality of LEDR used in this project.

LEDR index	Functionality (the LEDR is on if)
0	If there is at least one process is on
1	The reset signal is active
2	The plastic recycling process is running
3	The metal recycling process is running
4	The glass recycling process is running
5	The biowaste recycling process is running

4.4.2. The 7-segment displays

The board has six 7-segment displays {HEX0, …, HEX5} and it is active low. Figure 6 shows the connection of seven segments to pins on Cyclone V SoC FPGA.

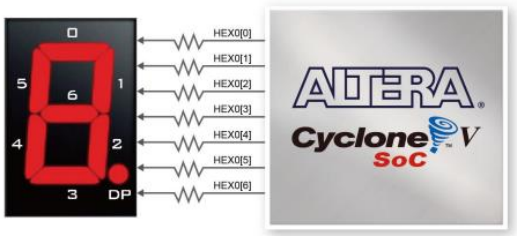


Figure 6. The connection of seven segments to pins on Cyclone V SoC FPGA [8]

The seven segments can be coded to show particular characters as described in table 6. As seen in Appendix 2, the input hex as four bits will be used as the parameter to choose the displayed character for seven segments, shown in table 6.

Table 6. The truth table for customed 7-segment decoder

Input Hex	Segment 0	Segment 1	Segment 2	Segment 3	Segment 4	Segment 5	Segment 6	Displayed character
0	0	0	0	0	0	0	1	0
1	1	0	0	1	1	1	1	1
2	0	0	1	0	0	1	0	2
3	0	0	0	0	1	1	0	3
4	1	0	0	1	1	0	0	4
5	0	1	0	0	1	0	0	5
6	0	1	0	0	0	0	0	6
7	0	0	0	1	1	1	1	7
8	0	0	0	0	0	0	0	8
9	0	0	0	0	1	0	0	9
A	1	1	1	0	0	0	1	L
B	1	1	0	0	0	0	0	B
C	0	0	1	1	0	0	0	P
D	0	1	0	0	0	0	1	G
E	0	1	1	0	0	0	0	E
F	0	1	0	0	1	0	1	S

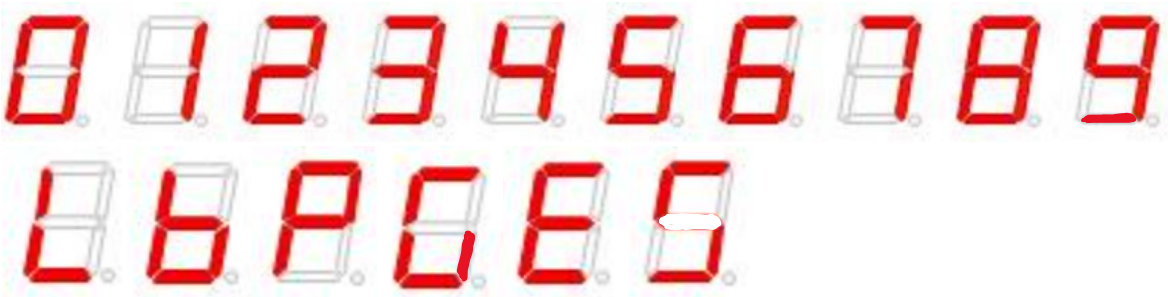


Figure 7. The 7-segment characters [9]

Characters like ‘P’, ‘L’, ‘B’, ‘E’, ‘G’, ‘S’, and ‘O’ are needed to be used as the abbreviation of the waste category according to the combination of two switches: SW[1] and SW[2], shown in table 7. A small modification was made to differentiate ‘S’ with ‘5’ and ‘G’ with ‘6’, which can be seen in Figure 7.

Table 7. The combination of switches to show a specific waste category on the 7-segment display

Switch		Material	7-segment display	
SW[2]	SW[1]		HEX 5	HEX 4
0	0	Plastic	P	L
0	1	Metal	E	L
1	0	Glass	G	S
1	1	Bio	B	O

Table 8. The functionality of six 7-segment display

7-segment displays		Functionality
HEX5	HEX4	Show the abbreviation of the selected waste category on 7-segment-displays described in Table 7.
HEX3	HEX2	The number of collected/inputted data in integer
HEX1	HEX0	The number of produced data in integer

4.5. System Clock

The circuit uses the CLOCK_50, which means the 50 MHz clock provided on the DE1-SoC board.

4.6. Timing constraints

The frequency was set to 200 MHz and the maximum possible frequency achieved is 213.4 MHz for the 85^{o} Celsius model and 214.5 MHz for the 0^{o} Celsius model.

- Maximum negative slack : -3.857 ns

- Worst case propagation delay : 4.857 ns
- Max Frequency : 205.89 MHz
- Clock-skew , t_skew : 4.571ns - 3.894ns = 0.677 ns
- Final clock-skew and clock pessimism : 0.677 ns + 0.346 ns = 1.023 ns
- t_slack = 1 ns - 4.216 ns + 1. 023 ns = -2.193 ns
- Clock uncertainity => t_slack = -2.193 ns - 0.31 ns = - 2.503 ns
- uTsu = - 3.857 ns - (2.503 ns) = - 1.354 ns

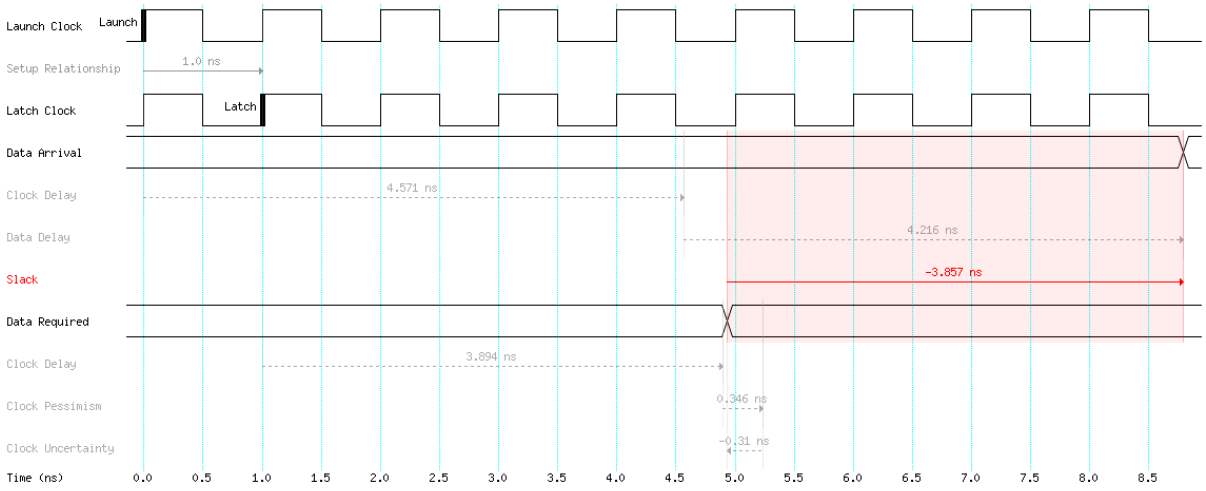


Figure 8. TimeQuest analysis before setting time constraint

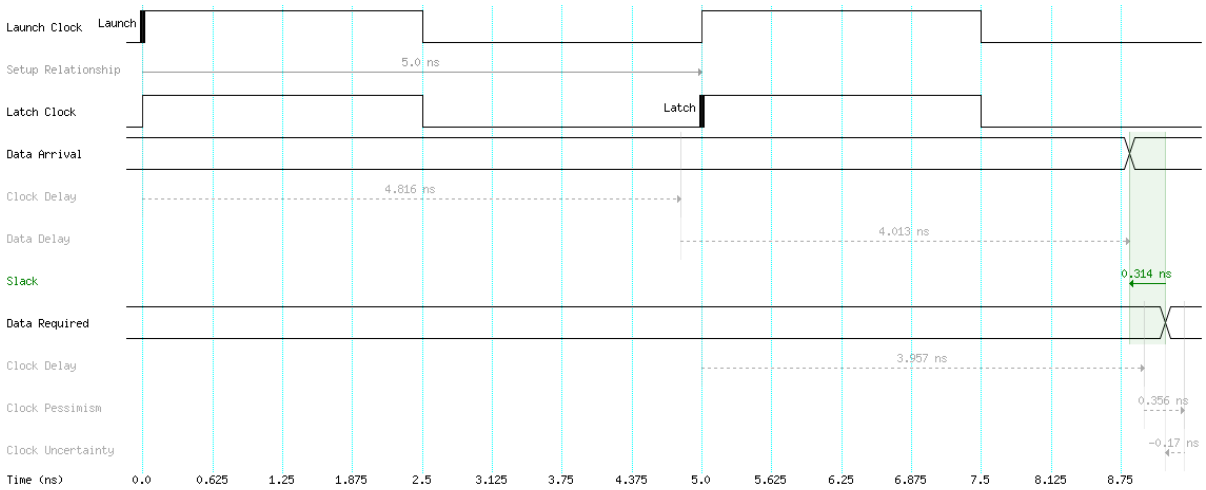


Figure 9. TimeQuest analysis after setting time constraint

4.7. State machine

By implementing the timing in table 1 in the recycling process in chapter 2.1, the state machine for each waste category can be generated and implemented in the VHDL, as shown in the four figures below.

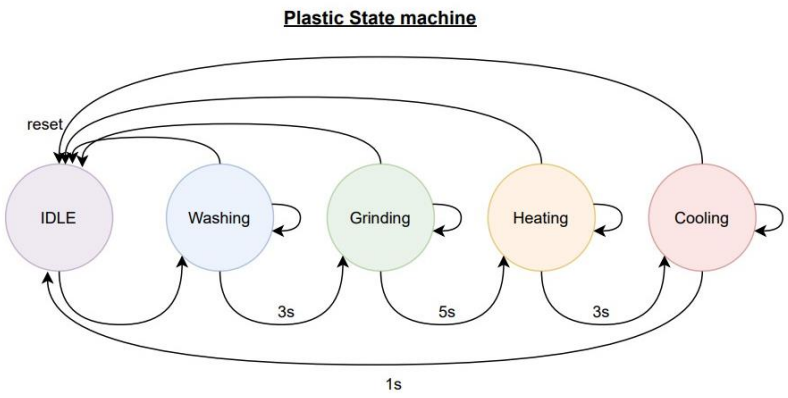


Figure 10. The plastic waste state machine

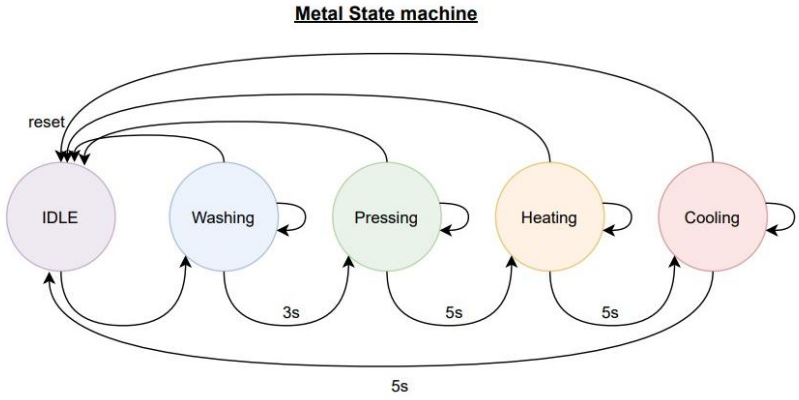


Figure 11. The metal waste state machine

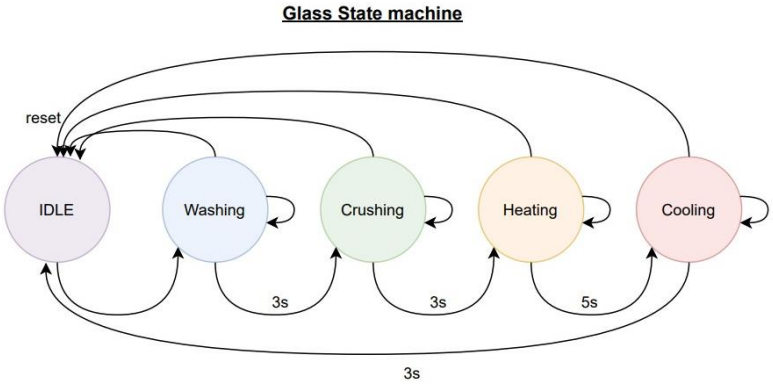


Figure 12. The glass waste state machine

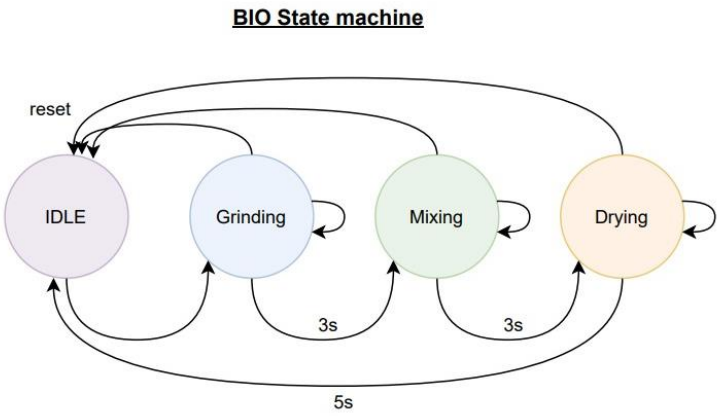


Figure 13. The biowaste state machine

4.7.1. Algorithmic State Machine (ASM) chart

The algorithmic state machine (ASM) chart can be seen in Appendix 3.

5. Result and Analysis

The result of the project includes the correctness of the overall design by inspecting the timing diagram. In addition, simulation test benches enforce input values and check the integrity of the design. The timing diagram of modules when different recycling processes were started through the key combination is illustrated in Figure 14, Figure 15, and Figure 16. There are four different processes: plastic, biowaste, metal, and glass. Each represents a finite state machine responsible for carrying on processing according to predefined steps and timing. In figure 14 can be seen the incremented value of collected and produced data, which will be saved in the memory. Figure 15 shows the triggers if the data is fetched/read from corresponding memory, incremented, and written back to the exact memory location. Lastly, it shows the signals when the process of each waste recycling process is done.

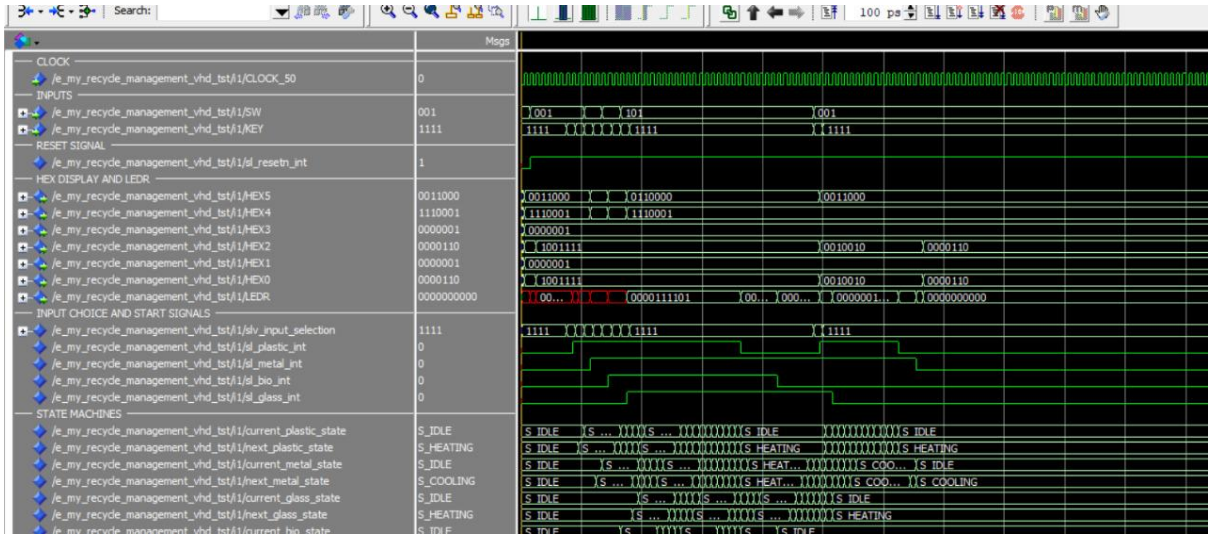


Figure 14. Timing diagram from clock until state machines

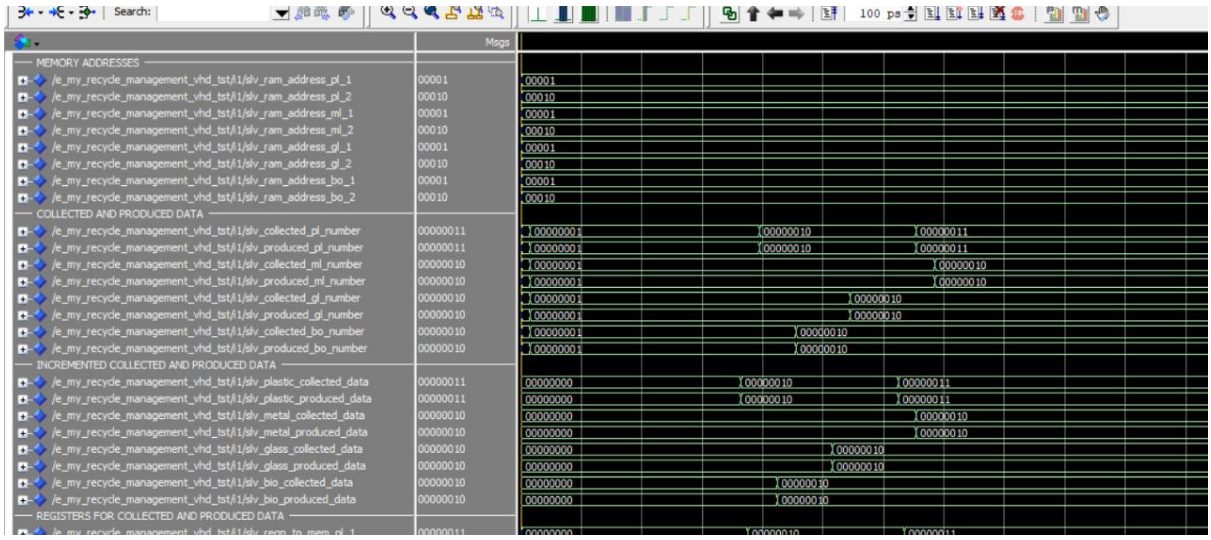


Figure 15. Timing diagram from memory addresses until incremented collected and produced data

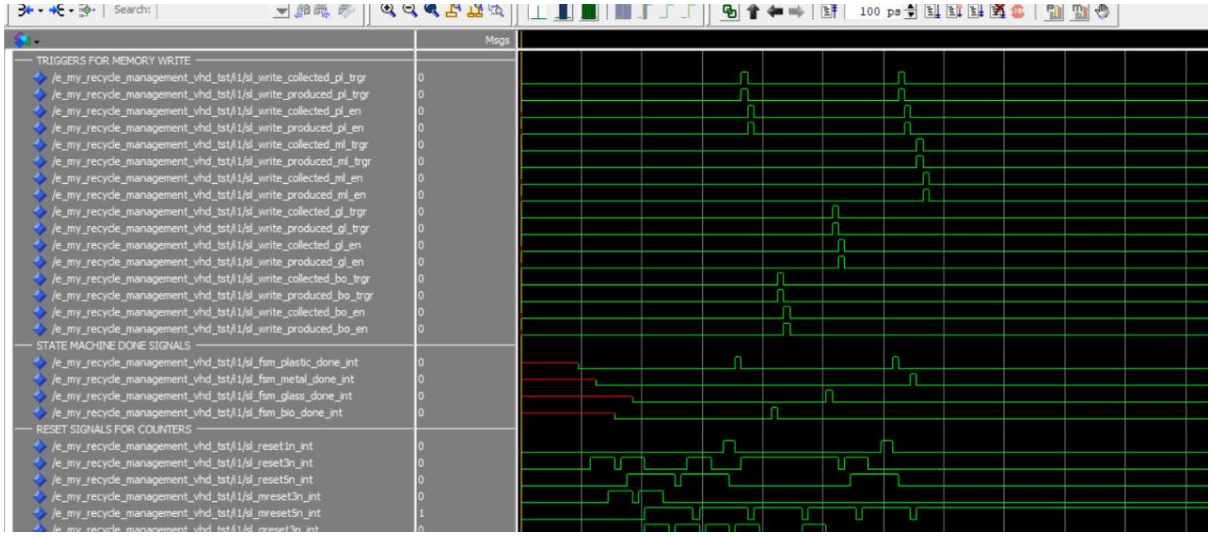


Figure 16. The timing diagram for triggers for writing in the memory until reset for counters

The analysis of the output using the test bench shows the correct output from submodules to the top module, shown in Figure 16. However, if we wanted to test for a real-life application, testing in actual board or hardware is necessary too, because it may show a different result. One of the reasons may seem to be a hardware bug, which resulted in unpredictable discrepancies.

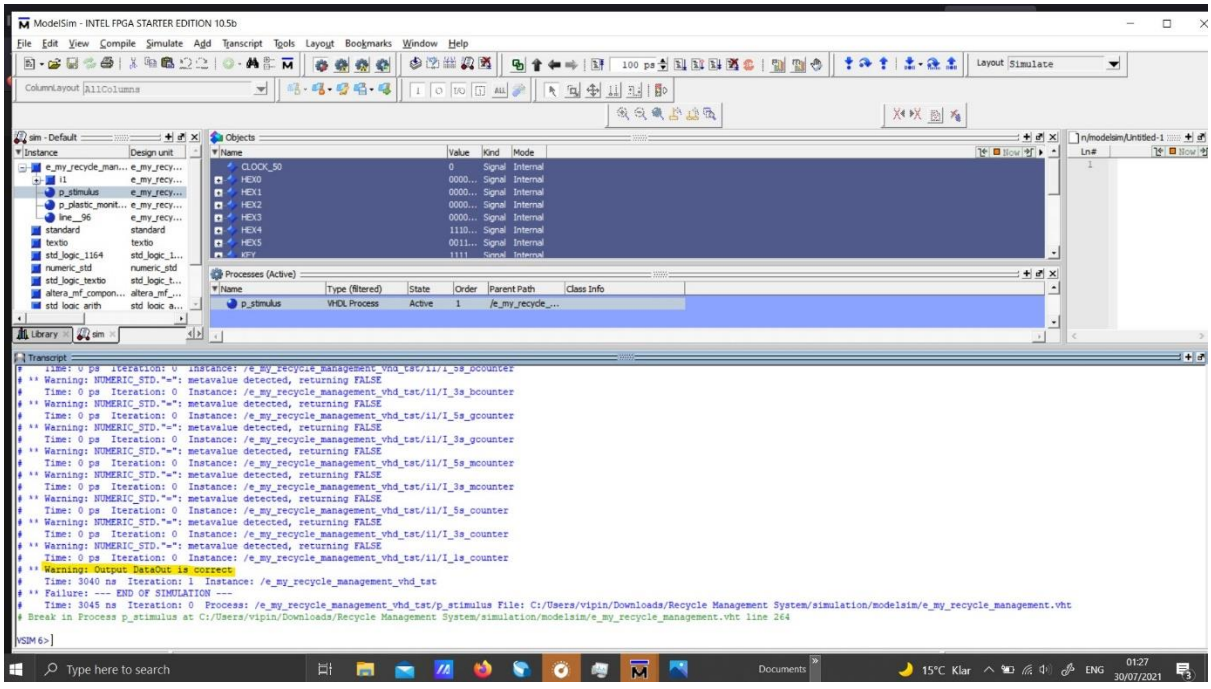


Figure 17. The result after running test bench on the transcript

6. Conclusion and Future Work

High performance and low power consuming computing devices like FPGA have been popular commercially, as well in research and educational fields. Moreover, its flexibility and configurable design with VHDL through IDE: Intel Quartus Prime Lite Edition and ModelSim allow us to get familiarized and get the chance to design and implement concepts, methods, and scenarios for developing advanced-level real-life projects in upcoming years, like recycling management process.

Future improvements for this project can involve the following changes.

- Test in real hardware

  Since the lecture was done during the Corona pandemic period, we did not have a chance to test the design on the actual board. Self-checking testbench has helped us to simulate and verify our FPGA designs. However, it will also be necessary to test it on the actual board to play around with the input module attached to it and discover unforeseen problems, such as hardware bugs.
- Increase the complexity of the recycling process and state machine

  Added more processing states for each waste category based on how it should be done in reality. Make the timing for each state customizable with, for example, pushbuttons or switches.
- Connect additional 7-segment display or maximize the usage of available 7-segment display

  There is a limited number of 7-segment displays available on the board. Based on the way we programmed, the number of collected objects and produced objects can only be displayed with a maximum of two digits long. By connecting the 7-segment display to the board, it will allow us to display a longer number. Alternatively, with a limited 7-segment display on the board, we can apply scrolling or moving text across the display. Therefore, we can display longer text or numbers.
- Save other data in memory

  In reality, more data are needed and valuable for recycling management systems, like the timestamp of when the object is detected on the very first sensor or last sensor, the current temperature of the heating machine and cooling machine, and much more that should be stored in the memory.

7. Bibliography

[1] S. Pandit and P. Sikka, “Design and Implementation of Power Optimized Dual Core and Single Core DLX Processor on FPGA,” IEEE, Bengaluru, India, 2018.

[2] “The waste problem,” Pearl Global, [Online]. Available: https://pearlglobal.com.au/the-waste-problem/. [Accessed 29 July 2021].

[3] A. Bahraini, “Waste4Change Supports 3R (Reduce-Reuse-Recycle) Green Concept!,” Waste4change, 6 May 2019. [Online]. Available: https://waste4change.com/blog/waste4change-supports-3r-reduce-reuse-recycle-green-concept/. [Accessed 29 July 2021].

[4] “Biodegradable Waste,” 2019. [Online]. Available: https://www.sciencedirect.com/topics/engineering/biodegradable-waste. [Accessed 29 July 2021].

[5] “The Complete Plastics Recycling Process,” rts, 12 October 2020. [Online]. Available: https://www.rts.com/blog/the-complete-plastics-recycling-process-rts/. [Accessed 29 July 2021].

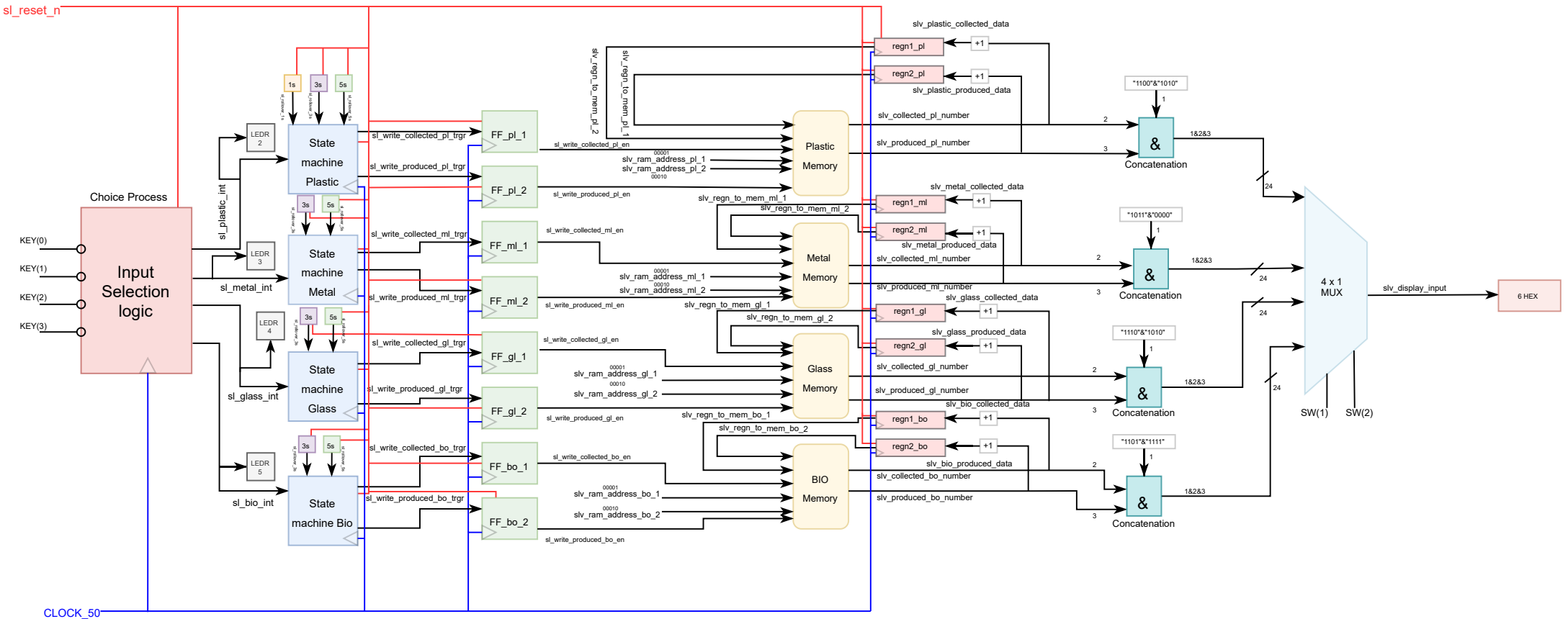
[6] R. Leblanc, “An Introduction to Metal Recycling,” 05 March 2021. [Online]. Available: https://www.thebalancesmb.com/an-introduction-to-metal-recycling-4057469. [Accessed 29 July 2021].

[7] “Glass Reprocessing,” gaskells-waste matters, [Online]. Available: https://gaskellswaste.co.uk/recycling/glass-reprocessor/. [Accessed 29 July 2021].

[8] “DE1-SoC User Manual,” Terasic Technologies, 2014.

[9] Kkaje, “Seven Segment Display 1,” 26 February 2019. [Online]. Available: https://resilinets.org/file_sevensegcharacters.html. [Accessed 29 July 2021].

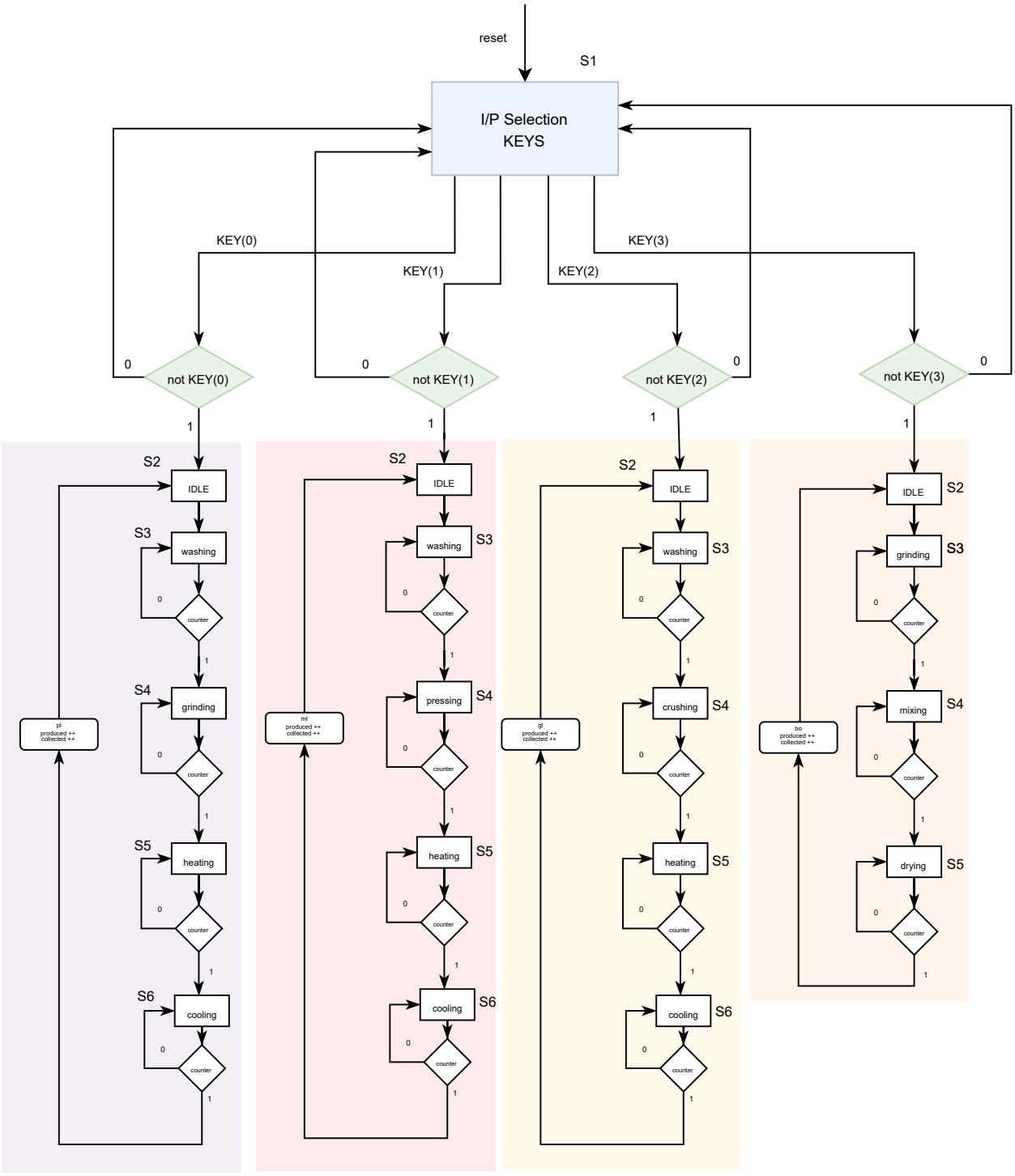
Appendix 1


sl_reset_n
Choice Process
Input Selection logic
KEY(0)
KEY(1)
KEY(2)
KEY(3)
sl_plastic_int
sl_metal_int
sl_glass_int
sl_bio_int
LEDR 2
LEDR 3
LEDR 4
LEDR 5
1s
3s
5s
State machine Plastic
State machine Metal
State machine Glass
State machine Bio
sl_write_collected_pl_trgr
sl_write_produced_pl_trgr
sl_write_collected_ml_trgr
sl_write_produced_ml_trgr
sl_write_collected_gl_trgr
sl_write_produced_gl_trgr
sl_write_collected_bo_trgr
sl_write_produced_bo_trgr
FF_pl_1
FF_pl_2
FF_ml_1
FF_ml_2
FF_gl_1
FF_gl_2
FF_bo_1
FF_bo_2
sl_write_collected_pl_en
sl_write_produced_pl_en
sl_write_collected_ml_en
sl_write_produced_ml_en
sl_write_collected_gl_en
sl_write_produced_gl_en
sl_write_collected_bo_en
sl_write_produced_bo_en
slv_regn_to_mem_pl_1
slv_regn_to_mem_pl_2
slv_ram_address_pl_1
slv_ram_address_pl_2
Plastic Memory
slv_regn_to_mem_ml_1
slv_regn_to_mem_ml_2
slv_ram_address_ml_1
slv_ram_address_ml_2
Metal Memory
slv_regn_to_mem_gl_1
slv_regn_to_mem_gl_2
slv_ram_address_gl_1
slv_ram_address_gl_2
Glass Memory
slv_regn_to_mem_bo_1
slv_regn_to_mem_bo_2
slv_ram_address_bo_1
slv_ram_address_bo_2
BIO Memory
slv_plastic_collected_data
regn1_pl
regn2_pl
slv_plastic_produced_data
slv_collected_pl_number
slv_produced_pl_number
slv_metal_collected_data
regn1_ml
regn2_ml
slv_metal_produced_data
slv_collected_ml_number
slv_produced_ml_number
slv_glass_collected_data
regn1_gl
regn2_gl
slv_glass_produced_data
slv_collected_gl_number
slv_produced_gl_number
slv_bio_collected_data
regn1_bo
regn2_bo
slv_bio_produced_data
slv_collected_bo_number
slv_produced_bo_number
+1
"1100"&"1010"
"1011"&"0000"
"1110"&"1010"
"1101"&"1111"
Concatenation
1&2&3
4 x 1 MUX
SW(1)
SW(2)
slv_display_input
6 HEX
CLOCK_50

Appendix 2

```
library ieee;
use ieee.std_logic_1164.all;

entity e_hex7seg is
   port ( slv_hex     : in  std_logic_vector(3 downto 0);
          slv_display : out std_logic_vector(6 downto 0));
end entity e_hex7seg;

architecture a_hex7seg of e_hex7seg is

---- Declaration Part ------------------------------------------------

begin

---- Assignment Part -------------------------------------------------

   --
   p_Decode: process (slv_hex)
   begin
      case slv_hex is
         when "0000" => slv_display <= "0000001";--0
         when "0001" => slv_display <= "1001111";--1
         when "0010" => slv_display <= "0010010";--2
         when "0011" => slv_display <= "0000110";--3
         when "0100" => slv_display <= "1001100";--4
         when "0101" => slv_display <= "0100100";--5
         when "0110" => slv_display <= "0100000";--6
         when "0111" => slv_display <= "0001111";--7
         when "1000" => slv_display <= "0000000";--8
         when "1001" => slv_display <= "0000100";--9
         when "1010" => slv_display <= "1110001";--A -> l -- done
         when "1011" => slv_display <= "1100000";--B -> b
         when "1100" => slv_display <= "0011000";--C -> p -- done
         when "1101" => slv_display <= "0100001";--D -> g -- done
         when "1110" => slv_display <= "0110000";--E
         when others => slv_display <= "0100101";--F -> S -- done
      end case;
   end process p_Decode;

end architecture a_hex7seg;
```

Appendix 3


reset
S1
I/P Selection
KEYS
KEY(0)
KEY(1)
KEY(2)
KEY(3)
not KEY(0)
not KEY(1)
not KEY(2)
not KEY(3)
0
1
S2
IDLE
S3
washing
counter
S4
grinding
S5
heating
S6
cooling
pi
produced ++
collected ++
mi
produced ++
collected ++
gi
produced ++
collected ++
bio
produced ++
collected ++
S2
IDLE
S3
washing
counter
S4
pressing
S5
heating
S6
cooling
S2
IDLE
S3
washing
S4
crushing
S5
heating
S6
cooling
S2
IDLE
S3
grinding
S4
mixing
S5
drying