VHDL SPECIFICATION

Circuit Design (WS2021/22)
Prof. Dr. -Ing Andreas Siggelkow

Engli Shehu & Amir Reza Edalati

Hochschule Ravensburg-Weingarten
Germany
09.11.2021

Contents

Chapter 1

History - Change Log

Version	Date	Description
V 0.1	26.10.2021	Requirements Table
V 0.2	09.11.2021	Spefication added
V 0.3	08.12.2021	Timer and counter entities
V 0.4	13.12.2021	Full top level and running simulation without uart
V 0.5	18.12.2021	Complete uart and synthesis
V 0.6	06.01.2022	Errors are fixed
V 0.7	15.01.2022	Top level changed
V 1.0	17.01.2022	Final specification

Current Version:	V 1.0
Description:	Final Specification

Chapter 2

Description

This is an application which aims to control the brewing time of a teabag. The system uses a single sensor to detect when a Tea Bag is dropped in, so this solution is only feasable on the condition that only one tabag is being brewed at a time. The brewing time is adjustable between 2:00-5:00 min. After the brewing time is done the system will ring an alarm to notify procedure finished and than the time is recorded. After this the system is rearmed again. The Data is saved on a PC to be accessed later.

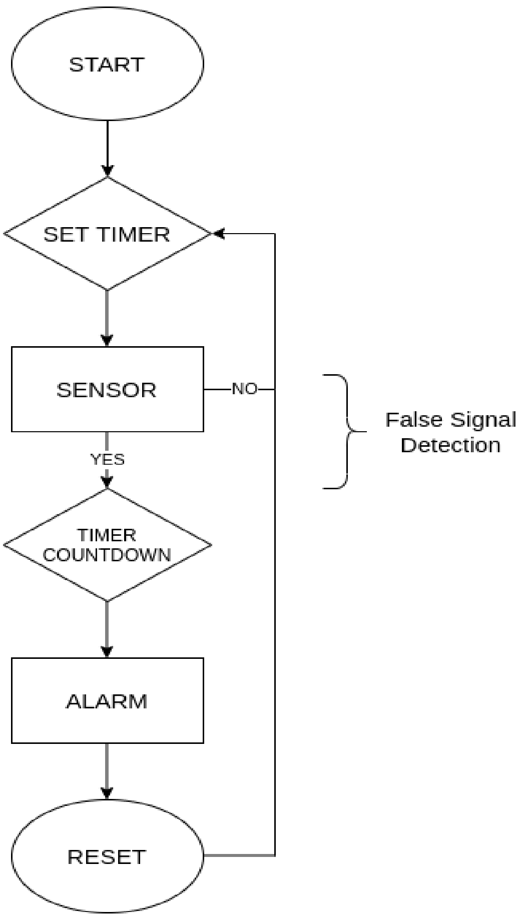

Figure 2.1: Flowchart.

Chapter 3

Requirements Table

ID	Name	Importance	Test	Description
General				
G1,1	Time	High	VHDL Testbench	Time of brewing.
G1,2	Start	High	VHDL Testbench	Remarks when the Tea Bag is droppped in.
G1,3	Stop	High	VHDL Testbench	Notifies when the brewing should stop.
Interface				
I2,1	Sensor	High	VHDL Testbench	Senses when the Tea Bag is dropped in.
I2,2	Alarm	High	VHDL Testbench	Activated when the brewing should Stop.
I2,3	GPS	High	VHDL Testbench	Sends the time via the NMEA Dataset.
I2,4	Local Key	High	VHDL Testbench	Sends the time localy.
I2,5	ASCII-receiver-interface	High	VHDL Testbench	Expects a time in Seconds in ASCII format.
UART				
U3,1	9K6	High	VHDL Testbench	The speed of the serial transmission shouuld be set to 9600 baud.
U3,2	8N2	High	VHDL Testbench	The data width of the serial transmission should be set to 8 bit.
U3,3	8N2	High	VHDL Testbench	The serial transmission should not be checked with a parity bit.
U3,4	8N2	High	VHDL Testbench	The serial transmission should have twos stop bits.
U3,5	UART Time	High	VHDL Testbench	The time stamp the event should be delivered to a PC.
U3,6	UART Start	High	VHDL Testbench	The start of the event should be delivered to a PC.
U3,7	UART Stop	High	VHDL Testbench	Ending of the event should be delivered to a PC.
U3,8	Working Data	High	VHDL Testbench	Working Data should be stored in 32-bit registers.
U3,9	UART-RX	High	VHDL Testbench	32-bit Registers should be loadable by UART-RX
U3,10	UART-TX	High	VHDL Testbench	32-bit Registers should be readable by UART-TX
U3,11	Register Accessibility	High	VHDL Testbench	32-bit Registers should be adressable
PC				
P4,1	PC Language	High	PC	The Programming Language is C++

Chapter 4

Top Level View

4.1 Block Diagram

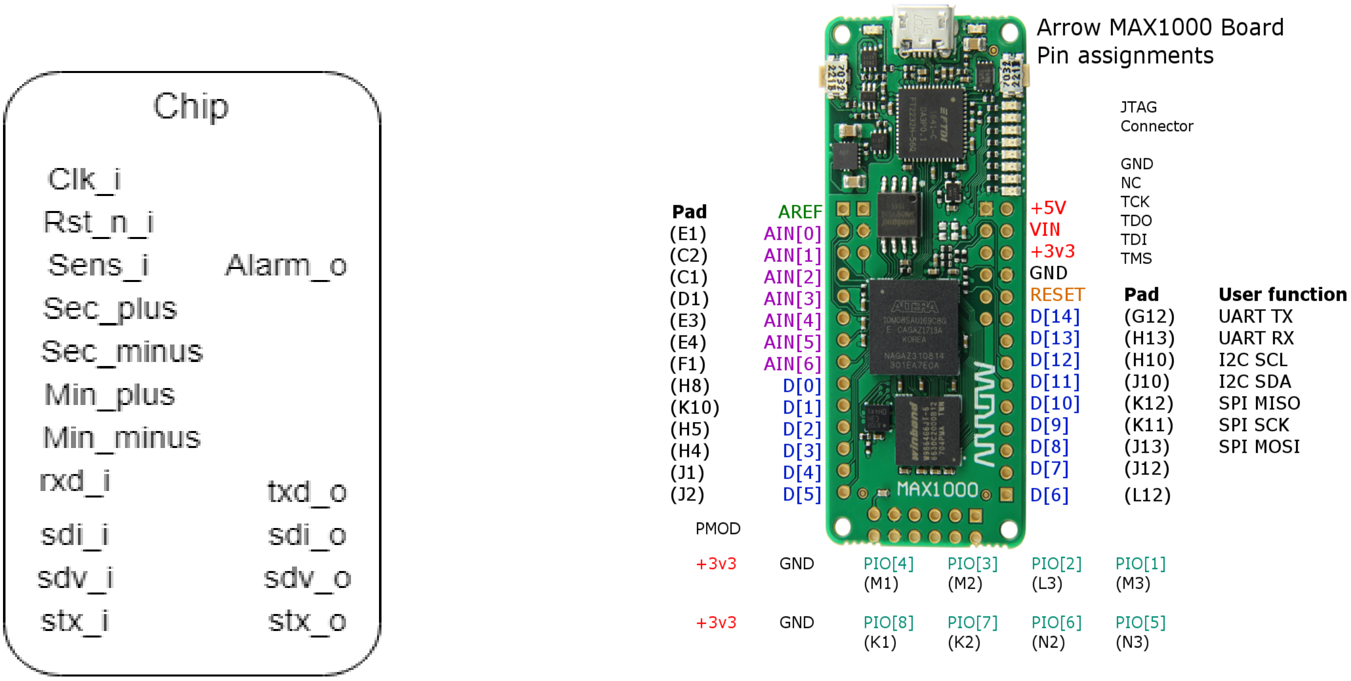

Figure 4.1: Top View.

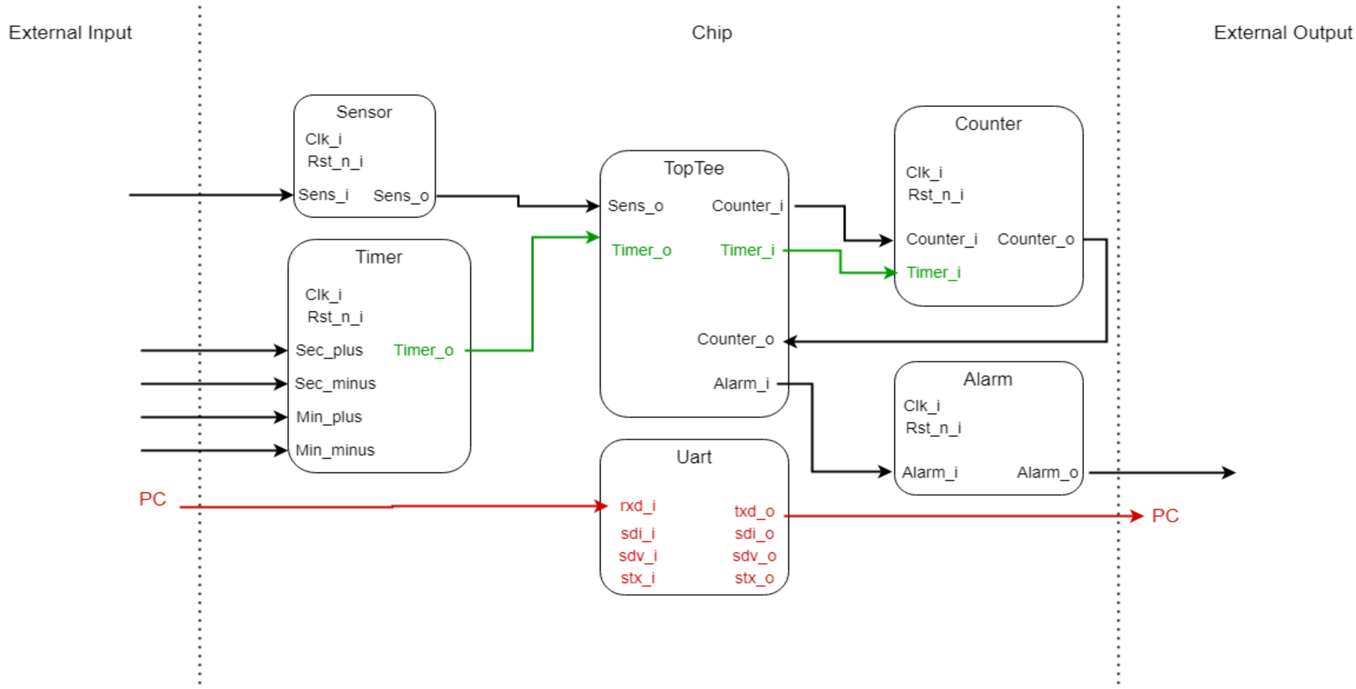

Figure 4.2: Top Level Block Diagram.

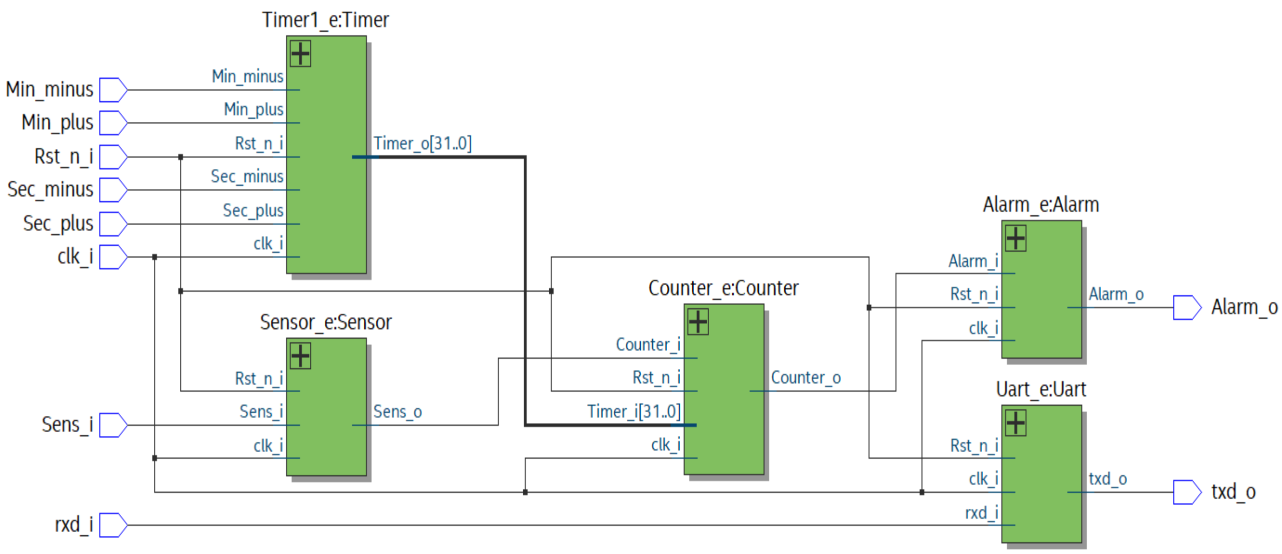

Figure 4.3: Top Level Block Diagram Rtl.

4.2 Description

The Top Level consists of the following entities:
- Timer_e
In the timer entity we take external inputs from the user for which we set the time that the tea has to boil. Then we send the time to the counter.
- Counter_e
In the counter we receive two inputs, one from the timer and the other from sensor. Then we count down the time and at the end we output the signal.
- Sensor_e
In the sensor we receive an external output signal, debounce it and then output it. The sensor is triggered when the tea bag is dropped.
- Alarm_e
In the alarm we output a signal for three seconds.
- Uart_e
In the Uart we receive and transmit data with the local PC.

4.3 Top level signal list

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
Sec_plus	IN	1	Increment one second
Sec_minus	IN	1	Decrement one second
Min_plus	IN	1	Increment sixty seconds
Min_minus	IN	1	Decrement sixty seconds
Sens_i	IN	1	Input from external sensor
Alarm_o	OUT	1	Output to external source
rxd_i	IN	1	Data input from Uart
txd_o	OUT	1	Data output from Uart
Timer_o	Signal	32	Output time from timer
Timer_i	Signal	32	Input time to counter
Counter_i	Signal	1	Start signal for counter
Counter_o	Signal	1	Output signal from counter
Alarm_i	Signal	1	Start signal for alarm

Chapter 5

System Design

5.1 Sensor

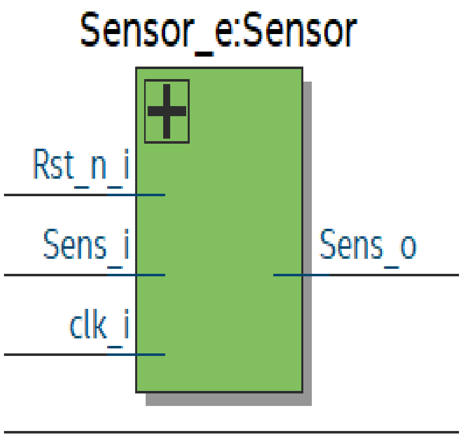

Figure 5.1: Sensor Top Level.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
Sens_i	IN	1	Input from external sensor
Sens_o	OUT	1	Output after debounce
debouncer_1	Signal	1	
debouncer_2	Signal	1	

DESCRITPION

In our system we operate the sensor using a push button. At first the signal is debounced in order to make sure that the system is not triggered by noise. After this the sensor gives an output when the button is released. This is done in order t prevent the sensor from triggering repeatedly.

5.2 Timer

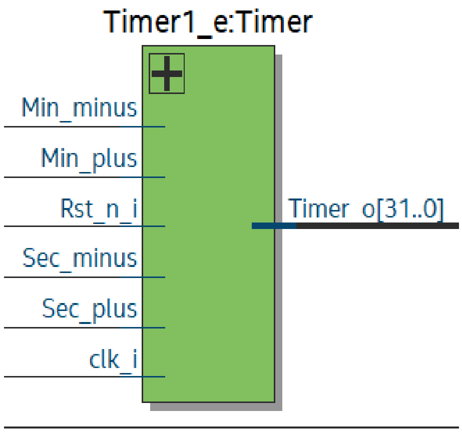

Figure 5.2: Timer Top Level.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
Sec_plus	IN	1	Increment one second
Sec_minus	IN	1	Decrement one second
Min_plus	IN	1	Increment sixty seconds
Min_minus	IN	1	Decrement sixty seconds
Timer_o	OUT	32	Register for sending time
timer	Signal	32	
debouncer_1	Signal	1	
debouncer_2	Signal	1	
timer_state	FSM	5 states	
state	signal	1	

DESCRITPION

In our system we set the time using four input signals that the user drives. Afterwards the time is saved to a register and then outputed inside the system for counting. All the input signals are arranged via a FSM and all of them are debounced. The system is designed so that if a tea is already boiling the timer ca not be changed.

5.3 Counter

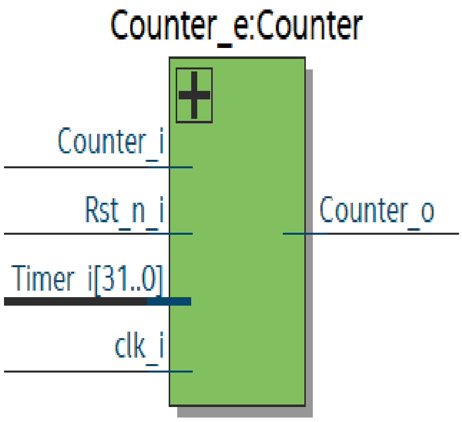

Figure 5.3: Counter Top Level.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
Timer_i	IN	32	Timer register
Counter_i	IN	1	Signal to start counting
Counter_o	OUT	1	Output signal when finished counting
Ticks	Signal	32	
Seconds	Signal	32	
Counter_input	Signal	32	
Counter_state	Signal	1	

DESCRITPION

The counter entity requires two inputs. First input is taken from the debounced sensor and starts the countdown of the time. The second input is the 32 bit time register which should not be zero for the counter to starts its process. In the counter we count every rising edge and save it to a register until ewe have one second. Then we save its to another register and count it until our value matches the timer input. When the values are equal the counter sends the output signal high to the alarm entity.

5.4 Alarm

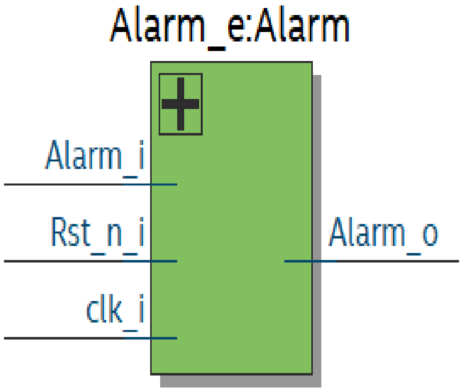

Figure 5.4: Alarm Top Level.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
Timer_i	IN	32	Timer register
Alarm_i	IN	1	Input signal to start the alarm process
Alarm_o	OUT	1	Output signal to alarm module
ticks	Signal	32	
state	Signal	1	

DESCRITPION

The alarm entity receives an input signal to start counting for three seconds. While it is counting, it outputs a high signal which is wired to the external alarm module. The same procedure as in the counter is used to countdown three seconds. When the countdown is finished, the output signal goes back to low and the alarm is in a ready state again.

5.5 Uart

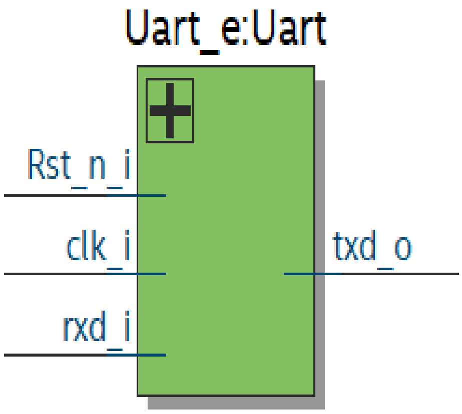

Figure 5.5: Uart Top Level.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
rxd_i	IN	1	received data signal
txd_o	OUT	1	transmitted data signal
state	FSM	10	
idle	signal	1	
state1	FSM	10	
idle1	signal	1	
start	Signal	1	
stop2	Signal	1	
store	Signal	8	

DESCRITPION

The uart is used to received and transmit data between the system and local PC. The PC listens to predefined COM port for data transmitted from the system. The data is transmitted in eleven bit pairs, one start bit, 8 data bits, No parity bit, two stop bits. When the system is triggered data is sent to the PC and the local time is displayed. At the alarm out high, the counted time is sent and displayed to the local PC as well as the local time.

Chapter 6

VHDL Design

6.1 Active low reset

An active low reset is triggered when the reset goes from a high state to a low one. This is implemented in order to make sure that if there is an electrical or wiring malfunction with the reset, the system will constantly be in reset mode and the user can not start the system.

6.2 Signal debouncer

OVERVIEW

Whenever a signal is triggered from high to low or vice versa multiple signals are generated. These undesired signal can also be generated by electrical noise. The effects that these staggering signal can cause are multiple inputs and outputs or undesired triggering of the signals. In order to avoid this we use debouncers. A debouncer reads the signal high or low and then wait for certain amount of time to check for the signal state again. If the signal is in the static state module creates a single clock cycle pulse.

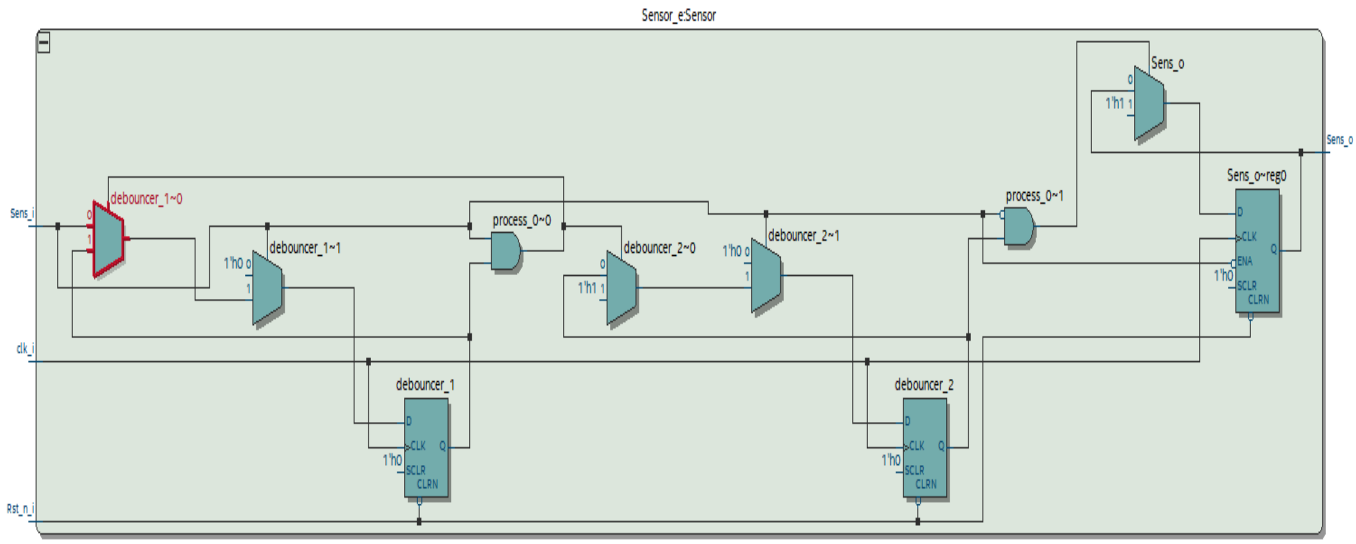

Figure 6.1: Debouncer Logic Diagram.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
udb_i	IN	1	Undebounced signal input
db_o	OUT	1	debounced output signal
state1	Signal	1	
state2	Signal	1	

DESCRIPTION

In our system the signal is debounced twice with each rising edge of the CLK. The first time the signal is debounced to check for noise. If after a new rising edge approximately 160 ns the signal is in static state then the debouncer is set to the signal. Then the debouced signal is kept on high until the input signal is goes low. After this the debounced signal is outputed for one clock cycle. This is done in order to make sure that upon each press of the button, even if the button is pushed for more than 160 ns, the system only gets one output cycle.

6.3 Finite state machine

OVERVIEW

FSM consists of limited states. Transitions between different states are done based on current state and an input. FSM are useful when we need to handle a lot of different cases one at a time.

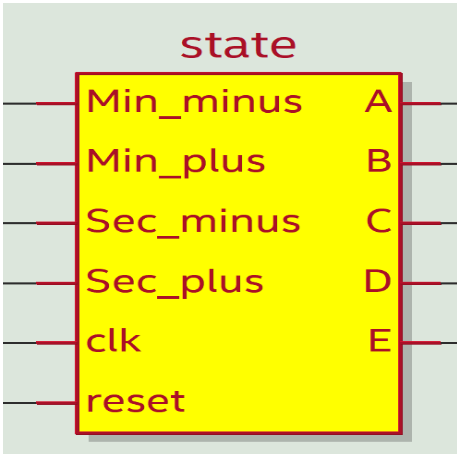

Figure 6.2: FSM Top View.

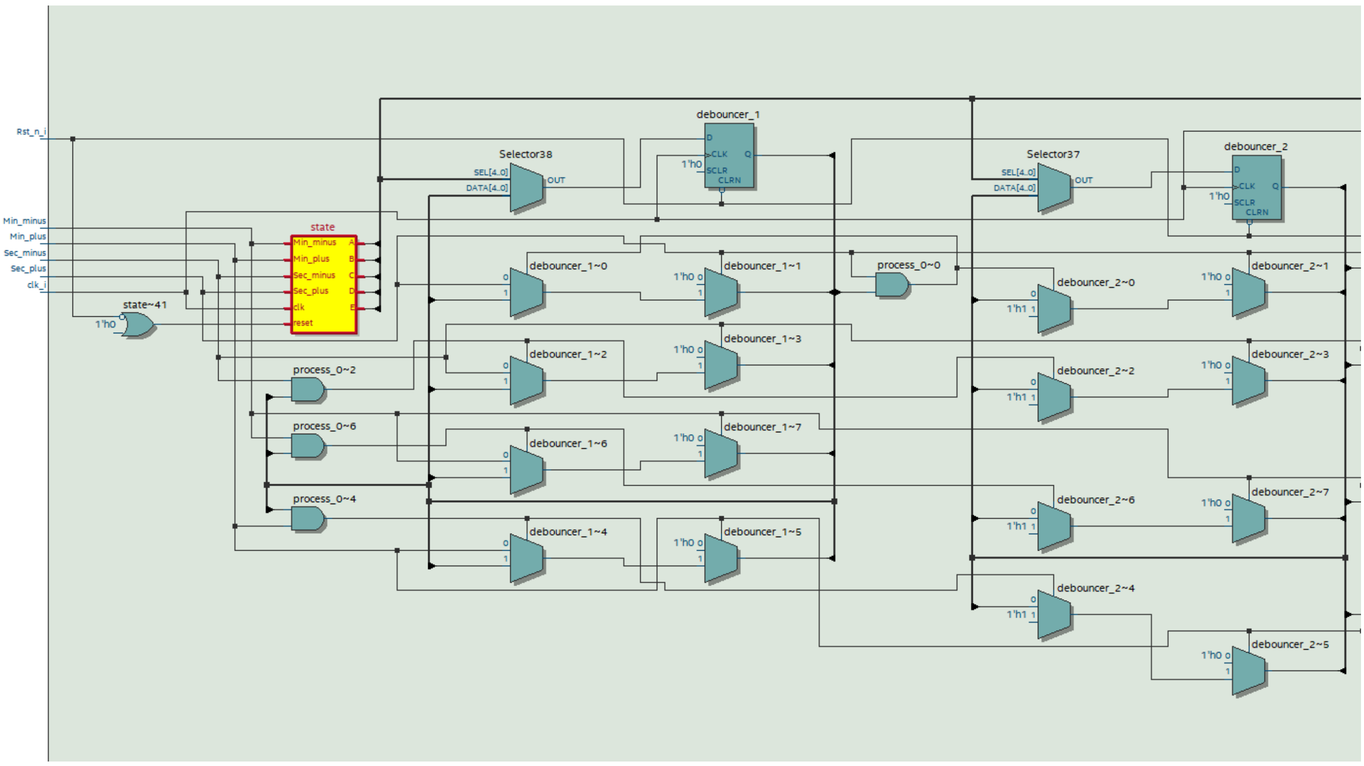

Figure 6.3: FSM Logic Diagram.

Pin	Direction	Width	Description
clk_i	IN	1	12MHz
Rst_n_i	IN	1	Reset, Active Low
state	signal	1	
FSM_name	FSM 5	1	

DESCRIPTION

Our FSM starts with a initial state which awaits an input signal that will determine the next state to go to. This is done every clock cycle. After we go to one of the other states we will start debouncing the input signal and write to our registers. After the signal debounced and we have an output the FSM goes back to initial state. When in the initial state we check the validity of inputs and continue waiting for other signal to go to different state. The advantage of the FSM in this case is that we are able to deal with one input signal and one process at a time.

6.4 Uart communication protocol

Uart transmission always starts with a start bit which goes from a initial high state to low then the 7 bytes are transmitted (in our case 8N2 transmission) from the Least significant bit to the highest significant bit. In the end we have two stop bits which go back to high. Every change of the signal happens at a baudrate of 9600 and is synchronized to happen at the middle of each bit.

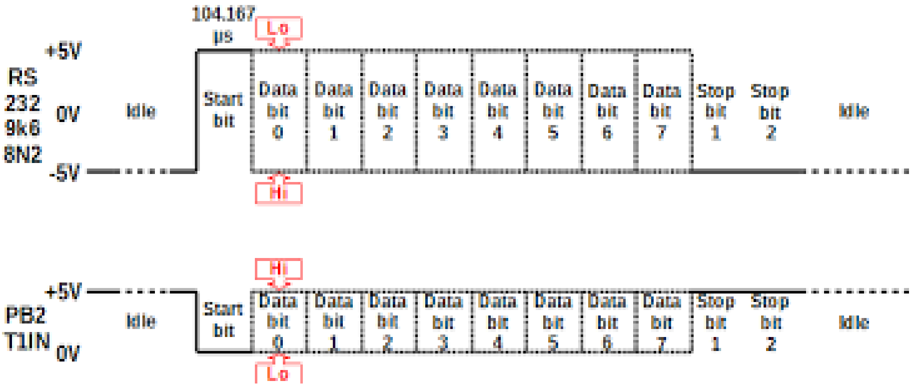

Figure 6.4: Uart 8N2 Serial communication.

Chapter 7

Testbenches

Testbenches are a great tool to test if the complex modules meet all the requirements.

- TopTee_TB
- Alarm_TB
- Counter_TB
- Sensor_TB
- Timer_TB
- UART_TB

Chapter 8

Repository

The Repository contains:

- VHDL Source Code
- VHDL Source Code
- Documentation

The code can be downloaded via this link: https://github.com/hackvoid/Tea_Brewer