HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Specification

Tea Bag Sensor

Study Program - Electrical Engineering and IT
Submitted by - Marko Gjorgjievski (33529)
Partha Pratim Boruah (33526)
Submitted to - Prof. Siggelkow

Contents

List of Figures

List of Tables

1 Requirements

1.1 Requirements

- Speaker : to indicate that the brewing time is completed.
- MAX1000 FPGA Board : to upload the code.
- Quartus Prime Lite : to synthesis the code.
- PC with Monitor : to display the time of arrival and brew completion time.
- PMOD RS-232 Adaptor : To have UART communication.
- Four Buttons : to switch timing modes.
- Four Resistors : to limit current going into the FPGA.
- Wires : to connect the components.
- Breadboard : to connect the components.
- One USB to RS-232 cable : to connect the PMOD Adapter to the PC.
- One USB to USB micro cable : to upload the synthesised with JTAG.

1.2 Table of Requirements

The below Table 1 provies the general requirements.

Requirement	ID	Importance	Verifiable	Description	Remarks
Gen.: Time of Arrival	G01	High	VHDL TB	Counter when tea bag is dipped in boiling water	
Gen.: Min	G02	High	VHDL TB	The brewing time must not be less than minimum (i.e. 2 mins)	
Gen.: Max	G03	High	VHDL TB	The brewing time must not be more than maximum (i.e. 5 mins)	
Gen.: one light sensor	G04	High	VHDL TB	Detection of the tea bag	
Gen.: resolution	G05	Medium	VHDL TB	The resolution of the time must be in seconds	
Sound: dipping	S01	High	VHDL TB	The tea bag is being dipped, play a sound	
Sound: completion	S02	High	VHDL TB	Brewing is completed, play a sound	
Sound: speaker	S03	High	VHDL TB	The signal is passed on to play the sound	If FPGA doesn't have a speaker.
LED: green1	LED01	High	VHDL TB	The system is reset.	
LED: red1	LED02	High	VHDL TB	The heart beat signal i.e. timer is running.	
LED: blue1	LED03	High	VHDL TB	Sets brewing time.	
LED: blue2	LED04	High	VHDL TB	Sets brewing time.	
LED: green2	LED05	High	VHDL TB	Sends a string of ASCII to the PC.	
LED: red2	LED06	High	VHDL TB	The brewing is done.	
UART: 9600 baud	UART01	High	VHDL TB	Serial transmission set to 9600 baud.	
UART: 8 bit	UART02	High	VHDL TB	Data width of transfer bus set to 8 bits	
UART: no parity	UART03	High	VHDL TB	No error checking with parity bit.	
UART: two stop bits	UART04	High	VHDL TB	Two stop bits for the end of serial transmission	
UART: brewing time	UART05	High	VHDL TB	Time stamp for required brewing time.	
UART: start time	UART06	High	VHDL TB	Start stamp provided by GPS System.	
PC: language	PC01	Medium	C Program	PC reads ASCII sent by serial out.	
PC: display	PC02	Medium		Monitor displays data.	

Table 1: Table of Requirements

2 Hardware - MAX1000

2.1 Overview

The MAX1000 is a customizable IoT / Maker Board ready for evaluation, development and/or use in a product. It is built around the Intel MAX10 FPGA, which is the industry's first single chip, nonvolatile programmable logic device (PLDs) to integrate the optimal set of system components. Users can now leverage the power of tremendous re-configurability paired with a highperformance, low-power FPGA system. Providing internally stored dual images with selfconfiguration, comprehensive design protection features, integrated ADCs and hardware to implement the Nios II 32-bit microcontroller IP, MAX10 devices are ideal solution for system management, protocol bridging, communication control planes, industrial, automotive and consumer applications. The MAX1000 is equipped with an Arrow USB Programmer2, SDRAM, flash memory, accelerometer sensor and PMOD/ADRUINO MKR connectors making it a fully featured plug and play solution without any additional costs. [1]

2.2 Layout and Components

Figure 1 shows a top view of the board. It depicts the layout of the board and indicates the location of the various connectors and key components. [1]

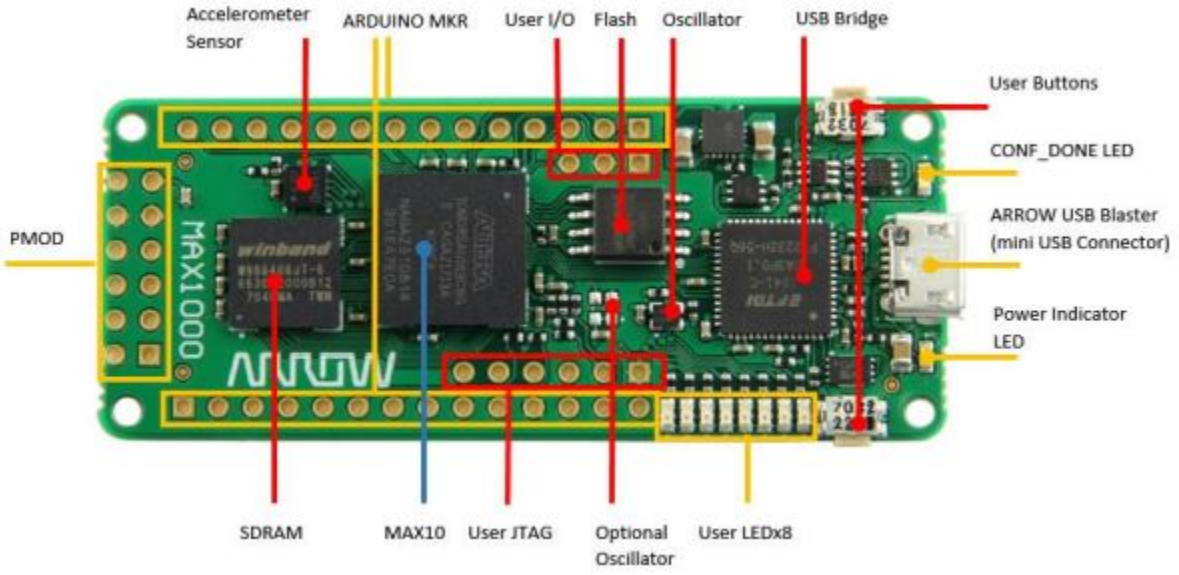

Figure 1: MAX1000 Board (top view) ; Sournce : [1]

The following are available on the MAX1000 board:

- Intel MAX®10 10M08SAU169C8G device
- Arrow USB Programmer2 on-board for programming; JTAG Mode
- 64MBit SDRAM (16-bit data bus)
- 64Mbit Flash Memory
- One 12MHz MEMS Oscillator
- One optional MEMS Oscillator of preferred frequency
- Eight red user LEDs

- Two board indication LEDs
- Two user buttons
- One 3-axis accelerometer
- One 12-pin PMOD header
- One Arduino MKR header
- One User JTAG header
- One User I/O header

2.3 Block Diagram

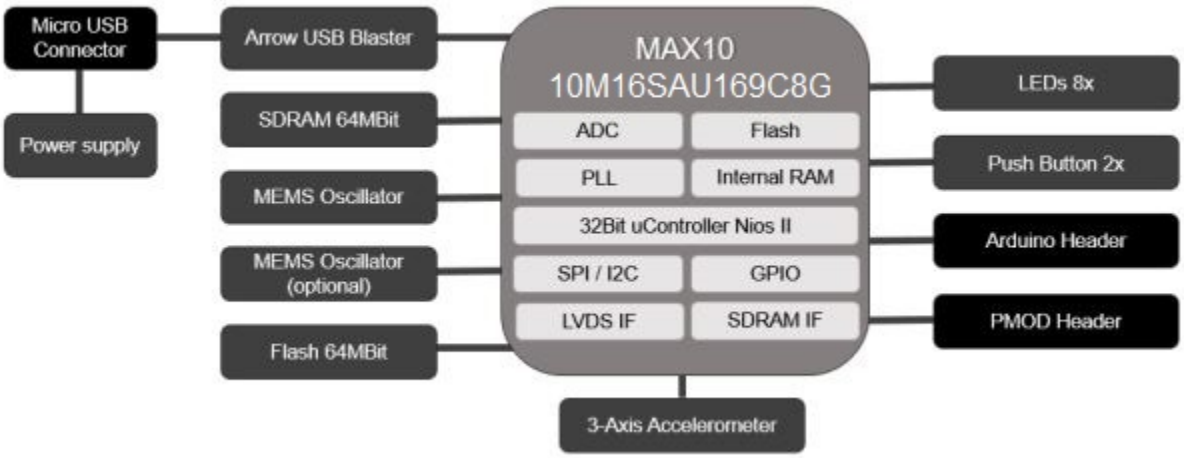

Figure 2: MAX1000 Block Diagram ; Source : [1]

Figure 1 represents the block diagram of the board. All the connections are established through the MAX 10 FPGA device to provide maximum flexibility for users. Users can configure the FPGA to implement any system design. [1]

3 Top Level View

3.1 Overview

A design entity that is the root of a design hierarchy. To compile or simulate a design, you compile or simulate the top-level design entity. A top-level design entity can contain any number of lower-level design entities (or, subdesigns), and is the parent for the lower-level design entities. When you create a project, you specify a top-level design entity for the project. You can also specify a different top-level design entity for each of a project's revisions. [2]

3.2 Block Diagram

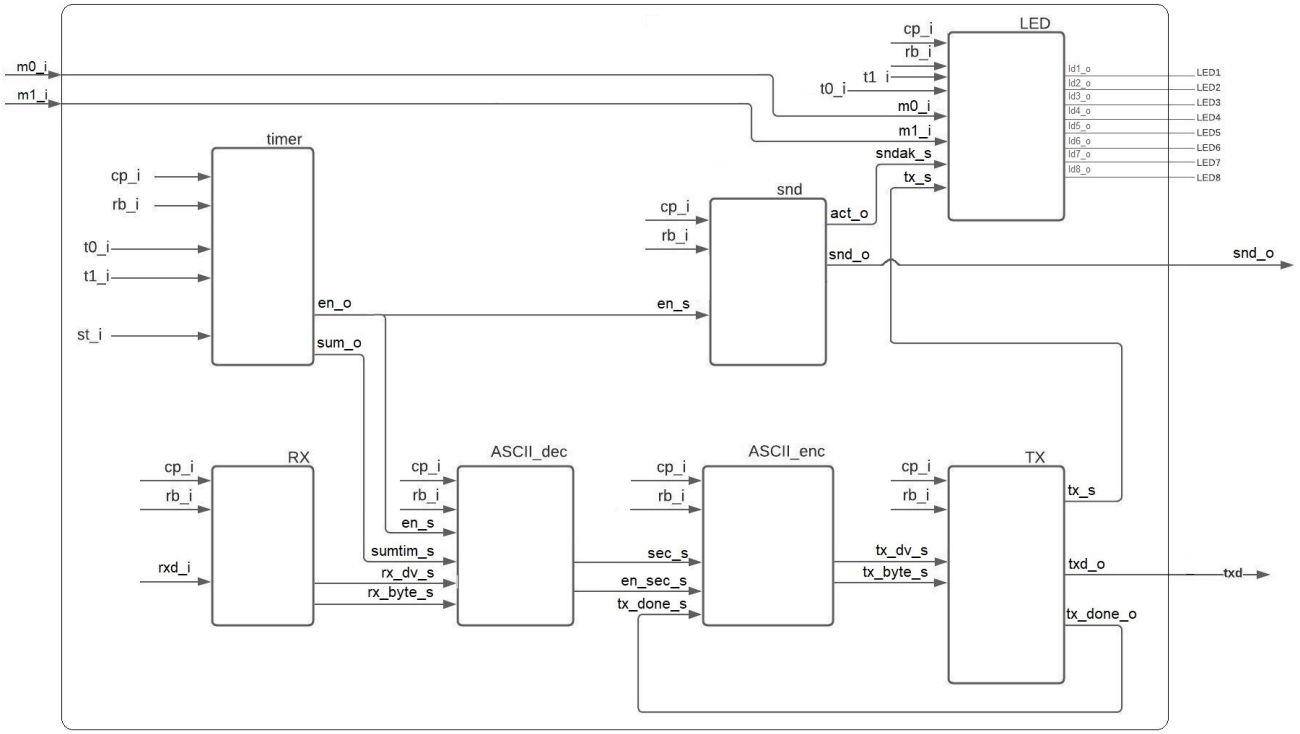

Figure 3: Block Diagram of the Top level

3.3 Description

3.3.1 General Description

The product is termed as Tea Bag controller. It is designed to measure the time taken for the completion of brewing. It indicates the brewing with the help of an alarm. There are also LEDs that indicates which mode it is in, one LED shows the reset button being pressed and another called heartbeat shows if the design is running.

3.3.2 Functional Description

3.3.2.1 timer

Whenever the sensor picks up the tea bag the st_i signal goes to 1 and starts the counting process of the timer. To be specific, the integer second_s increments every rising edge of the clock. And everytime it reaches its threshold (defined in generic) we increment the minute signal. The threshold for the minute signal is defined by the combination of t0_i and t1_i. The modmin register is given the constant depending on the combination of these two signals. And this is also sent to the ASCII decode component as an integer signal. When min_s reaches its threshold, enable_s goes to 1 which drives three other components. On a related note every counting integer signal resets itself whenever it reaches its threshold.

If st_i is zero i.e. the sensor doesn't pick up tea so the enable stays zero.

3.3.2.2 RX

RX is the receiver of the UART with inputs of reset and clock as well as a receiving serial port. For its output we have an 8 bit vector and a data valid signal that has a rising edge which lasts one clock cycle. The incoming bits are received into two registers sequentially so that we could avoid meta-stability issues. Here we have a latency of one clock cycle. The registers are named rxd_s and rxd2_s. By the means of using the finite state machine we designed the flow of receiving each bit accordingly and saving it to our 8 bit vector register with its name rx_byte_s.

The following is the state behavior :

- idle_st - reset state : We reset the byte register, the baud enable register and the data valid register. If incoming bit is zero we go to the check state.
- chk_st - check state : We check the middle of the start bit to make sure that the bit is still zero. If it is zero we go to baud reset state.
- baudrst_st - baud reset state : This state last only one clock cycle. It resets the baud register so that we could start counting and sampling every other bit in the middle to avoid reading unstable bit at the beginning of each state.
- bit_st - start bit : If baud counts to 1250 we go to bit 1. We use the register nx_s to flip through each state.
- bit1_st - bit 1 state : Samples first bit and waits for baud to count to 1250.
- bit2_st - bit 2 state : Samples second bit and waits for baud to count to 1250.
- bit3_st - bit 3 state : Samples third bit and waits for baud to count to 1250.
- bit4_st - bit 4 state : Samples fourth bit and waits for baud to count to 1250.
- bit5_st - bit 5 state : Samples fifth bit and waits for baud to count to 1250.

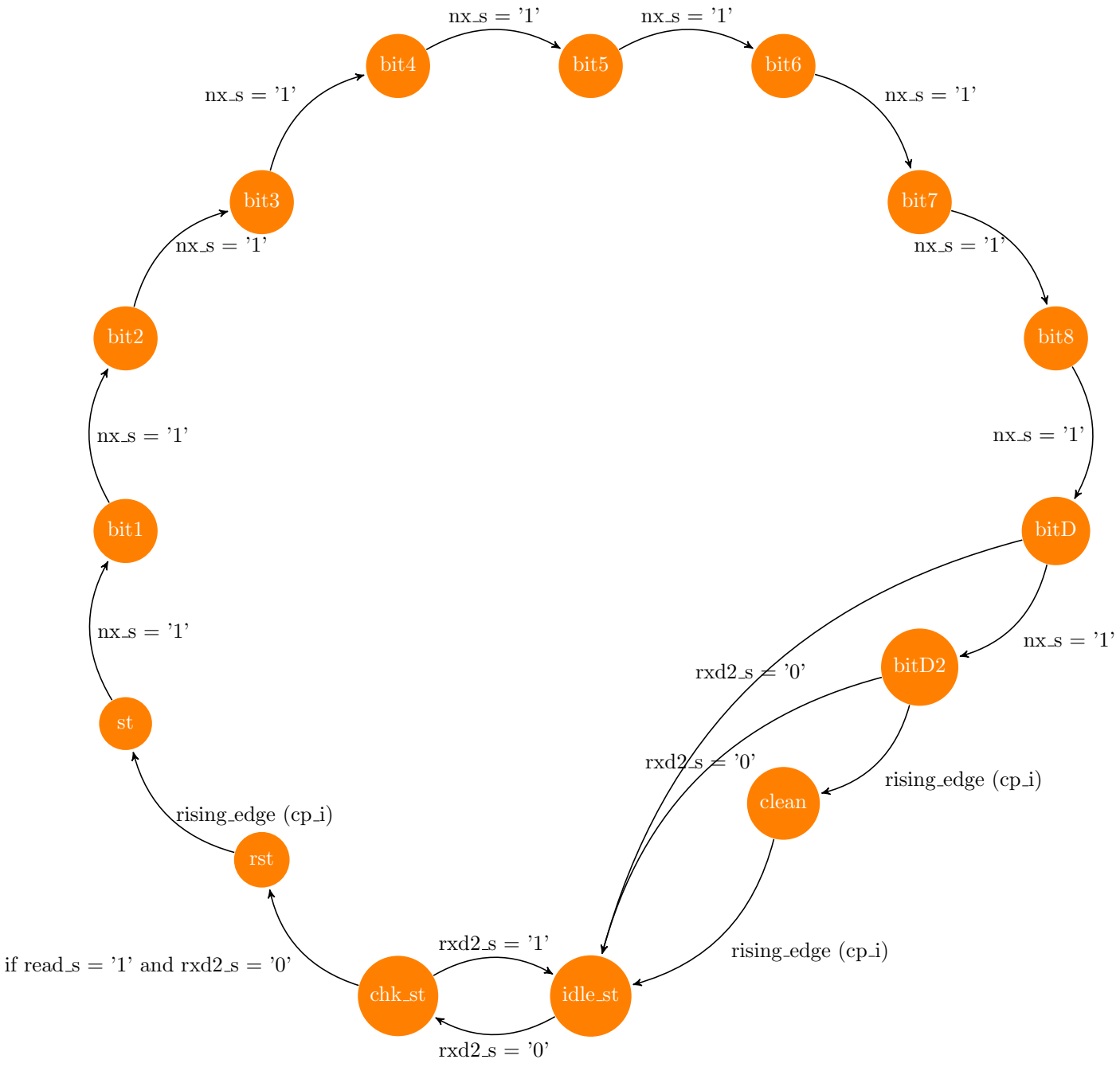

Figure 4: State flow graph of RX

- bit6_st - bit 6 state : Samples sixth bit and waits for baud to count to 1250.
- bit7_st - bit 7 state : Samples seventh bit and waits for baud to count to 1250.
- bit8_st - bit 8 state : Samples eight bit and waits for baud to count to 1250.
- bitD_st - bit D state : Checks the middle of stop bit, if its not zero goes to idle_st. If it is the stop bit,we count with baud up to 1250 we go to the next stop bit.
- bitD2_st - bit D2 state : Here checks the middle of stop bit again, if its zero goes to idle_st. If its the stop bit, we go to clean state.
- clean_st - clean state : Data valid goes up for one clock cycle, byte register and baud register resets.

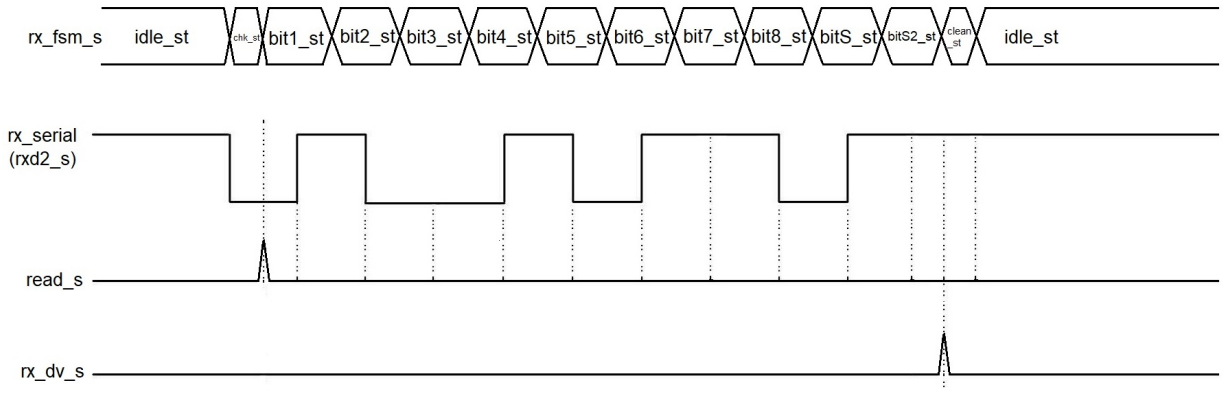

Figure 5: RX Timing Diagram

3.3.2.3 snd

The snd component waits for the enable from the timer. Whenever the enable from the timer goes up we store it in the register called en_s. With the register we make a rising edge detector. If rising edge is detected the sec_s register starts being incremented by 1 every clock edge and we give one to the act_s register so that we can enable the snd signal i.e. the snd_s signal takes the square wave generated by gen_s register. Everytime it reaches its threshold the len_s is incremented by 1 and when len_s reaches its threshold the register act_s goes to 0. The freq_s register independent from the enable increments by 1 every clock edge and flips the gen_s register every time it reaches its threshold. This is where the 1 kHz square wave is generated.

3.3.2.4 ASCII_dec

The ASCII decode component is used for receiving the bytes from the receiver of the TOA (Time of Arrival) of the tea bag, transferring them into seconds and summing them up into a single integer register. Then we pass the information of this register to the encoding part (ASCII encode component). Near the end of the summation process we also add the expected time of brewing of the tea bag. We accomplish this by means of a finite state machine, a procedure that converts hex codes into integers and multiple arithmetic operations like multiplication and addition inside the FPGA.

3.3.2.5 ASCII_enc

The ASCII encode component receives the seconds decoded from the ASCII decoder. It takes those seconds and it converts them into ASCII characters which are sent through the transmitter and to our PC. This is the brewing completion time. To accomplish this feat we do this with the help of multiple finite state machines and arithmetic operations such as division and modulo as well as a procedure that converts integer back into ASCII characters.

3.3.2.6 TX

The TX is the transmitter of the UART, it transmits 9600 baud and 8N2 bits. It does this by using a finite state machine and receiving already completed bytes. In this case the ASCII encoded time provided by the ASCII encoder.

This is following state behavior :

- idle_st - reset state : transmitter waits for the data valid (tx_dv_i) before it starts transmitting the byte in tx_byte_i.
- bit_st - start bit : The txt_data_s register takes the incoming byte that needs to be transmitted. Transmitter is transmitting the start bit.
- bit1_st to bit8_st - bit 1 to 8 state : transmitter is transmitting the first to eighth bit for respectively bits.
- bitD_st - bit D state : transmitter is transmitting the first stop bit
- bitD2_st - bit D2 state : transmitter is transmitting the second stop bit.
- clean_st - clean state : We clean the tx_data_s register, turn off the baud generator and we drive tx_done_s to 1 to indicate that transmission is finished and return to idle state by the next rising edge of the clock.

The State Diagram shows the flow of states for the transmitter.

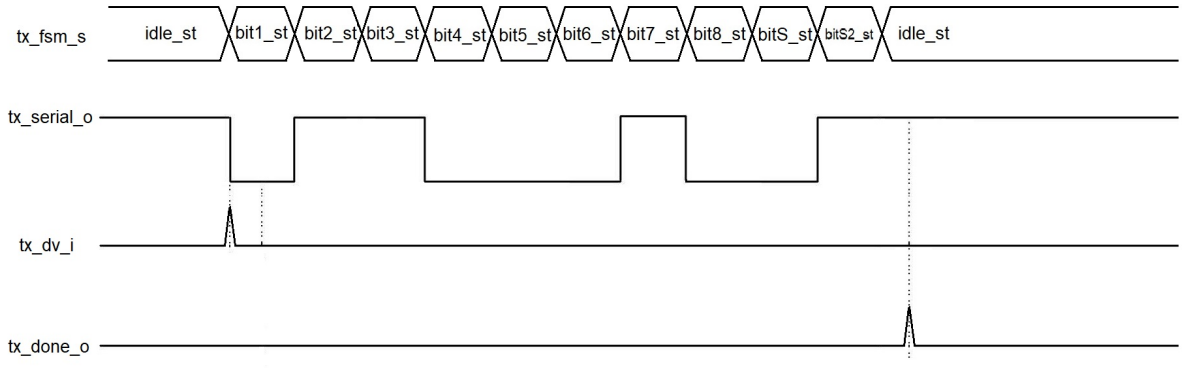

Figure 6: TX Timing Diagram

3.3.3 LED

The LED components controls the LEDs. It contains 8 LEDs.

- LED 1 : reset
- LED 2 - heart beat LED : Individual component instantiated inside the main LED block used for blinking every half second.
- LED 3 : mode 0 pin
- LED 4 : mode 1 pin
- LED 5 : timer mode 0 pin
- LED 6 : timer mode 1 pin
- LED 7 : active transition from transmitter
- LED 8 : active sound square wave

3.4 Port Description

The following table describes the ports and signals used.

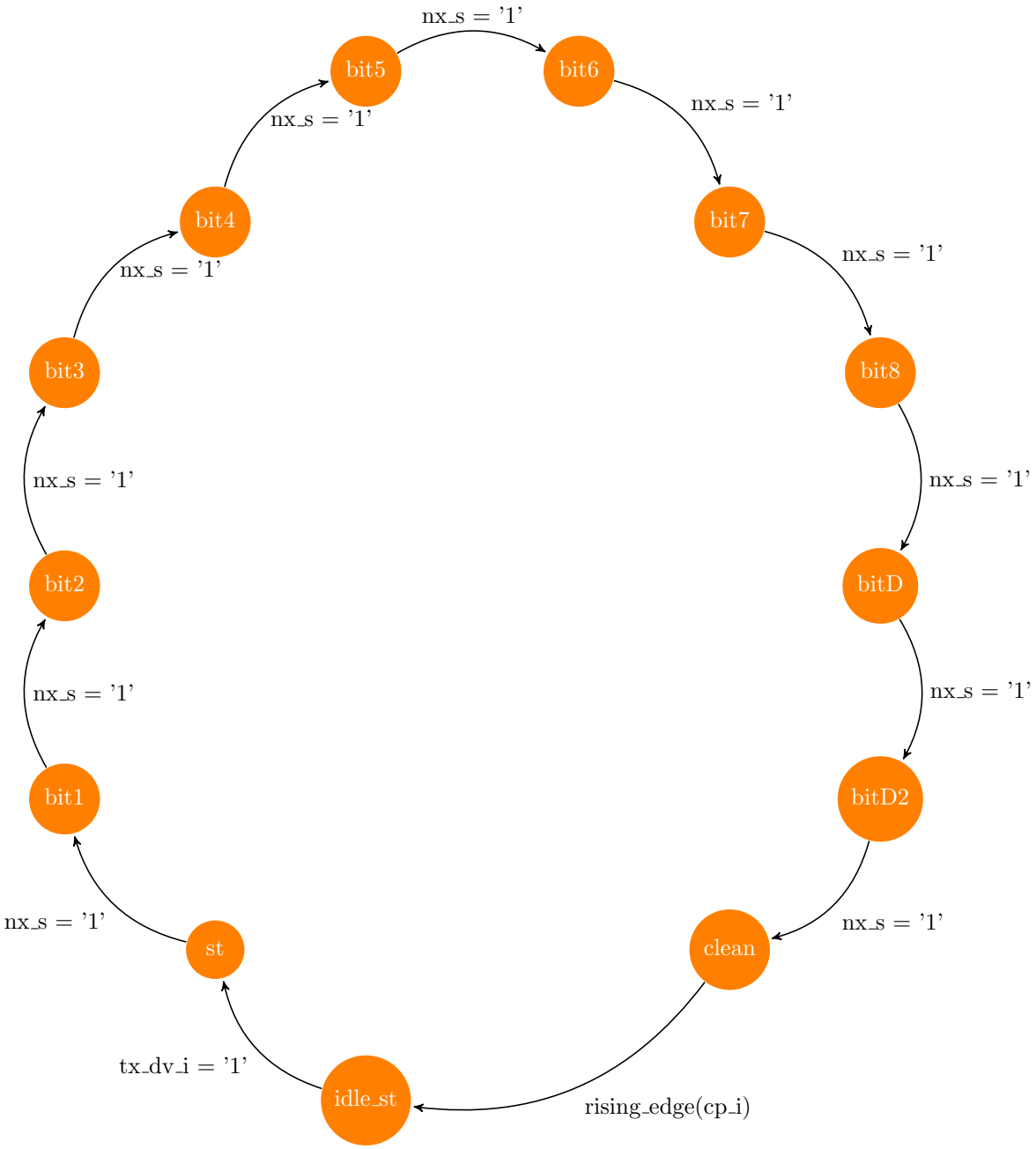

Figure 7: State flow graph of TX

Signal	Type	Description
Global		
cp_i	in	system clock
rb_i	in	low active asynchronous reset
External Interface		
- timer		
t0_i	in	tea brew time mode pin
t1_i	in	tea brew time mode pin
st_i	in	sensor input
en_o	out	enable for tea brew completion
sum_o	out	length of brew time in seconds
- RX		
rxd_i	in	incoming TOA (time of arrival) bytes; serial input of receiver
rx_dv_o	out	data valid output from receiver
rx_byte_o	out	ready byte from receiver
- ASCII_dec		
en_sum_i	in	enable from time for tea brew completion (take integer seconds of brew length)
sum_i	in	length of brew time in seconds
rx_dv_i	in	take ready byte from receiver
rx_byte_i	in	ready byte from the receiver
sec_o	out	brew length and time of arrival in seconds
en_sec_o	out	take seconds enable
- ASCII_enc		
sec_i	in	brew length and time of arrival in seconds
en_sec_i	in	take seconds enable
tx_done_i	in	feedback loop from transmitter when one byte is transmitted
tx_dv_o	out	start transmission of byte
tx_byte_o	out	the byte that is being transmitted to the transmitter
- TX		
tx_dv_i	in	enable byte transfer
tx_byte_i	in	the byte that is being transmitted
tx_active_o	out	transmission is active
tx_serial_o	out	serial output of transmitter
tx_done_o	out	transmission is finished
- snd		
en_i	in	enable from timer starts snd
snd_o	out	square wave for sound
act_o	out	sound is active
- LED		
m0_i	in	mode pin 1 for architecture choice ; LED 3 on
m1_i	in	mode pin 2 for architecture choice ; LED 4 on
t0_i	in	mode pin 1 for timer mode ; LED 5 on
t1_i	in	mode pin 2 for timer mode ; LED 6 on
snd_i	in	sound active from snd ; LED 7 on
tx_i	in	transmission active from transmitter ; LED 8 on
ld1_o	out	output for LED 1
ld2_o	out	output for LED 2
ld3_o	out	output for LED 3
ld4_o	out	output for LED 4
ld5_o	out	output for LED 5
ld6_o	out	output for LED 6
ld7_o	out	output for LED 7
ld8_o	out	output for LED 8

3.5 HW Interfaces

	Name used in module functional description	Chip level name
Connecting Signals	en_i	en_s
	act_o	sndak_s
	en_o	en_s
	sum_o	sumtim_s
	snd_i	sndak_s
	tx_i	tx_s
	rx_dv_o	rx_dv_s
	rx_byte_o	rx_byte_s
	tx_dv_i	tx_dv_s
	tx_byte_i	tx_byte_s
	tx_active_o	tx_s
	tx_serial_o	txd_o
	tx_done_o	tx_done_s
	rx_dv_i	rx_dv_s
	rx_byte_i	rx_byte_s
	sum_i	sumtim_s
	en_sum_i	en_s
	sec_o	sec_s
	en_sec_o	en_sec_s
	en_sec_i	en_sec_s
	tx_done_i	tx_done_s
	sec_i	sec_s
	tx_dv_o	tx_dv_s
	tx_byte_o	tx_byte_s

3.6 Product details

The chip being used is Altera 10M16SAU169C8G. [3]

- **Family Signature : 10 M -** Intel® MAX ®10
- **Member Code : 16 -** It has 16000 logic elements.
- **Feature Option : SA -** Single supply ; analog and flash features with RSU option.
- **Package type : U -** The package of the chip is Ultra Fine Line BGA (UBGA)
- **Package Code : 169 -** It has 169 pins with the dimensions 11mm x 11mm
- **Operating Temperature : C -** Commercial (T = 0°C to 85°C)
- **FPGA Fabric Speed Grade : 8**
- **Optional Suffix : G -** RoHS6

4 System View

4.1 Application System Description and Use Cases

The system can be used to accurately measure the time for tea brewing.

4.2 System Diagram

The below figure represents the system diagram. It shows the basic components used in the system and their connections with each other.

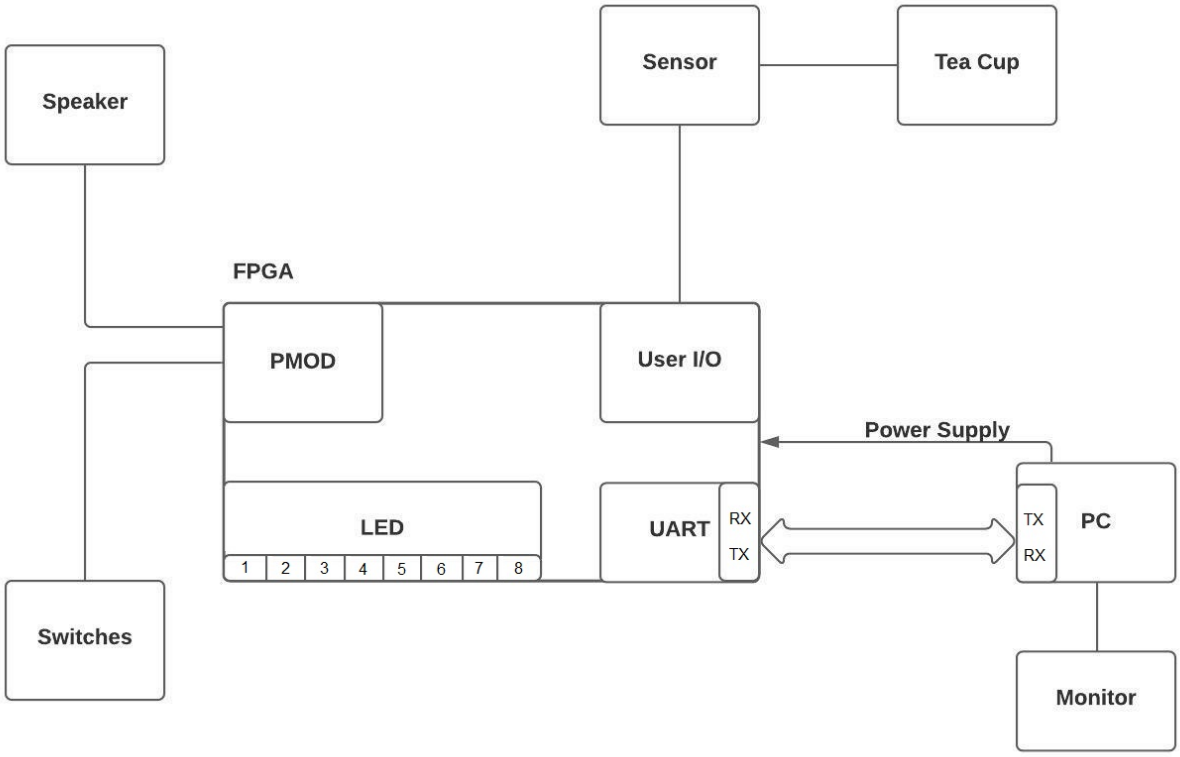

Figure 8: System Diagram

4.3 System Description

The system consists of the main part FPGA which gets the power supply from the PC. The FPGA comprises of PMOD, User Input/Output ports, LEDs and UART. The FPGA sends and receives signal from the PC with the help of transmitters and receivers. In the FPGA, the PMOD is connected to switches which can be used to give input signals and is also connected to the speaker for the alarm when brewing is done. A sensor is connected to the FPGA which is further connected with the tea cup to sense the tea bag. When the tea bag is placed the sensor senses it and send signals to the FPGA.

4.4 Compliances

The product has RoHS 6 compliance.

RoHS stands for Restriction of Hazardous Substances, and impacts the entire electronics industry and many electrical products as well. The original RoHS, also known as Directive 2002/95/EC, originated in the European Union in 2002 and restricts the use of six hazardous materials (lead, mercury, cadmium, hexavalent chromium, PBB's & PBDE's) [4] found in electrical and electronic products. All applicable products in the EU market since July 1, 2006 must pass RoHS compliance. [5]

RoHS 5/6 refers to compliance for 5 out of the 6 restricted substances (no compliance for lead (Pb)). Lead in very specific applications for Categories 8 and 9 is also exempted under Annex III for a few more years. [6]

5 Architecture Concepts Overview

5.1 Technology

Built on 55 nm TSMC Embedded Flash (Flash + SRAM) process technology. [3]

5.1.1 System Memory Concept

Simple and fast configuration - Secure on-die flash memory enables device configuration in less than 10 ms.

Memory Devices :

- 64MBit to 128 MBit external flash memory
- 64MBit to 256 MBit external SDRAM memory

It has User flash memory (UFM) and its key features are :

- User accessible non-volatile storage
- High speed operating frequency
- Large memory size
- High data retention
- Multiple interface option

5.2 Software Architecture [3]

- 4-input look-up table (LUT) and single register logic element (LE)
- LEs arranged in logic array block (LAB)
- Embedded RAM and user flash memory
- Clocks and PLLs
- Embedded multiplier blocks
- General purpose I/Os

6 Functional Block Overview

6.1 Bus Peripherals

- USB to USB micro : It is needed for JTAG programming.
- USB to RS232 converter : It is needed to communicate with UARTs.

6.2 Pin List

The below Table describes the top level pins [1]:

Ports	Pin name	I/O Standard	Desciption
Input ports			
cp_i	PIN_H6	3.3V LVTTL	12 MHz clock input
rb_i	PIN_E7	3.3V Schmitt Trigger	GPIO : General Purpose Input Output
m0_i	PIN_N3	3.3V Schmitt Trigger	PMOD Pin 5
m1_i	PIN_N2	3.3V Schmitt Trigger	PMOD Pin 6
t0_i	PIN_K2	3.3V Schmitt Trigger	PMOD Pin 7
t1_i	PIN_K1	3.3V Schmitt Trigger	PMOD Pin 8
st_i	PIN_E6	3.3V Schmitt Trigger	User button
rxd_i	PIN_M1	3.3V LVTT	PMOD Pin 4
Output ports			
ld1_o	PIN_A8	3.3V LVTTL	LED
ld2_o	PIN_A9	3.3V LVTTL	LED
ld3_o	PIN_A11	3.3V LVTTL	LED
ld4_o	PIN_A10	3.3V LVTTL	LED
ld5_o	PIN_B10	3.3V LVTTL	LED
ld6_o	PIN_C9	3.3V LVTTL	LED
ld7_o	PIN_C10	3.3V LVTTL	LED
ld8_o	PIN_D8	3.3V LVTTL	LED
snd_o	PIN_L3	3.3V LVTTL	PMOD Pin 2
txd_o	PIN_M2	3.3V LVTTL	PMOD Pin 3

Table 2: Pin List Table

Abbbreviations : LVTTL - Low Voltage Transistor-Transistor Logic , PMOD : Peripheral Module Interface

References

[1] www.arrow.com, "Max1000 user guide," https://shop.trenz-electronic.de/trenzdownloads/Trenz_Electronic/Modules_and_Module_Carriers/2.5x6.15/TEI0001/User_Guide/MAX1000%20User%20Guide.pdf, July 2017.

[2] A. Corporation, "Top-level design entity," https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/reference/glossary/def_top_level.htm#:~:text=A%20design%20entity%20that%20is%20the%20root%20of%20a%20design%20hierarchy.&text=A%20top%2Dlevel%20design%20entity,design%20entity%20for%20the%20project., 2005-2013.

[3] I. Corporation, "Intel® max® 10 fpga device overview," https://d2pgu9s4sfmw1s.cloudfront.net/DITA-technical-publications/PROD/PSG/m10_overview-683658-666776.pdf?Expires=1642504657&Key-Pair-Id=APKAJKRNIMMSNYXST6UA&Signature=4OnH9FtkE7AV~BMBhoAbxImqABkefXHvjlBU-Iorcpy4ELuocNfnJoN94o1hcoxkLHvc~Rwq~7fLq7qytA1vEuQGeU2rN_, 2021.

[4] M.-M. M. CORP., "Rohs frequently asked questions," https://www.mill-max.com/sites/default/files/external/assets/2017-08/ROHS-FAQ.pdf, 2022.

[5] RoHSGuide.com, "Rohs guide," https://www.rohsguide.com/, 2022.

[6] ——, "Rohs guide," https://www.rohsguide.com/rohs-faq.htm, 2005-2022.