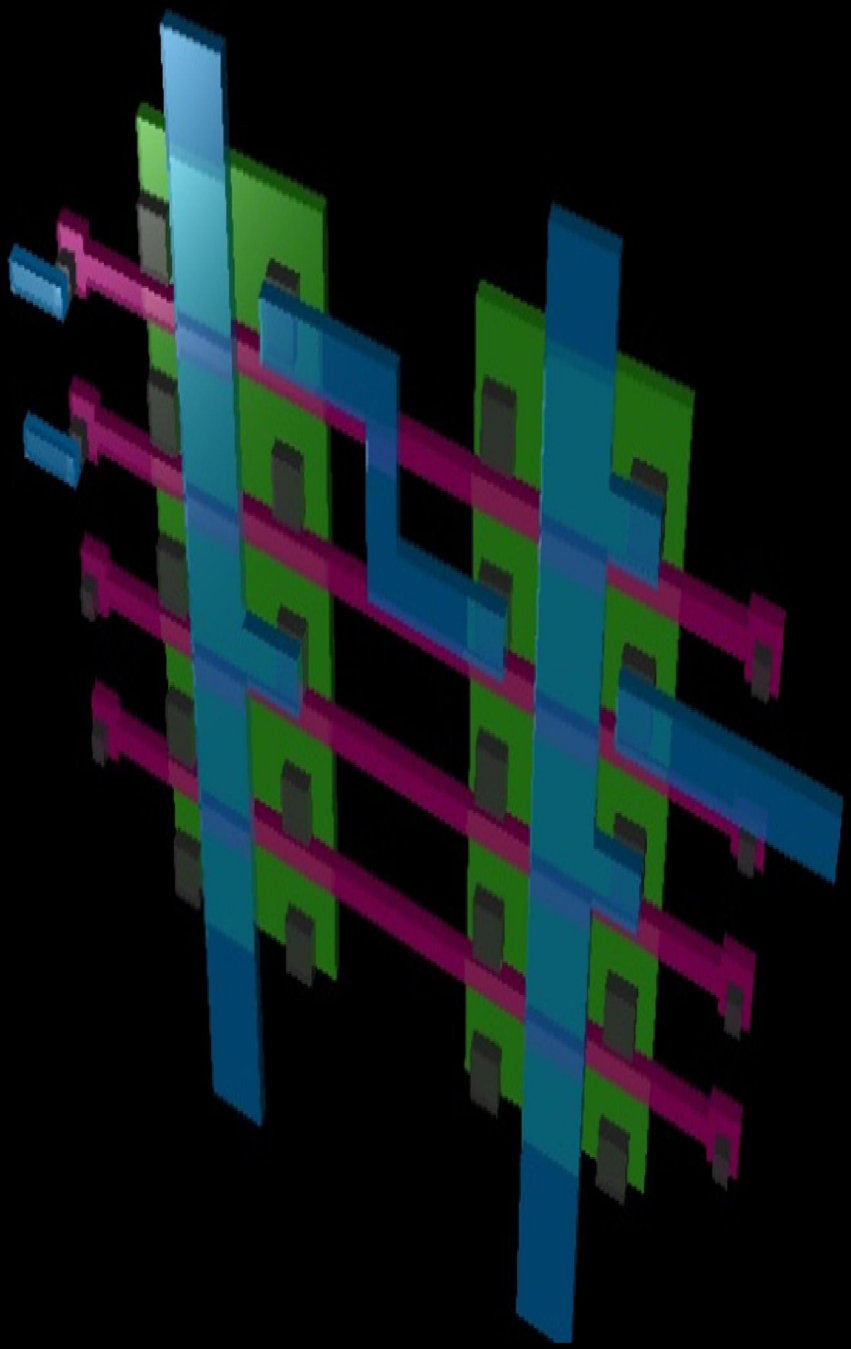
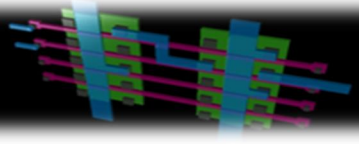
pipeline temperature detection system

An implementation OF UART BY Verilog



Contents



Abstract

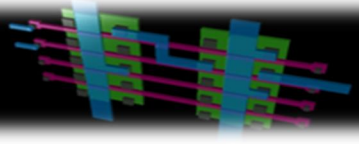
This document provides a full documentation and brief explanation for the implantation of a pipeline temperature detection system using Verilog.
Implemented by:

1. Ziad Abdelkarim
2. Rozan Magdy
3. Esrra Gaber
4. Marwa Medhat
5. Norhan ALaraby.

This is implemented as a phase1 final project for chipions'20 programme power by IEEE ISSCS ALEX SB and VLSI Alexandria section.
The document contains all the implemented system design, FSM, algorithms design. Even more, it includes brief implementation simulation.

Introduction

Nowadays, communication protocols among different electronics system is has involved various fields. Moreover, the universal asynchronous transmition and receiving protocol (UART) has various application. Therefore, implementing the UART driver module by HDL as Verilog can have much benefits as using it in the system discussed within this document. Which is the design of temperature detection and monitor device. The system provides multiple digital modules such as analog to digital converter, serial to parallel shift register, transmitter, receiver, parallel to serial shift register and seven segment display drivers.



System architecture

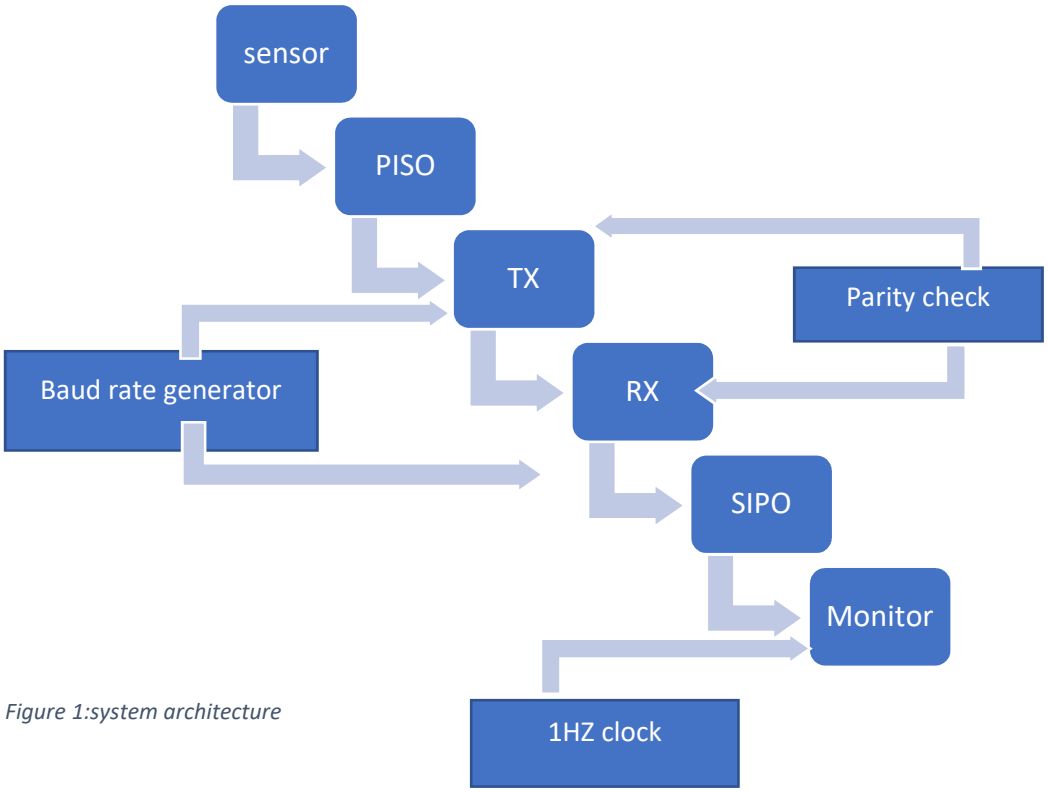


Figure 1:system architecture

Modules description

PISO

It's a parallel in serial out 8-bit shift register that convert the data sent by sensor to the data bus of the Transmitter module.

SIPO

It's a serial in parallel out 8-bit shift register that convert the data sent by sensor to the data bus of the Transmitter module.

TX

Transmitter module that is control the data bus from sensor to receiver.

RX

Receiver module that is control the data bus from transmitter to monitor.

1HZ clock

It is the clock gate of displaying monitor. The frequency divider that divided the system clock to produce the needed baud rate for both TX and RX.

Parity check

The result of XORing the system data to check the data corruptions.

Modules inputs and outputs

module	**inputs**	**Outputs**
PISO	Parallel Data, clk	Serial data
SIPO	Serial data, clk	Parallel data
TX	TXbaud, data, parity	Data
RX	Rxbuad, data, parity	Data
Baud rate generator	System clk	TX baud, Rx baud
1HZ clock	System clk	1HZ clk
Parity check	Data	XORing data bits
Digital monitor	data	Data display

Table 1:modules i/o

UART

the system implementation is done using the mainly the UART protocol where the data is send and received from sensor and display it on station. Our system services the full featured UART that mange the start bit and number of send data bits in addition to the parity check bits in each module.

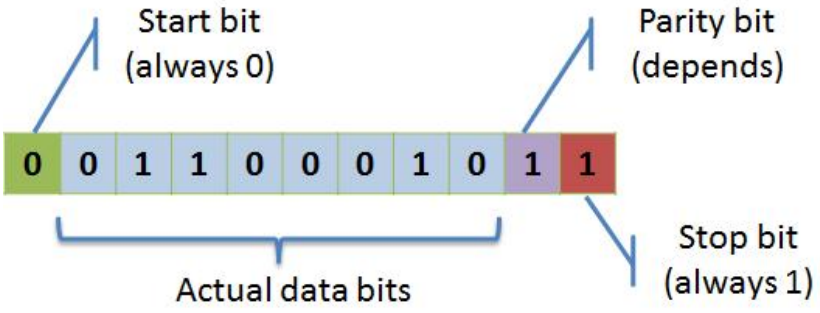


Figure 2:UART protocol

Calculations

The counter of the system is based on the 50 MHZ system clock to produce multiple baud rate with changing one parameter n

$$counter = \frac{50MHZ}{1200n} = \frac{41666}{n}$$

Equation 1:baud counter

Where to generate 9600 bud rate n = 8, and to produce baud rate for 16 times of the TX to RX

n =128.

And to initialize the communication every 15 mint. We make the TX transmit the data from sensor every special counter overflow.

$$\frac{1}{Z} \times \frac{n}{41666} \quad 900$$

Equation 2:Z parameter

Where n=8 as mentioned before to produce the required baud rate.

Over sampling

The over sampling system is used by RX module to determine the data received is correct and avoid glitches. As the system provide the RX clock 16 times the TX clock and the data is sampled at the middle of the received bit to make sure of the data sent. The baud rate ratio can be managed through controlling of the value of parameter n indicated in pervious section.

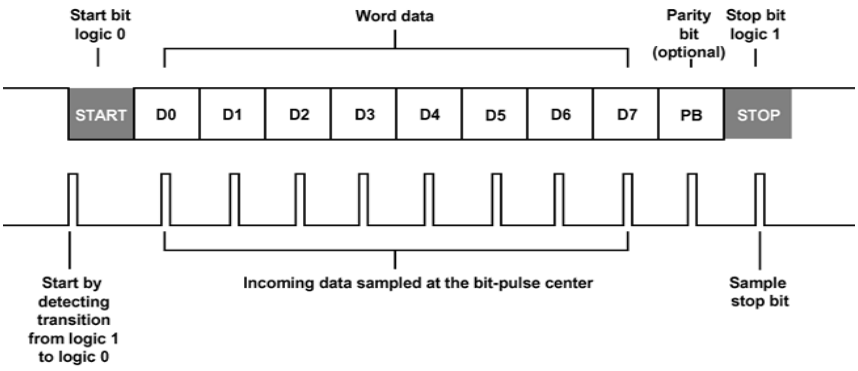


Figure 3:oversampling

FSM

Transmitter module

The various states indicated in the FSM is the states of transmitter module. Where S0 indicate the rest state where system is initialized, S1 is the ideal state when the bus is Hight TX=1 indicating no data to transfer, S2 is the state of sending the start bit to initialize communication; and it started after counter1 reaches z which is the number of ticks in the 15 mints according to transmitter baud rate (Z according to equation2). S3 is the state of sending data on the bus (TX=data), it maintains 8 bits of data counted by counter2. S4 is the parity check state where the data sent on the bus is the XORing value of sent data. And it maintains I bit. S5 is the stop communication bit where the bus is released high again.

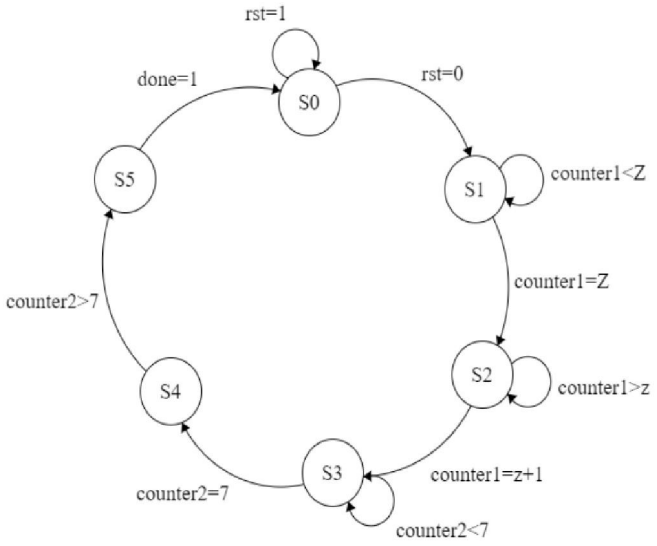


Figure 4:TX FSM

Receiver module

The various states indicated in the FSM is the states of receiver module. Where S0 indicate the rest state where system is initialized, S1 is the ideal state when the bus is Hight RX=1 indicating no data to receive. S2 started when the bus is taken to low RX=0 which indicate the communication star. S2 is maintained for 8 bits counter to sample the data at the middle of the start bit. S3 and S4 represent the stats of data sampling where the 8 bits of data is sampled by looping by bitnumber counter 8 times between S3 and S4. S3 represent the single bits sampling by 16 counts to sample that bit at its middle.so as to S5 which represent the sampling of parity pit by 16 counts of counter3. S6 is the state of indicating the stop communication.

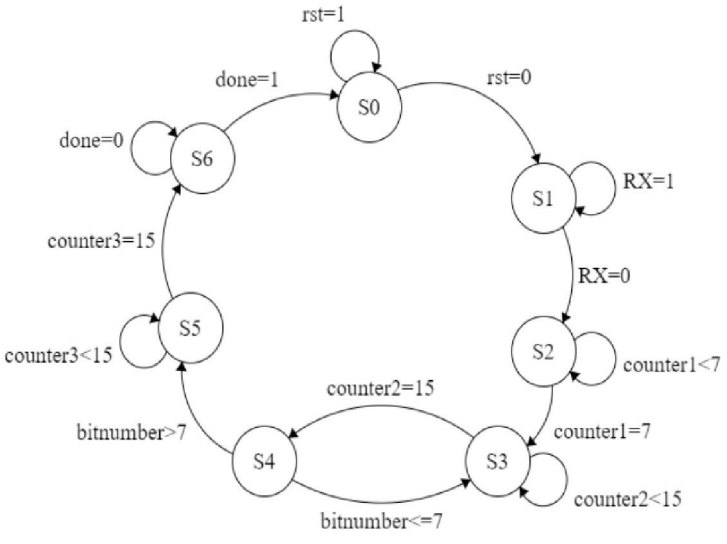


Figure 5:RX FSM

Baud rate generator module

The various states indicated in the FSM is the states of baud rate generator module. Where S0 indicate the rest state where system is initialized, S1 is the have cycle of frequency where its value is high. It is maintained for Z counts which is half the number of bit/second needed to produce the needed baud rate (calculate in equation1). S2 is the have cycle of frequency where its value is low. It is maintained for 2Z counts which is complementary half number of bit/second needed to produce the needed baud rate.

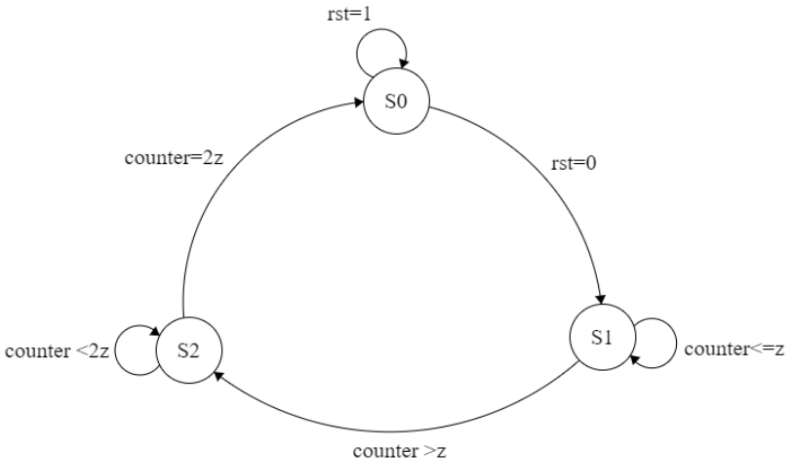


Figure 6:Baud rate generator FSM

Baud rate generator

Baud rate generator module can be implemented by more than one logic sequence either finite state machine or frequency divider. The frequency divider is preferable due to its reliability in testing and simple coding.

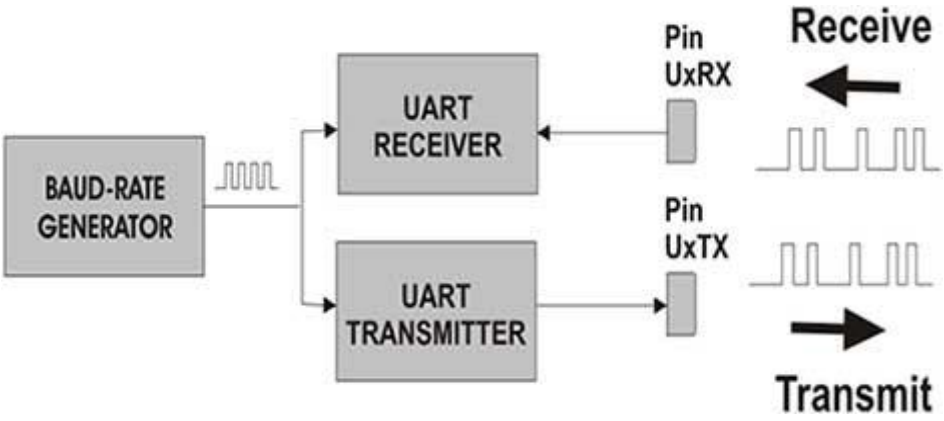


Figure 7:Baud rate generator divider

Digital monitor

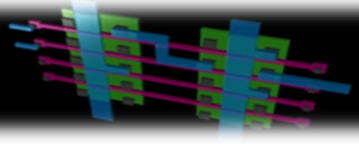
In the digital monitor display, we use the algorithm of double dabble to convert binary numbers into binary coded decimal to display on seven segment and LCD panel. This module doesn't need clocking as its combinational. The algorithm operates as the original number is stored in register that n bits wide its BCD representation: $n + 4 * ceil(\frac{n}{3})$ bits will be enough. It takes a maximum of 4 bits in binary to store each decimal digit. Then partition the scratch space into BCD digits (on the left) and the original register (on the right).

The scratch space is initialized to all zeros, and then the value to be converted is copied into the "original register" space on the right. The algorithm then iterates *n* times. On each iteration, the entire scratch space is left-shifted one bit. However, *before* the left-shift is done, any BCD digit which is greater than 4 is incremented by 3. The increment ensures that a value of 5, incremented and left-shifted, becomes 16, thus correctly "carrying" into the next BCD digit.

The double-dabble algorithm, performed on the value 243_{10}, looks like this:

```
0000 0000 0000   11110011   Initialization
0000 0000 0001   11100110   Shift
0000 0000 0011   11001100   Shift
0000 0000 0111   10011000   Shift
0000 0000 1010   10011000   Add 3 to ONES, since it was 7
0000 0001 0101   00110000   Shift
0000 0001 1000   00110000   Add 3 to ONES, since it was 5
0000 0011 0000   01100000   Shift
0000 0110 0000   11000000   Shift
0000 1001 0000   11000000   Add 3 to TENS, since it was 6
0001 0010 0001   10000000   Shift
0010 0100 0011   00000000   Shift
   2    4    3
       BCD
```

Figure 8:double dabble algorithm



Digital clock

The digital clock display is used to indicate the system timing and display it on the digital monitor. It works with the clock gating of 1HZ within cases of all timing increments from seconds to mints to hours and months with taking on consideration the different days of each month. The out but is send to BCD to be displayed into seven segments

The Accuracy of the system according to 8-bits ADC and temperature sensor

The 8 bits **ADC** represent a binary number between 0 and 255.

If the ADC's reference voltage is 5V, it will split up its 256 possible values evenly across 0V to 5V.

0V will read as 0 (00000000 in binary).

5V will read as 255 (11111111 in binary).

A voltage between 0V and 5V will read as a linear function between 0 and 255.

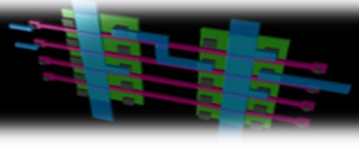
The smallest possible voltage changes an 8-bit ADC can measure is called its accuracy.

The accuracy reference voltage by total number of unique binary values.

$$5V/256 = 0.0195V = 0.02$$

Equation 3:accuracy of system

These calculations show us that if the voltage changes from 4 V to 4.01 V the ADC system will not detect this variation in the voltage and because the smallest step that can be detected is 0.02 not 0.01.



Appendix:

Table of figures

Cross references

- D Akhila et al, International Journal of Computer Science and Mobile Computing, Vol.5 Issue.12, December- 2016, pg. 38-47

- Mohd Yamani Idna Idris, Mashkuri Yaacob, Zaidi Razak, “A VHDL Implementation of UART Design with BIST capability”.

- Parul Gupta et al, International Journal of Advance Research in Science And Engineering “A REVIEW PAPER ON DESIGN AND SIMULATION OF UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER ON FIELD PROGRAMMABLE GATE ARRAY USING VHDL”.