International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 29987-29998

http://www.ripublication.com

Implementation of Low Pin Debug (LPD) Interface For Data Acquisition In Automotive Applications

S.M.Dhanya
Student of M.E VLSI Design, Sri Ramakrishna Engineering College
Coimbatore, India, dhanya2392@gmail.com
N. Kirthika
Assistant Professor, Department of ECE (PG- VLSI Design)
Sri Ramakrishna Engineering College
Coimbatore, India, kirthika.natarajan@srec.ac.in

Abstract

In data acquisition system in automotive application, there is a need to acquire real time data from Electronic Control Unit (ECU) when it is in operating conditions. Traditionally some techniques like KWP2000 and CCP are used to acquire the real time data from ECU. But these techniques need ECU support to acquire the real time data which influences the ECU functionalities. Also data throughputs of these techniquies are very low. Nowadays ECU microcontrollers are coming with user debug interfaces like JTAG, AUDR etc. Low Pin Debug interface is one of such debug interface. To improve throughput and reduce impact on ECU functionalities, this debug interface can be used to access the internal microcontroller memory and resources. The Low Pin Debug Unit (LDU) interface based data acquisition method is developed to reduce the influence of data acquisition functions on ECU functionalities and throughput.

Keywords: Low pin Debug Interface (LPDI), Electronic Control Unit(ECU), LPD Interface Contoller.

Introduction

Data Acquication is an important technique to acquire the data from the system. Here we are going to acquire a ECU data. Earlier the techniques like KWP2000 and CCP are used to acquire the real time data from ECU. But these technique used the resourses of ECU. So the ECU based microcontroller were introduced. It has a separate RAM and it is independent of ECU. The ECU microcontroller has interfaces such as JTAG, AUDR and LPD. The interface is used for connecting microcontroller and the FPGA to acquire the data. The interface used here is low pin debug

interface.The advantage of using the low pin debug interface is that it has lower number of pins compared to JTAG interface. So to acqire the data from the microcontroller to the FPGA we need to design LPD microcontoller with the hardware requirements so that the data's can be accessed. Initially we design a transmitter, receiver and a top module. Where the transmitter and receiver are used for the transmission and reception of the data's. Whereas the top module controlls the transmission and reception of data.

LPD Interface Pins:

The LPD interface has following pins

- LPD CLK
- LPDI
- LPD CLKOUT
- LPDO

LPD CLK

This is a unidirectional input pin for the low pin count debug clock (LPD clock). A fixed-speed/non-stop clock from the external debugger is input on it. The low pin count debug clock functions as the basic communication receive clock.

LPDI

This is a unidirectional data input pin. Read/write transfer commands, write data, special requests, etc., are input from the external debugger in synchronization with the low pin count debug clock. The input data is sampled in synchronization with the rising edge of the low pin count debug clock.

LPDCLKOUT

This is a unidirectional output pin for the low pin count debug clock output (LPD clock output). The low pin count debug clock output (LPD clock output) functions as the basic communication transmit clock. The output data (transmit data) is output in synchronization with the rising edge of the low pin count debug clock output (LPD clock output).

LPDO

This is a unidirectional data output pin. Read data responses, special responses, etc., to the external debugger (development tool) are output on it.

Communication Format

We need to design a overall system. First we design a transmitter for transmission of the data from the FPGA to microcontroller, receiver for receiving the data from the microcontroller and overall module to control both the transmitter and receiver. There are two types of frames for communication with the LPD interface. They are Frame B and Frame H. Frame B consist of 8 bit data, one start bit and two stop bit

wheras frame H consist of 16 bit data, one start bit and two stop bit. The stop bit should be one and start bit should be zero. The format of data of frame B and frame H to be transmitted to low pin debug interface is shown in the figure 1 and 2 respectively.

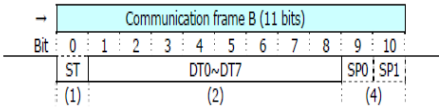

Figure 1: Communication Frame B

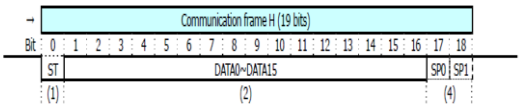

Figure 2: Communication Frame H

Transmitter

The default frame format is used which consist of 8 or 16 bit data, one start bit, zero parity bit and two start bit. The transmitter has two main parts combinational and sequential. The start bit must be written as zero and the stop bit must be written as one. When the reset is enabled it remains in idle state and the initialization of the signals is done. When the reset is disabled and the transmission signal is enabled the data starts transmitting the signal. The state diagram of the transmitter is shown in the figure 3.

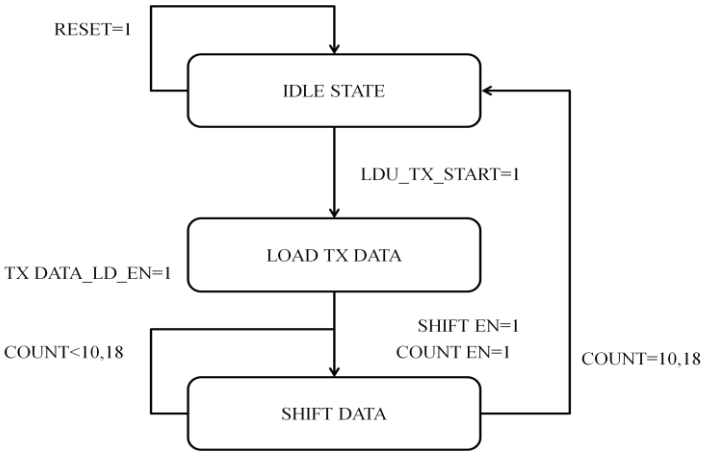

Figure 3: State Diagram of Transmitter

When reset is disabled it goes to next state. When reset is disabled and the present state is idle state then the next state will be data load. When frame_tx_H is zero 8-bit

is transmitted if it is one 16 bit is transmitted. The data load makes the load signal high and the data is loaded into the shift register. Where the concatenation operation takes place and no shifting is done. During concatenation the start bit (i.e. the zeroth bit) is set as zero and the stop bit (i.e. the last two bits) are set as one.

After the data is loaded then it enables the data shift signal and the counter signal. When the data shift signal is enabled the data starts shifting in the shift register. It is in the concept of parallel to serial shift. The counter starts counting when the counter signal is enabled. The counter and shift register will be running parallel. Once the shift it done it goes to the idle state. The transmission status indicates whether the transmission is taking place or not. If the transmission is taking place then the signal will be high or the signal will be low. The Block diagram of the transmitter is shown in the figure 4.

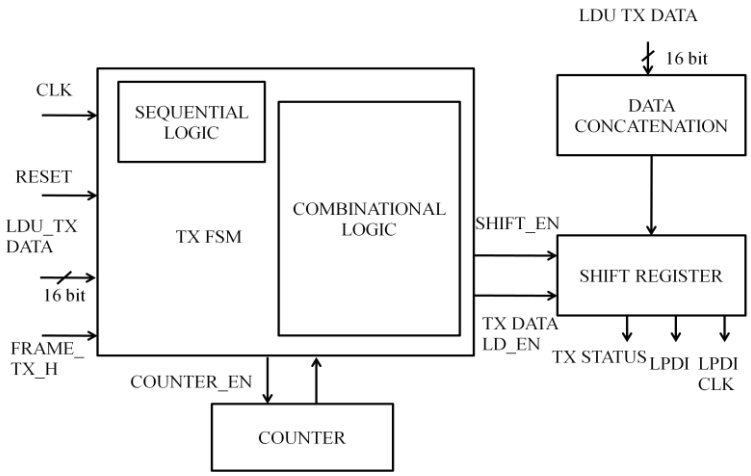

Figure 4: Block Diagram of Transmitter

Receiver

The receiver also has two parts sequential and combinational. The data of the receiver is received one by one in a serial manner. When the reset is enabled it remains in ideal state and the initialization of the signal is done. When the reset is disabled it goes to next state, then the input is received from the lpdo. When frame_rx_H is zero then 8 bit is transmitted or else 16 bit is transmitted. Then it is checked whether the start bit is detected or not. If the start bit is detected the data will be received and goes to the data shift state otherwise it remains in idle state. The shift register used is serial to parallel shift register. The state diagram of the receiver is shown in the figure 5.

The data of 11 bits are received for 8 bit and data of 19 bit are received for 16 bit data. After the data's are received it goes to the load state where the 8 bit or 16 bit ldurxdataout is received eliminating the start and stop bit. Then it goes to the next state stop bit detection. It checks whether the stop bit is detected. If it is detected then the error status will be zero otherwise it will be high. Then after the completion of

receiving 11 bit or 19 bits data rx status is made low when the data is being received the rx status will be high. The Block diagram of the receiver is shown in the figure 6.

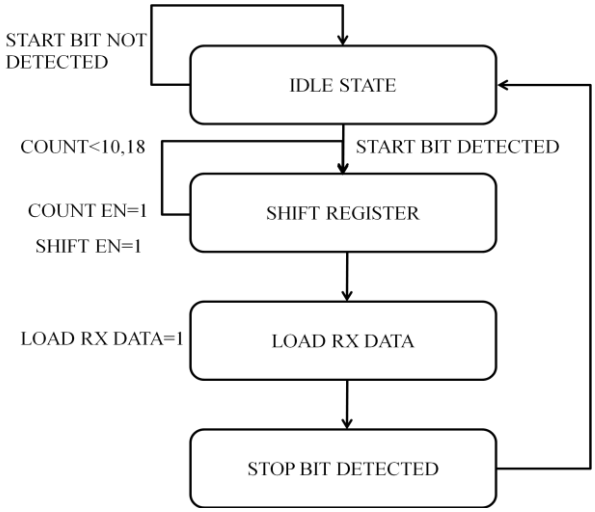

Figure 5: State Diagram of Receiver

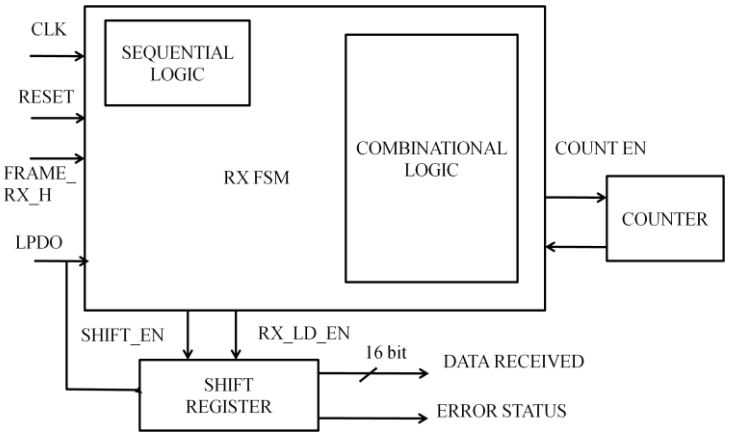

Figure 6: Block Diagram of Receiver

LPD IF Controller

It consist of finite state machine, multiplexer, counter, transmitter and receiver. The finite state machine has number of states. Signals used are start, clock, reset, address, read data and write data. The write data to be transmitted through the transmitter is given by the input write data and the read data is transmitted from the receiver is given as output in the read data. The overall block for LPD IF is shown in the figure 7.

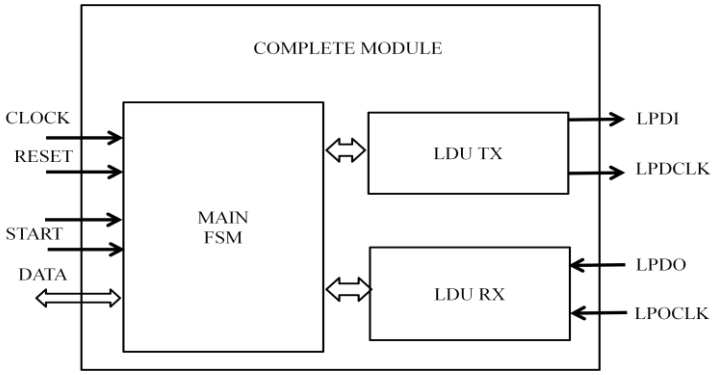

Figure 7: LPD IF Controller

The flow for data acqisation is based on the flow shown in the flow chart. The flow start with initiallization where the connection of the LPD is establised. During the swith Frame B to Frame H the data is converted from 8 bit to 16 bit. So that the data from the MCU can be accessed. Activate the DCU to access the register from the DCU. Verify the ID authentiation to know that you are authorised user. The CPU activation so that the data's are accessed.The complete flow of state machine is shown in the figure 8.

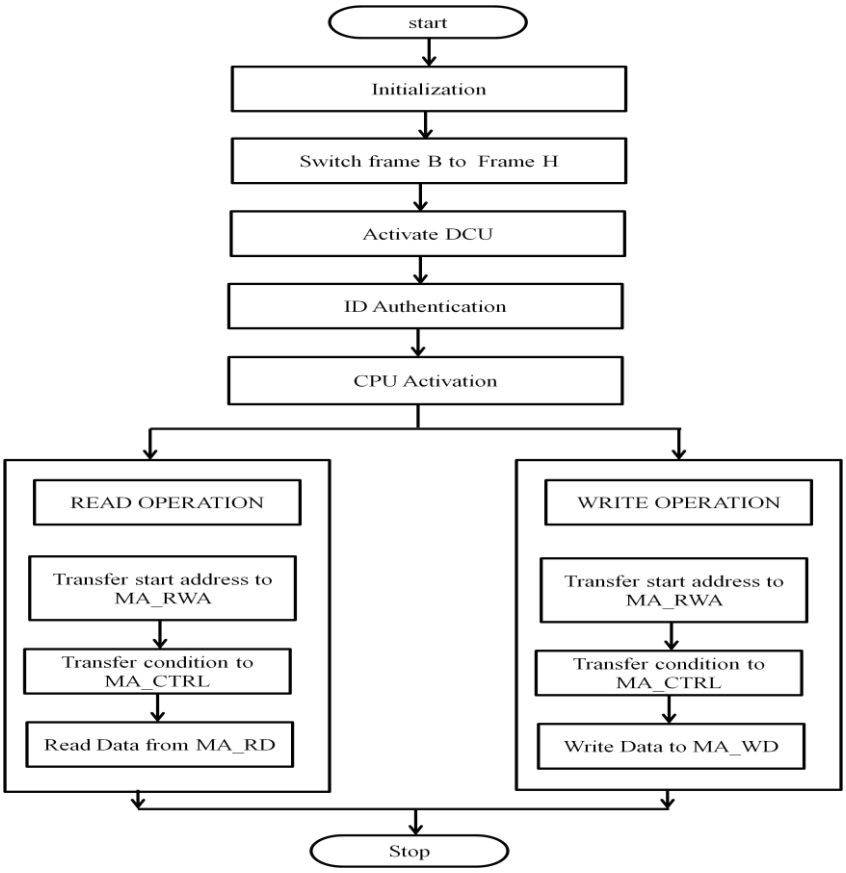

Figure 8: Complete Flow of Read and Write Operation

During the read and write opearation the transfer address is set in the MA_RWA register. Based on the conditions in the MA_CTRL reg. It is either read or written. Then the process is completed.

Results and Discussions

The results of the transmitter, receiver and the LPD IF controller is shown in this section.

Transmitter Output:

The transmits data from tool to MCU. Since the interface is serial interface. The parallel data is converted to serial data with repect to clock. The output of both 8 bit and 16 bit data are shown below. tx_status bit is used to check whether the transmiter is free or is under some operation.The output of the transmitter of Frame B and Frame H is shown in the figure 8 and 9.

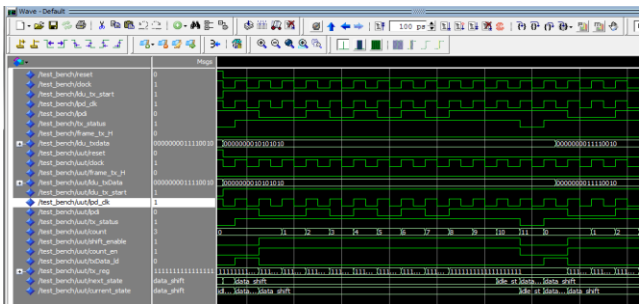

Figure 8: Transmitter Output of 8 bit data

Figure 9: Transmitter Output of 16 bit data

Receiver Output

It receives the data serially and convert it to parallel data and displays. It may be 8 bit or 16 bit based on the frame_rx_H. The output of the receiver 8 bit and 16 bit is shown in the figure 10 and 11.

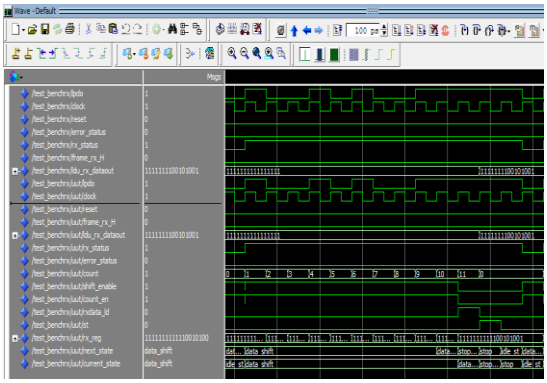

Figure 10: Receiver Output of 8 Bit Data

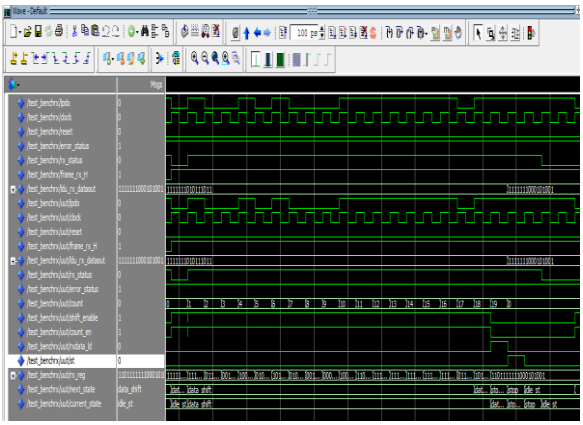

Figure 11: Receiver Output of 16 Bit Data

LDU Controller

The transmitter first transmits the command frame from the tool to MCU. Based on the command frame it decides read or write operation and also access the register. If it is read the receiver receives the data frame and sends to the main FSM. If it is write operation it sends the data from the transmitter to the MCU. The read and write operation from the LDU registers are shown in the figure 12 and 13 repectively.

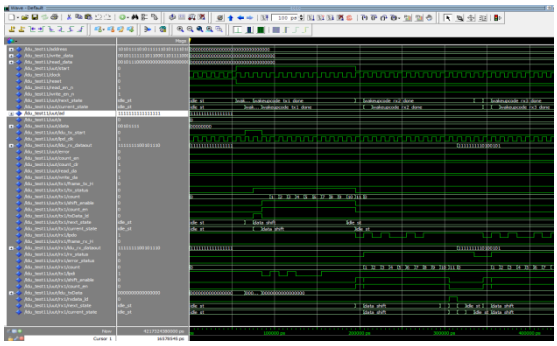

Figure 12: Read Operation

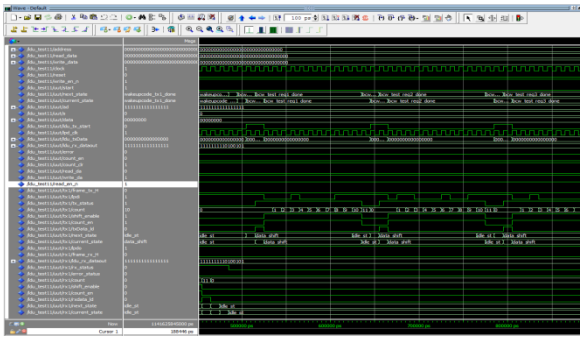

Figure 13: Write Operation

Flow Summary	
Flow Status	Successful - Wed Apr 29 17:06:19 20
Quartus II 64-Bit Version	14.1.0 Build 186 12/03/2014 SJ Web
Revision Name	ldu_interface
Top-level Entity Name	ldu_interface
Family	Cyclone IV E
Device	EP4CE30F29C6
Timing Models	Final
Total logic elements	678 / 28,848 (2 %)
Total combinational functions	676 / 28,848 (2 %)
Dedicated logic registers	283 / 28,848 (< 1 %)
Total registers	283
Total pins	98 / 533 (18 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 14: Flow Summary of LDU Controller

Flow Status	Successful - Wed Apr 29 16:56:13 2015
Quartus II 32-bit Version	11.1 Build 173 11/01/2011 SJ Full Version
Revision Name	VF_JTAG
Top-level Entity Name	VF_JTAG
Family	Cyclone IV E
Device	EP4CE30F29C6
Timing Models	Final
Total logic elements	1,240 / 28,848 (4 %)
Total combinational functions	1,165 / 28,848 (4 %)
Dedicated logic registers	618 / 28,848 (2 %)
Total registers	618
Total pins	282 / 533 (53 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 15: Flow Summary of JTAG controller

The comparison result for the JTAG and LDU is shown in the table 1. The Hardware result is shown in the figure 16. The flow summary of the JTAG and LDU controller is shown in the figure 14 and 15 respectively.

Table 1 Comparision of JTAG and LDU Controller

Parameters	**JTAG**	**LDU**
Area	4%	2%
Pins	282	98
Registers	618	283
Throughput	451 KBytes/s	1457 KBytes/s

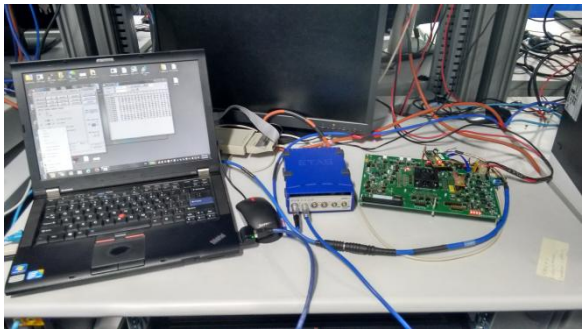

Figure 16: Hardware Implementation of LDU Controller

Conclusion

The LDU controller is designed in VHDL and simulated using MODELSIM. They are synthesised using Quartus II 14.1 software. The implementation of the low pin Debug interface has been done in FPGA based hardware successfully and the results are obtained. The data has been acquired from the Microcontroller through the Low Pin Debug Interface. It consumes 2% of total area and requires 98 pins. This technique reduces the impact of ECU.The throughput of the system is increased by 31% compared to JTAG interface.

References

[1] Functional specifications of G3 generation low pin count debug unit function specification.
[2] G3 Generation memory access unit function specification.
[3] John Andrews and Bill Aronson (et.al.) “IEEE Standard Test Access Port and Boundary-Scan Architecture” IEEE Computer Society 23 July 2001.
[4] Lee Whetsel “A High Speed Reduced Pin Count JTAG Interface” IEEE International Test Conference, 2006
[5] Xuhui Chen and Hongyun Yang “Design And Implementation Of A Single-Chip Arm-Based USB Interface JTAG Emulator” IEEE computer society, 2008.