Power Management Bus

Reference Design

FPGA-RD-02097-1.2

December 2019

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

1. Introduction

The Power Management Bus (PMBus) is an open standard protocol that was defined as a means to communicate with power conversion and other devices. As an industry standard serial communication interface based on SMBus protocols, it is viewed as an extension of the System Management Bus and supports many new functions. Like SMBus, the PMBus protocol defines three layers, the Physical Layer, the Data Link Layer and the Network Layer. This design contains a WISHBONE interface and focuses on the implementation of the Data Link Layer protocol. It is convenient to connect this design with a microcontroller which implements the Network Layer protocol.

This design is based on Lattice reference design I2C Master with WISHBONE Bus Interface (FPGA-RD-02072). It is available in both Verilog and VHDL languages.

2. Features

- Compatible with PMBus specification
  - Multi-master operation
  - Software-programmable SDL clock frequency and 400KHz allowed
  - Clock stretching and wait state generation
  - Timeout monitor for the master
  - Interrupt flag generation
  - Arbitration lost interrupt, with automatic transfer cancellation
  - Bus busy detection
  - Supports 7-bit addressing mode
- Compliant with WISHBONE specification
  - Compliant with WISHBONE Classic interface, Revision B
  - All output signals are registered
  - Two-cycle access time
  - A non-WISHBONE compatible signal, arst_i, is an asynchronous reset signal provided for FPGA implementations

3. Functional Description

The PMBus master core supports the critical features described in the PMBus specification and is suitable for most applications involving PMBus slave control. The design responds to the read/write cycles initiated by the microcontroller through the WISHBONE interface. It provides the correct sequences of commands and data to the PMBus slave device and then transfers the required data from the PMBus slave device through the two open-drain wires. The PMBus master with WISHBONE interface offloads the microcontroller from needing to administrate many details of the PMBus commands and operation sequences.

Table 3.1. lists the I/O ports of the design. Signals ending in “_i” indicate an input and those ending in “_o” indicate an output. All signals on the WISHBONE side are synchronous to the master clock. The two PMBus wires, scl and sda, must be open-drain signals and are externally pulled up to Vcc through resistors.

Table 3.1. Pin Descriptions

Signal	Width	Type	Description
WISHBONE Interface			
wb_clk_i	1	Input	Master clock
wb_rst_i	1	Input	Synchronous reset, active high
arst_i	1	Input	Asynchronous reset
wb_adr_i	3	Input	Lower address bits
wb_dat_i	8	Input	Data towards the core
wb_dat_o	8	Output	Data from the core
wb_we_i	1	Input	Write enable input
wb_stb_i	1	Input	Strobe signal/core select input
wb_cyc_i	1	Input	Valid bus cycle input
wb_ack_o	1	Output	Bus cycle acknowledge output
wb_inta_o	1	Output	Interrupt signal output
PMBus Interface			
SCL	1	Bi-directional	Serial clock line
SDA	1	Bi-directional	Serial data line
SMBA	1	Input	Alert signal, active is low
CONTROL	1	Output	Control signal, active is low

4. Design Module Description

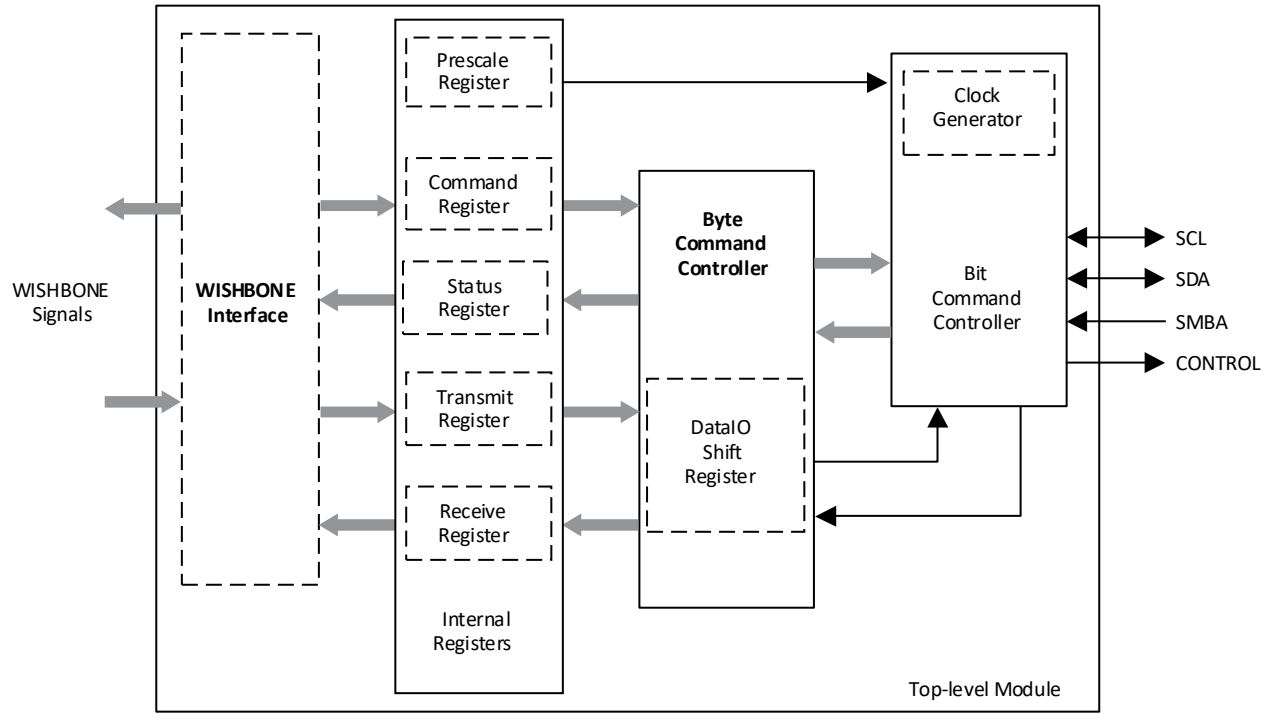

Figure 4.1. Design Modules

4.1. Top-level Module (PMBus_master_top.v)

In addition to connecting all the functional blocks together, this module generates byte-wide data, acknowledgement, and interrupt for the WISHBONE interface. Depending on the parameter ARST_LVL, the reset polarity is determined and distributed to all the modules.

4.2. Internal Registers Module (PMBus_master_registers.v)

A 2-bit by 8-bit register space constitutes the internal register structure of the PMBus core. The space houses the six 8-bit registers listed in Table 4.1. The addresses not used are reserved for future expansion of the core.

Table 4.1. Internal Register List

Name	Address	Width	Access	Description
PRERlo	0x00	8	R/W	Clock prescale register lo-byte
PRERhi	0x01	8	R/W	Clock prescale register hi-byte
CTR	0x02	8	R/W	Control register
TXR	0x03	8	W	Transmit register
RXR	0x03	8	R	Receive register
CR	0x04	8	W	Command register
SR	0x04	8	R	Status register

The **Prescale Register** (address = 0x00 and 0x01) is used to prescale the scl clock line based on the master clock. This design uses an internal clock, named clk_en, to generate the scl clock frequency. The frequency of clk_en is calculated by the equation [wb_clk_i frequency / Prescale Register+1] and this frequency is five times the scl frequency. The contents of the Prescale Register can only be modified when the core is not enabled.

Only two bits of the **Control Register** (address = 0x01) are used for this design. The MSB of this register is the most critical one because it enables or disables the entire PMBus core. The core will not respond to any command unless this bit is set.

The **Transmit Register** and the **Receive Register** share the same address (address = 0x30) depending on the direction of data transfer. The data to be transmitted via PMBus will be stored in the Transmit Register, while the byte received via PMBus is available in the Receive register.

The **Status Register** and the **Command Register** share the same address (address = 0x04). The Status Register allows the monitoring of the PMBus operations, while the Command Register stores the next command for the next PMBus operation. Unlike the rest of the registers, the bits in the Command Register are cleared automatically after each operation. Therefore, this register has to be written for each start, write, read, or stop of the PMBus operation. Table 4.2. provides a detailed description of each bit in the internal registers.

Table 4.2. Description of Internal Register Bits

Internal Register	Bit #	Access	Description
Control Register (0x02)	7	RW	EN, PMBus core enable bit. '1' = the core is enabled; '0' = the core is disabled.
	6	RW	IEN, PMBus core interrupt enable bit. '1' = interrupt is enabled; '0' = interrupt is disabled.
	5:0	RW	Reserved
Transmit Register (0x30)	7:1	W	Next byte to be transmitted via PMBus
	0	W	This bit represents the data's LSB. This bit represents the R/W bit during slave address transfer '1'= reading from slave; '0'= writing to slave
Receive Register (0x30)	7:0	R	Last byte received via PMBus
Status Register (0x04)	7	R	RxACK, Received acknowledge from slave. This flag represents acknowledge from the addressed slave. '1' = No acknowledge received; '0' = Acknowledge received
	6	R	Busy, indicates the PMBus bus busy '1'= START signal is detected; '0'= STOP signal is detected
	5	R	AL, Arbitration lost This bit is set when the core lost arbitration. Arbitration is lost when: A STOP signal is detected, but not requested. The master drives SDA high, but SDA is low.
	4	R	This bit is set when signal SMBA is active.
	3	R	Timeout. '1' = scl and sda line have been high for 50ms.
	2	R	Timeout. '1' = scl line has been low for 25ms.
	1	R	TIP, Transfer in progress. '1' = transferring data; '0' = transfer is complete
	0	R	IF, Interrupt Flag. This bit is set when an interrupt is pending, which will cause a processor interrupt request if the IEN bit is set. The Interrupt Flag is set when: One byte transfer has been completed. Arbitration is lost.
Command Register (0x04)	7	W	STA, generate (repeated) start condition
	6	W	STO, generate stop condition
	5	W	RD, read from slave
	4	W	WR, write to slave
	3	W	ACK, when a receiver, sent ACK (ACK = '0') or NACK (ACK = '1')
	2	W	When set, clears the timeout status.
	1	W	Corresponds to the signal CONTROL
	0	W	IACK, Interrupt acknowledge. When set, clears a pending interrupt.

4.3. Byte Command Controller Module (PMBus_master_byte_ctrl.v)

The microcontroller issues commands and data through the WISHBONE interface in byte format. The information is fed into the Byte Command Controller module and is translated into PMBus sequences required for a byte transfer. This module includes a state machine, as shown in Figure 4.2., to handle normal PMBus transfer sequences. The module then breaks up a single command into multiple clock cycles for the Bit Command Controller to work on bitlevel PMBus operations. This module also contains a shift register which is used for both READ and WRITE cycles. During a READ cycle, the input to the shift register comes from the sda line. After eight scl cycles, the shifted-in data is copied into the Receive Register. During a WRITE cycle, the input to the shift register comes from the WISHBONE data bus. The data in the shift register is shifted out to the sda line during WRITE.

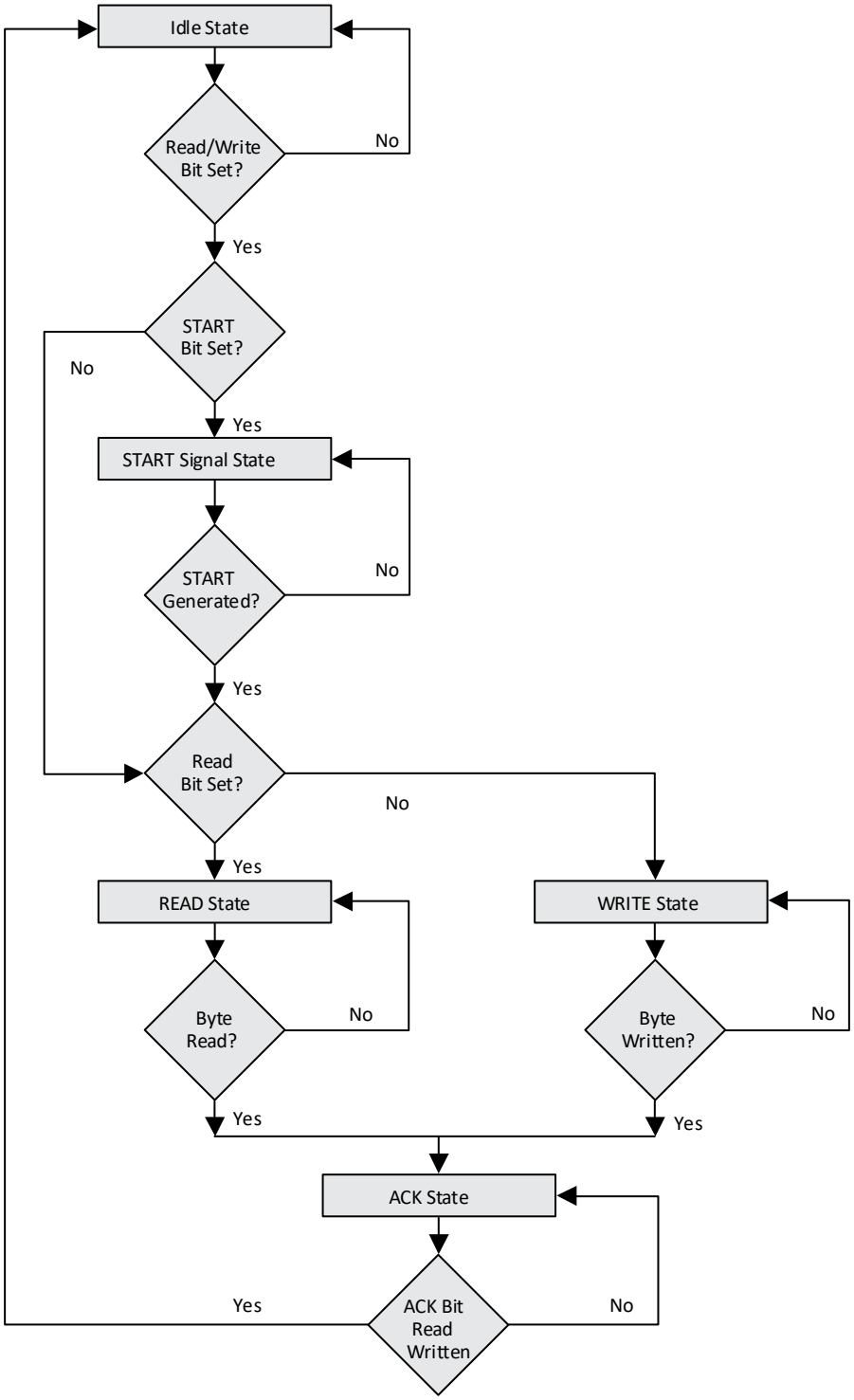

Figure 4.2. PMBus Byte Command State Machine

4.4. Bit Command Controller Module (PMBus_master_bit_ctrl.v)

This module directly controls the PMBus bus, scl and sda lines, by generating the correct sequences for START, STOP, Repeated START, READ, and WRITE commands. Each bit operation is divided into five (5 x scl frequency) clock (clk_en) cycles (idle, A, B, C, and D), except for the START command that has six clock cycles. This ensures that the logical relationship between the scl and sda lines meets the PMBus requirement for these critical commands. The internal clock (clk_en) running at 5 x scl frequency is used for the registers in this module (Figure 4.3., Table 4.3.). The designer can calculate the appropriate frequency of the internal clock clk_en to meet the PMBus timing specification by setting a appropriate value to the Prescale Register.

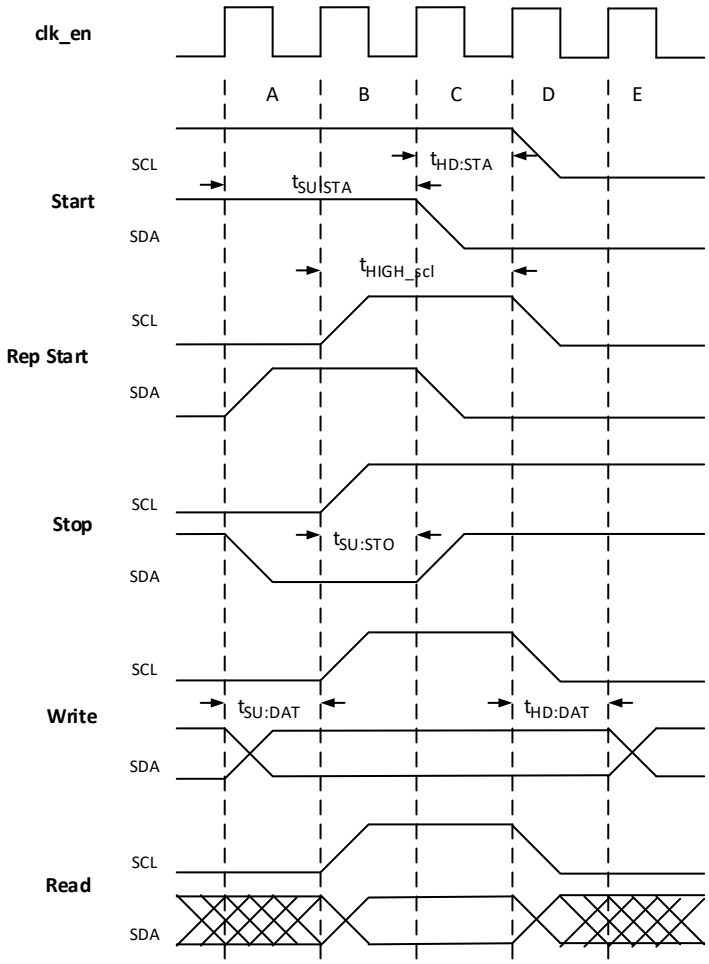

Figure 4.3. PMBus Bit Command Illustration

Table 4.3. PMBus Timing Specifications

Symbol	Parameter	Value
tHD:STA	Hold time after Start condition	1 clk_en cycle
tSU:STA	Start condition setup time	1 clk_en cycle
tSU:DAT	Data setup time	1 clk_en cycle
tHD:DAT	Data hold time	1 clk_en cycle
tHIGH_scl	Scl high period	2 clk_en cycles
tLOW:scl	Scl low period	3 clk_en cycles
tSU:STO	Stop condition setup time	1 clk_en cycle

4.5. Miscellaneous Features

By the nature of open-drain signals, the PMBus provides clock synchronization through a wired-AND connection on the scl line. This clock synchronization capability can be used as a handshake between the slave and master PMBus devices. By holding the scl line low, the slave device tells the master to slow down the data transfer until the slave device is ready. This design detects the scl line to determine if the line is being held.

This design supports multiple masters and thus incorporates the arbitration lost detection. The master that loses the arbitration reports the status in Status Register bit 5. The arbitration is lost when the master detects a STOP condition which is not requested, or when the master drives the sda line high but the sda line is pulled low. The arbitration lost resets the bits in the Command Register to clear the current command for the master to start over again.

This design monitors the scl and sda bus values to determine if either of two timing conditions have been violated:

- The PMBus specification requires that when scl has been low for 25ms, the PMBus master must generate a stop condition within or after the current data byte in the transaction. This design generates a stop condition after the current data byte in the transaction when the master detects this condition and reports the status in Status Register bit 2. An appropriate value can be set for the parameter SCL_LOW_TIMEOUT in the source code to meet the 25 ms requirement according to the clock wb_clk_i frequency.
- The PMBus specification also requires that when scl and sda have been high for 50ms, the PMBus is considered free. This design reports the status in Status Register bit 3 when such condition is detected. An appropriate value can be set the parameter SCL_SDA_HIGH_TIMEOUT in the source code to meet the 50ms requirement according to the clock wb_clk_i frequency.

The signal SMBA is an interrupt line for devices that want to trade their ability to master. When this signal is active, this design reports the status in Status Register bit 4.

The CONTROL signal is an input signal on a power converter. It is used to turn the unit on and off in conjunction with commands received via the serial bus. This signal can be set by the Command Register bit 1.

These features are described in detail in the PMBus specification.

4.6. Operation Sequence of PMBus Controller

To Write Data to the PMBus Slave:

1. Write the appropriate data to the **Prescale Register** based on the frequency of scl through the WISHBONE bus.
2. Write 0x01 to the **Control Register** to enable the PMBus controller through the WISHBONE bus.
3. Write the PMBus slave address+write bit to the **Transmit Register** through the Wishbone bus.
4. Write 0x90 to the **Control Register** through the WISHBONE bus to start the PMBus write operation.
5. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.
6. Write the byte that was sent to the PMBus slave to the **Transmit Register** through the WISHBONE bus.
7. Write 0x10 to the **Control Register** through the WISHBONE bus to set the PMBus write operation.
8. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.
9. When all the bytes are sent, write 0x40 to the **Control Register** through the WISHBONE bus to stop the PMBus write operation.
10. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.

To Read Data from the PMBus Slave:

1. Write the appropriate data to the **Prescale Register** based on the frequency of scl through the WISHBONE bus.
2. Write 0x01 to the **Control Register** to enable the PMBus controller through the WISHBONE bus.
3. Write the PMBus slave address+read bit to the **Transmit Register** through the WISHBONE bus.
4. Write 0x90 to the **Control Register** through the WISHBONE bus to start the PMBus read operation.
5. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.
6. Write 0x20 to the **Control Register** through the WISHBONE bus to read data from the slave.
7. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.
8. Read data from the **Receive Register** through the WISHBONE bus.
9. When the read operation is finished, write 0x40 to the **Control Register** through the WISHBONE bus to stop the PMBus read operation.
10. Read the **Status Register** through the WISHBONE bus until bit 1 of the Status Register is set.

5. HDL Simulation and Verification

The PMBus master with WISHBONE interface design is simulated using a PMBus slave model (PMBus_slave_model.v) and a WISHBONE master model (wb_master_model.v). The slave model emulates the responses of a PMBus slave device by sending an ACK signal when the address matches and when the WRITE operation is completed. The master model contains several tasks to emulate the WISHBONE READ, WRITE, and Compare commands normally issued by the microcontroller. The top-level testbench (tst_bench_top.v) controls the flow of the PMBus operations. The START, WRITE, REPEATED START, READ, consecutive READ, ACK/NACK, STOP, and clock stretching operations are simulated with this test bench.

The following timing diagrams shows the major timing milestones in the simulation.

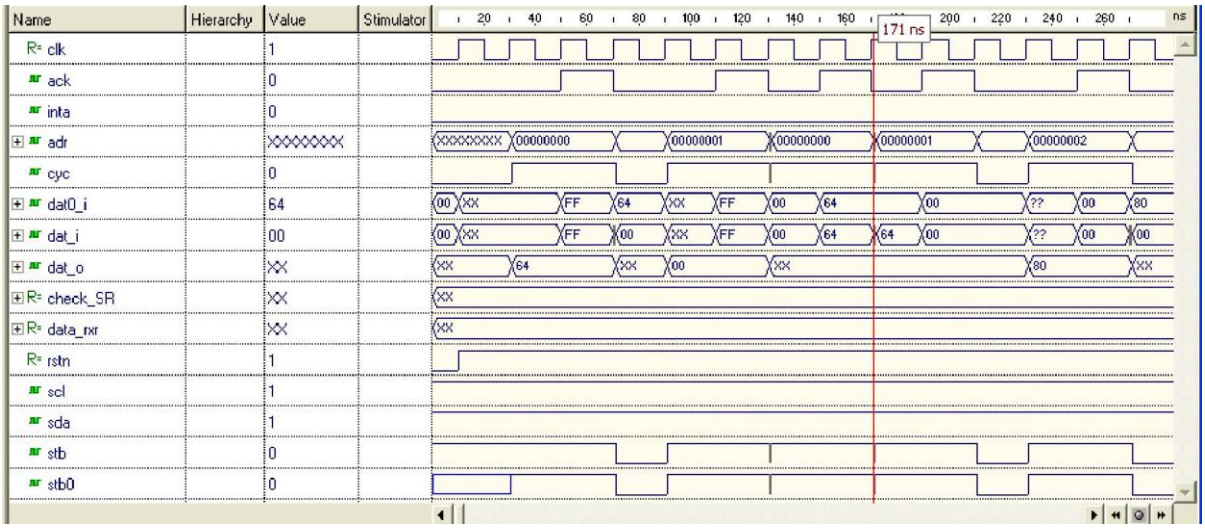

Figure 5.1. Writing Prescale Register with 0x64 and 0x00 at Addresses 0x00 and 0x01 Respectively

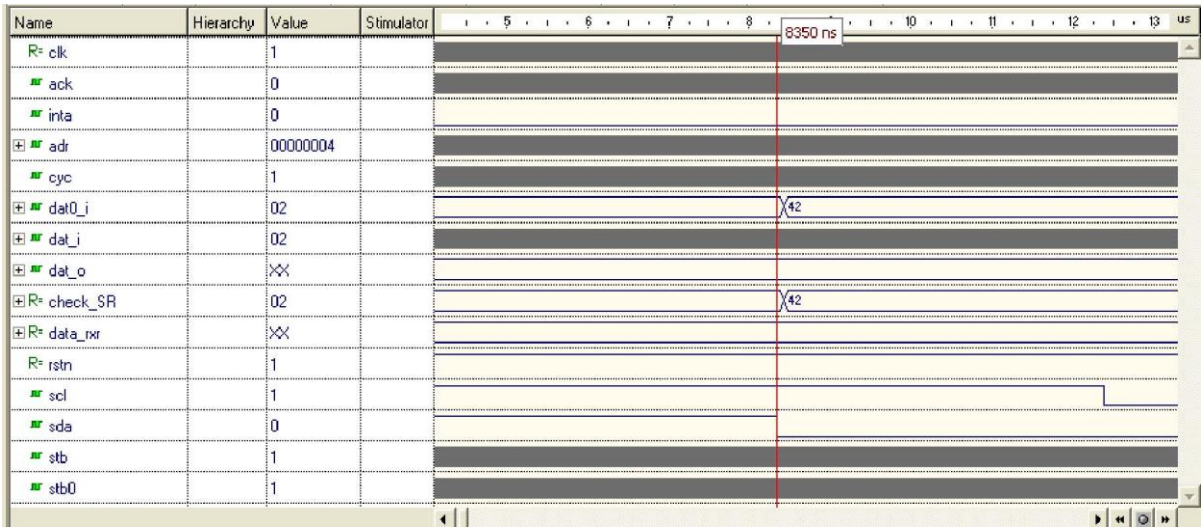

Figure 5.2. Initiate a START, SR[1] (Transfer in Progress) and SR[6] (Busy) Are Set

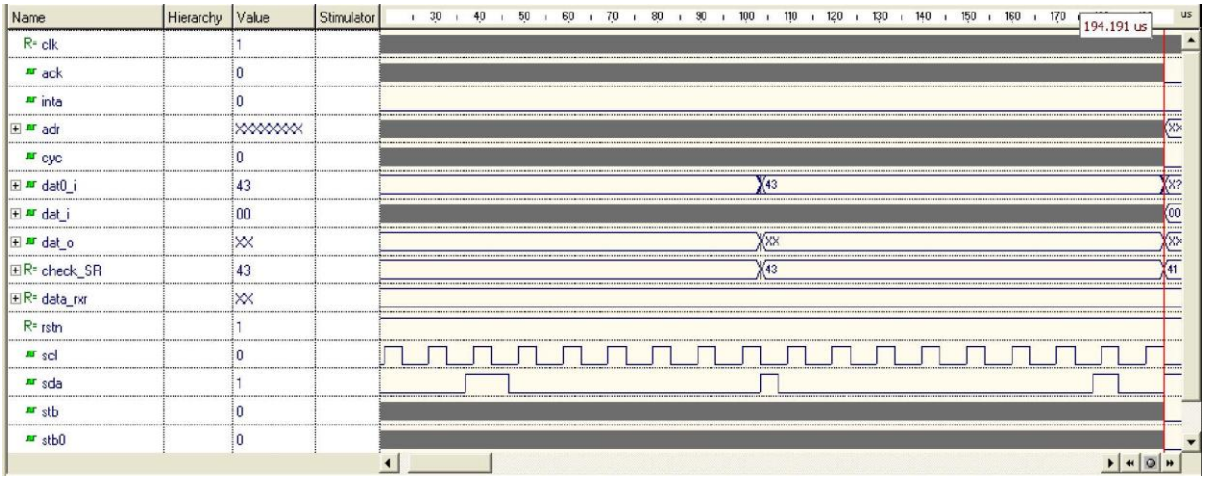

Figure 5.3. Transfer Slave Address + WR, Receive ACK from Slave, Transfer Slave Memory Address 0x01, Receive ACK from Slave, Release SR[1] (Transfer in Progress)

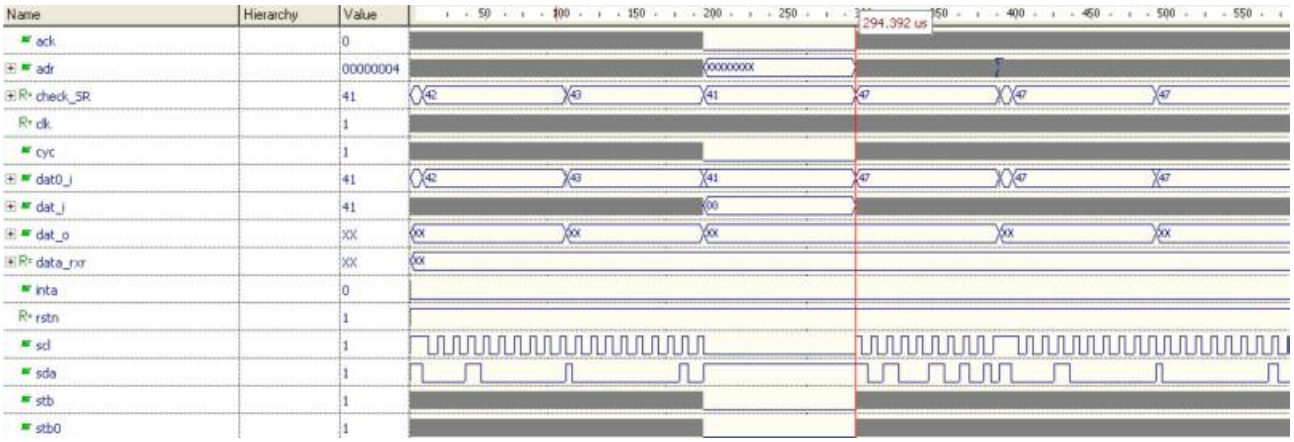

Figure 5.4. Clock Stretching by Slave, scl Line Held Low for 25ms, Master Generate the Stop condition

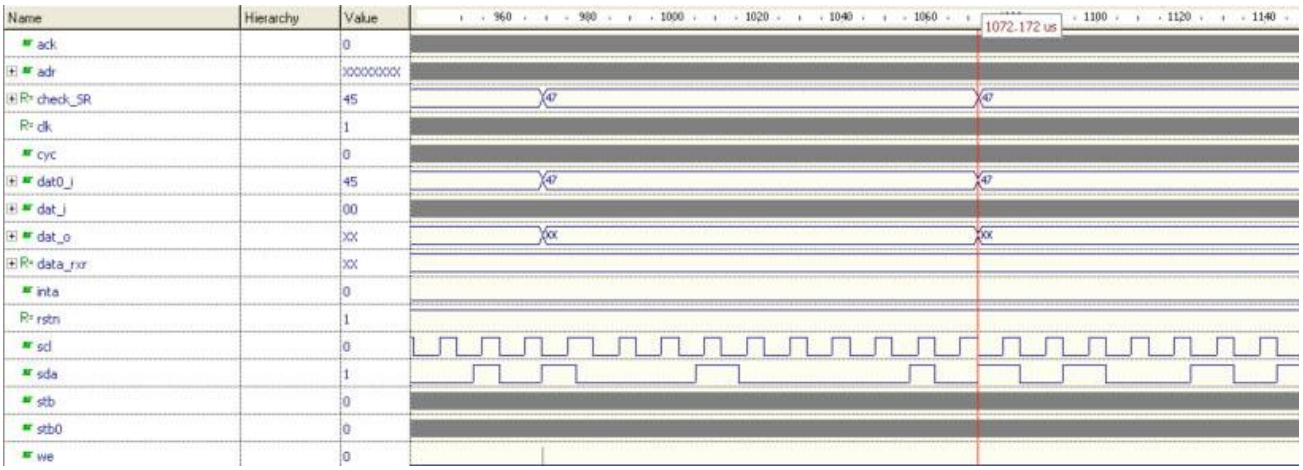

Figure 5.5. Repeated START with Slave Address + RD Command

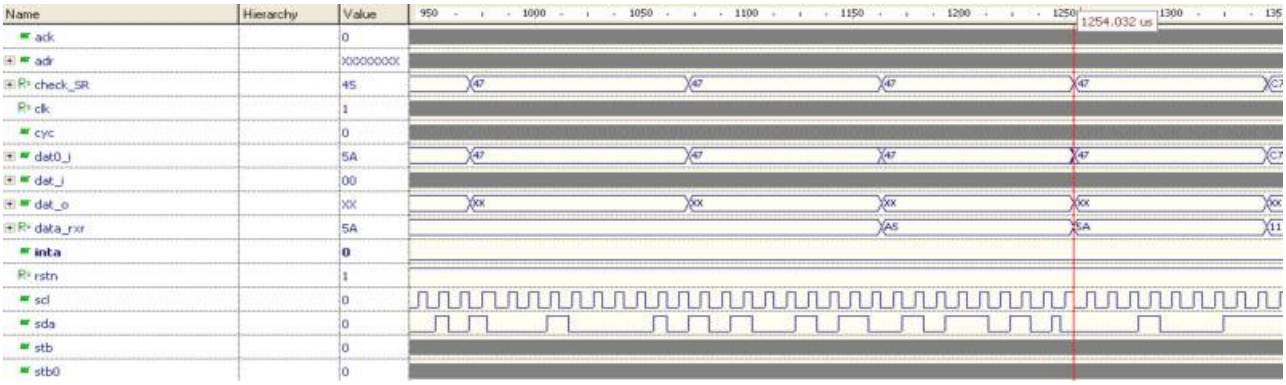

Figure 5.6. Consecutive READ from the Slave, Data Read are 0xA5, 0x5A, and 0x11

6. Implementation

Table 6.1. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization	fMAX(MHz)	I/Os	Architecture Resources
MachXO2™ [1]	Verilog	-4	299 LUTs	>50	31	N/A
	VHDL	-4	290 LUTs	>50	31	N/A
MachXO™ [2]	Verilog	-4	292 LUTs	>50	31	N/A
	VHDL	-4	288 LUTs	>50	31	N/A
Platform Manager[3]	Verilog	-3	291 LUTs	>50	31	N/A
	VHDL	-3	288 LUTs	>50	31	N/A

Notes:

1. Performance and utilization characteristics are generated using LCMXO2-2000HC-4TG100CES, with Lattice Diamond™ 1.1 and isp- LEVER® 8.1 SP1 software. When using this design in a different device, density, speed, or grade, performance and utilization may vary
2. Performance and utilization characteristics are generated using LCMXO1200C-3T100C, with Lattice Diamond 1.1 and ispLEVER 8.1 SP1 software When using this design in a different device, density, speed, or grade, performance and utilization may vary.
3. Performance and utilization characteristics are generated using LPTM10-12107-3FTG208CES,with ispLEVER 8.1 SP1 software. When using this design in a different device, density, speed or grade, performance and utilization may vary.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.2, December 2019

Section	Change Summary
All	• Changed document number from RD1100 to FPGA-RD-02097. • Updated document template.
Disclaimers	Added this section.

Revision 1.1, December 2010

Section	Change Summary
Implementation	Added support for the Platform Manager device family.

Revision 1.0, November 2010

Section	Change Summary
All	Initial release.

www.latticesemi.com