I2C Slave Peripheral Using the Embedded Function Block

Reference Design

FPGA-RD-02073-1.5

July 2021

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, with all faults and associated risk the responsibility entirely of the 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

Microprocessors often have a limited number of general purpose I/O (GPIO) ports that reduce pin count and shrink package size, but limit the number of I/Os to which a microprocessor can be connected. To allow more I/Os to be connected to microprocessors, I/O expanders or port expanders are used to provide I/O expansion capability. Most generic GPIO expanders use low pin count protocols, such as I^2C or SPI, as the interface to the master. They allow designers to save the GPIO ports on the microprocessor for other critical tasks.

The I^2C GPIO Memory Interface reference design provides a programmable solution for serial expansion of GPIOs. It uses an Inter IC Communication (I^2C) interface between the microprocessor and the GPIOs. The design provides additional control and monitoring capabilities for a microprocessor when it does not have sufficient GPIOs to do the job.

Apart from the GPIO expander, this design also provides a memory interface to the microprocessor. This memory is accessible via the I^2C interface. The I^2C memory command interface is similar to those commonly found in discrete I^2C memory devices.

This reference design is intended to provide a familiar and intuitive interface extension to the MachXO2™ and MachXO3L Embedded Function Block (EFB). The EFB I^2C module supports the major features listed in the I^2C specification. Users can take advantage of the MachXO2 and MachXO3L hardened I2C port to access userdefined internal registers or provide a memory extension. The user is spared from learning operational details of the I^2C protocol, the WISHBONE bus or the EFB block.

2. Interface

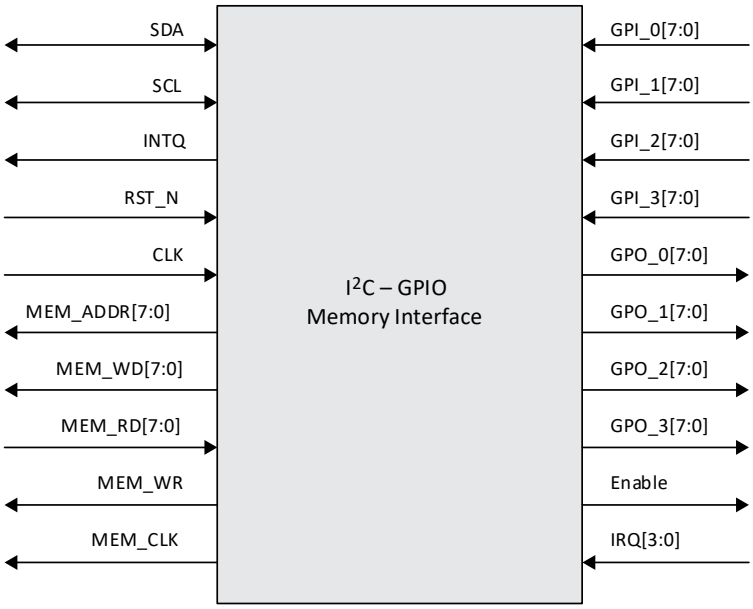

Figure 2.1. I^2C to GPIO Memory Interface

3. Functional Description

This design provides eight bytes of I/O port and memory interface, which is controlled through the I²C slave interface. There are four single-byte input ports and four single-byte output ports. The interrupt generation block generates an interrupt signal to the master when at least one of the interrupt pins is active high. This design interfaces with an embedded memory block. The memory read and write operations are controller by the I²C commands. The operation of the design will be activated only when an enable command is received by the I²C slave.

The state machine in the design is designed according to Using User Flash Memory and Hardened Control Functions in MachXO2 Devices (FPGA-TN-02162). and Using Hardened Control Functions in MachXO3 Devices Reference Guide (FPGA-TN-02064-1.8). It responds to the commands issued by an I²C master. In normal situations, the I²C slave acknowledges the successful receipt of an address or data by issuing an Ack command. In the case of an erroneous transfer, the state machine will respond with a Nack command. The state machine is capable of handling error conditions such as invalid commands, unexpected or wrong operands, abrupt stop, etc. The detailed state machine flow for read and write commands is shown in Figure 4.1. , Figure 4.2. and Figure 4.3. Figure 4.1. shows the common steps to be followed for a read and write operation. Figure 4.2. show the steps to be followed for a read operation and Figure 4.3. shows the steps to be followed for a write operation.

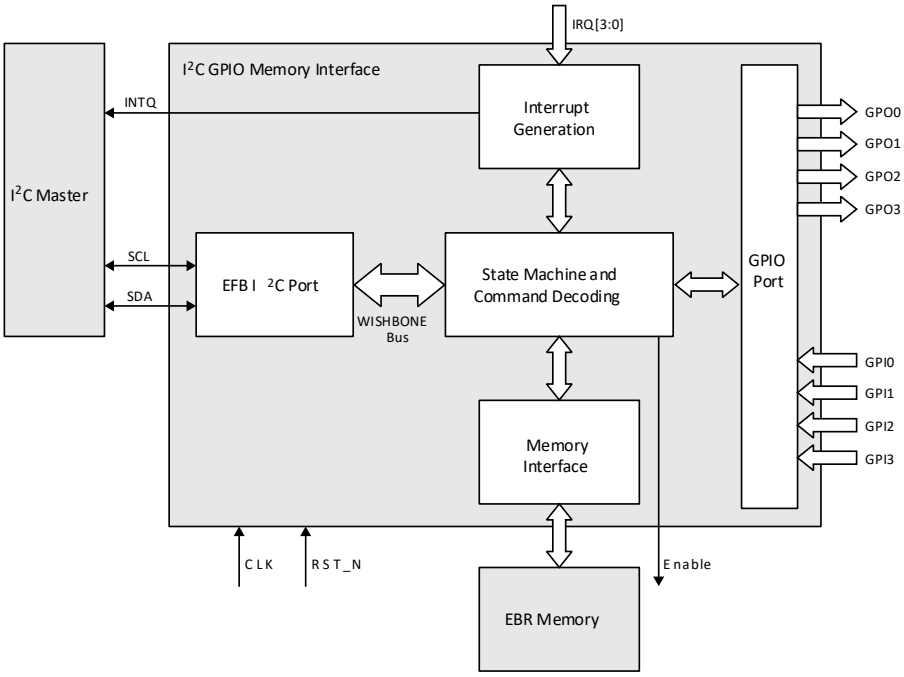

Figure 3.1. Functional Block Diagram

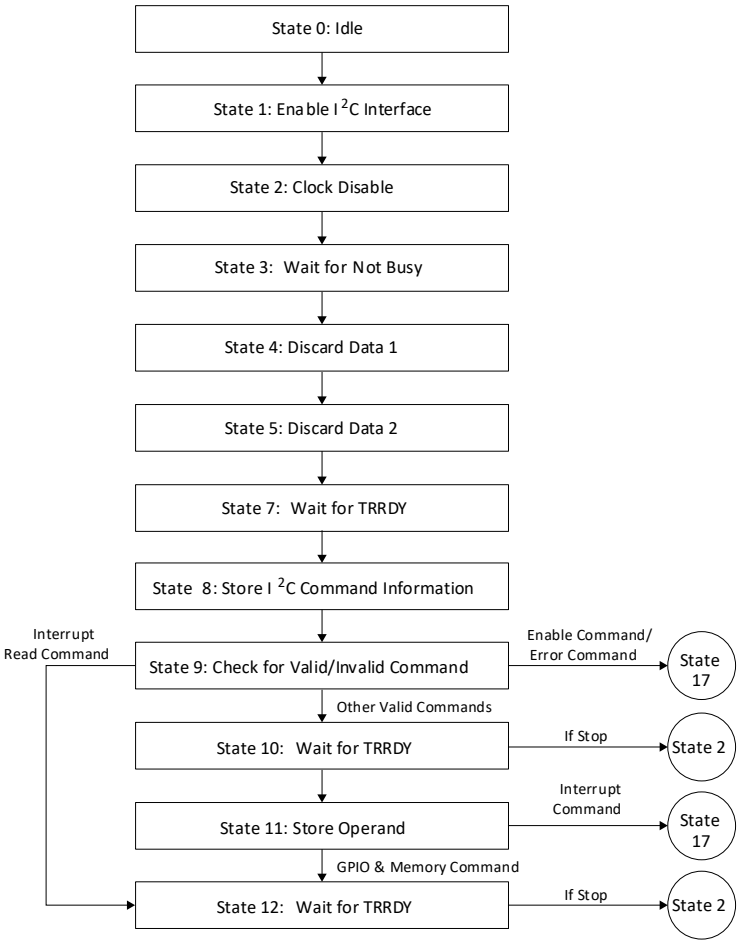

Figure 3.2. State Machine Flow for Common Operations

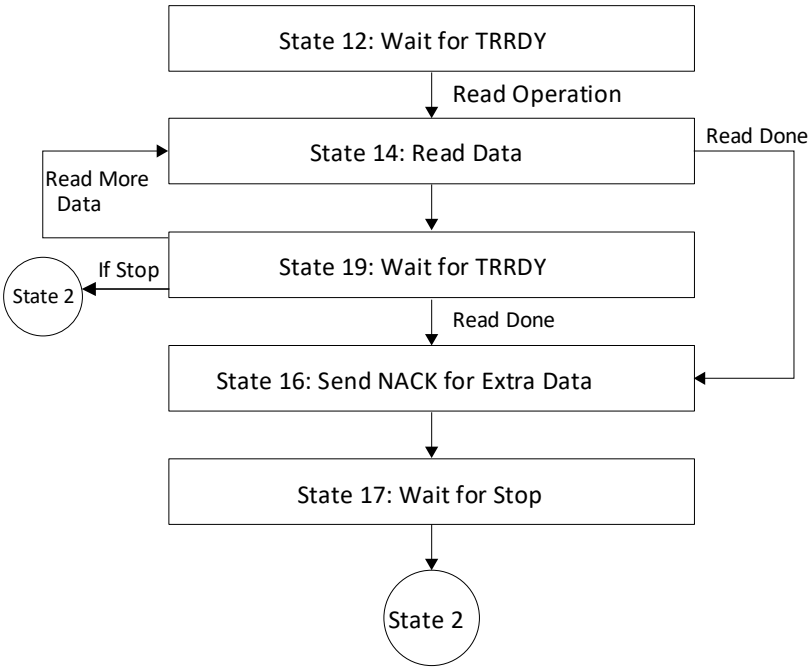

Figure 3.3. State Machine Flow for Read Operations

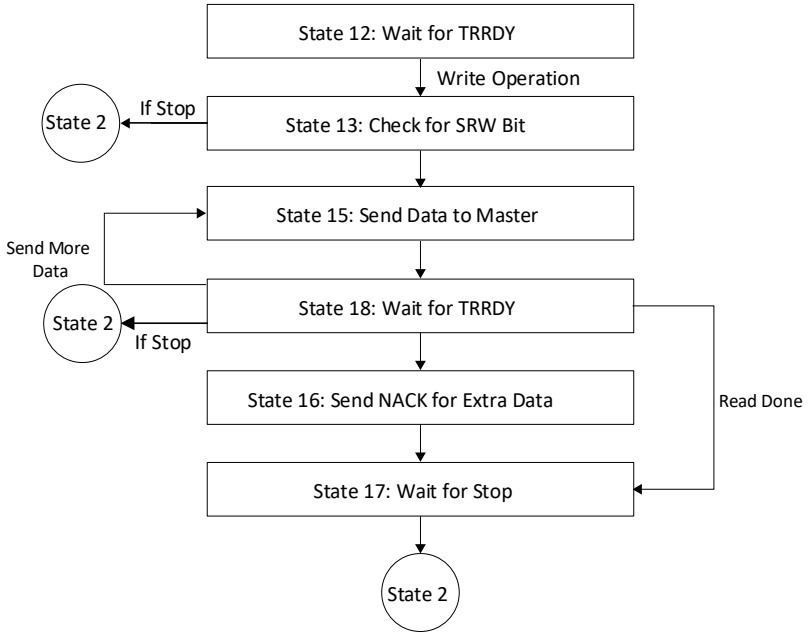

Figure 3.4. State Machine Flow for Write Operations

Table 3.1. I²C GPIO Expander I/O Interface Descriptions

Signal	I/O	Description
I²C Master Interface		
SCL	Input	Serial clock
SDA	Input	Serial Data in
INTQ	Output	External interrupt, asserted low. Asserted low when any IRQ status register bit is set (open drain output).
GPIO Interface		
GPO_0[7:0]	Output	General purpose output byte port
GPO_1[7:0]	Output	General purpose output byte port
GPO_2[7:0]	Output	General purpose output byte port
GPO_3[7:0]	Output	General purpose output byte port
GPI_0[7:0]	Input	General purpose input byte port
GPI_1[7:0]	Input	General purpose input byte port
GPI_2[7:0]	Input	General purpose input byte port
GPI_0[7:0]	Input	General purpose input byte port
IRQ[3:0]	Input	Input interrupt
General Signals		
CLK	Input	Master clock
RST_N	Input	Active low reset
Enable	Output	General purpose enable signal
Memory Interface		
MEM_CLK	Output	Clock port for memory
MEM_WR	Output	Memory write signal
MEM_ADDR[7:0]	Output	Memory address
MEM_WD[7:0]	Output	Memory write data
MEM_RD[7:0]	Input	Memory read data

4. I²C Slave Interface

The I²C GPIO Memory Interface design has a 7-bit I²C slave address format. This address is defined by the user while instantiating an EFB block from the IPexpress™ tool. In this design the I²C slave address is set as '0001001B'. The first seven bits of the first byte sent by the I²C master after a start condition are the I²C slave address. The eighth bit determines the direction of the message. A '0' in the least significant position of the first byte indicates a write operation from the I²C master to the I²C slave. A '1' in this position indicates a read operation on the I²C bus.

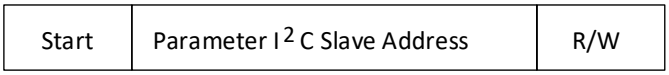

Figure 4.1. I²C Slave Address

A typical I²C write and read command sequence is shown in the figures below.

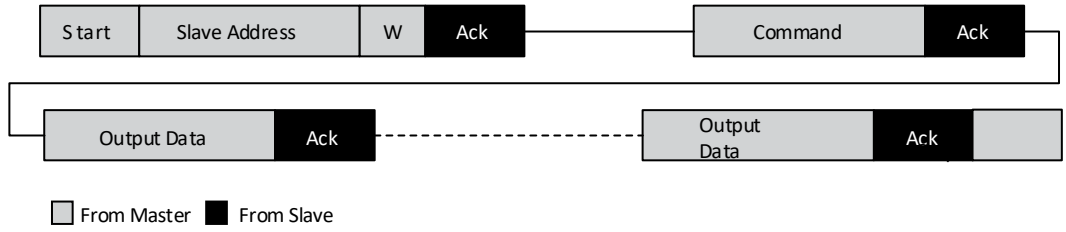

Figure 4.2. I²C Write Operation

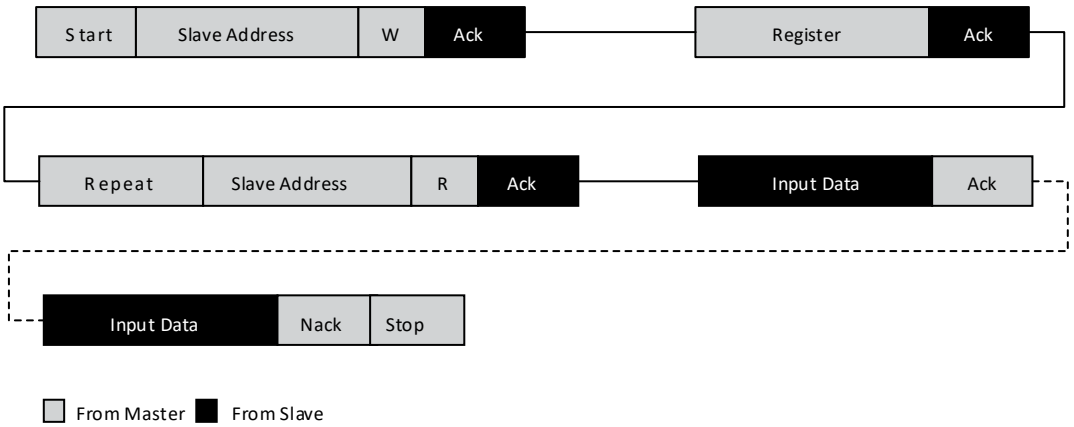

Figure 4.3. I²C Read Operation

5. HDL Parameter Descriptions

This design uses a number of parameters to control various aspects of the design. This allows the user to specify the interfaces to meet custom requirements without modifying the underlying Verilog RTL code. Table 5.1. provides descriptions of the parameters used in the design.

Table 5.1. Parameter Descriptions

Operation	Description	Value
GPI_PORT_NUM	Specifies the number of general purpose input ports	1|7
GPI_DATA_WIDTH	Specifies the width of general purpose input ports	1|8
GPO_PORT_NUM	Specifies the number of general purpose output ports	1|7
GPO_DATA_WIDTH	Specifies the width of general purpose output ports	1|8
MEM_ADDR_WIDTH	Specifies the width of memory address	1|8
IRQ_NUM	Specifies the number of input interrupt pin	1|8
MAX_MEM_BURST_NUM	Specifies the burst width for memory operation	1|255
INTQ_OPENDRAIN	Specify whether INTQ output will be open drain or not	ON/OFF

6. Commands

Table 6.1. Commands

Operation	Command	Operands	Data	Number of Bytes in Command
Enable	0x06	Nil	Nil	1 byte
Disable	0x04	Nil	Nil	
IRQ Enable Write	0x66	Nil	1 byte bitwise enable	2 bytes
IRQ Status	0x65	Nil	1 byte bitwise interrupt source	
IRQ Clear	0x61	Nil	1 byte bitwise interrupt clear	
IRQ Enable Read	0x6A	Nil	1 byte bitwise enable read	
Write GPO	0x01	Port # (0 to 3)	1 byte write data	3 bytes
Read GPI	0x05	Port # (0 to 3)	1 byte read data	
Write Memory	0x02	1-byte address	8 bytes write data	10 bytes
Read Memory	0x0B	1-byte address	8 bytes read data	

6.1. Enable/Disable Command

The Enable command sets the Enable pin of the design. This signal is used to activate the custom logic or memory interface. Figure 6.1. shows the enable command operation. The Disable command has the same structure except for the command value.

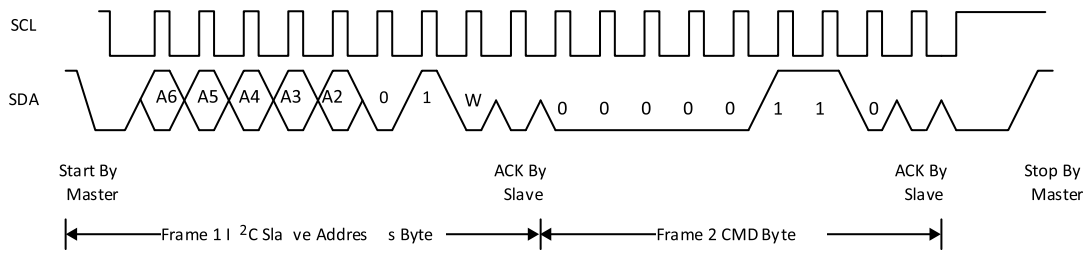

Figure 6.1. Enable Command Operation

6.2. Interrupt Commands

This design supports four input interrupts and based upon their status and enable bit it will generate an output interrupt, INTQ. These inputs can be enabled/disabled by the I²C master. The interrupt output signal is activated (active low) when one of the interrupt signals is set and the corresponding interrupt enable bit is also set. The I²C master can read the status of these interrupts and its enable settings by issuing a read command.

The design maintains two registers to hold the values of the interrupt enable and interrupt status. The interrupt enable register will hold the values of enables set by the I²C master. The I²C master can perform a read operation on this register. The interrupt status register will hold the status of the interrupt input when it has changed from '0' to '1'. The status register will be cleared by the I²C master only by sending IRQ clear command. Figure 3.2. shows the structure of the interrupt enable and status register.

Table 6.2. Interrupt Enable and Status Register

Register	Width	Access	Reset Value	Bit Definition
Interrupt Enable	8	R/W	0x00	[7-4] – Reserved for future use 3 – Enable IRQ[3] signal 2 – Enable IRQ[2] signal 1 – Enable IRQ[1] signal 0 – Enable IRQ[0] signal
Interrupt Status	8	R/W	0x00	[7-4] – Reserved for future use 3 – Set when IRQ[3] goes from 0 to 1 2 – Set when IRQ[2] goes from 0 to 1 1 – Set when IRQ[1] goes from 0 to 1 0 – Set when IRQ[0] goes from 0 to 1 [3-0] will be reset when I²C master sends IRQ Clear. The corresponding bit will be cleared in the status register.

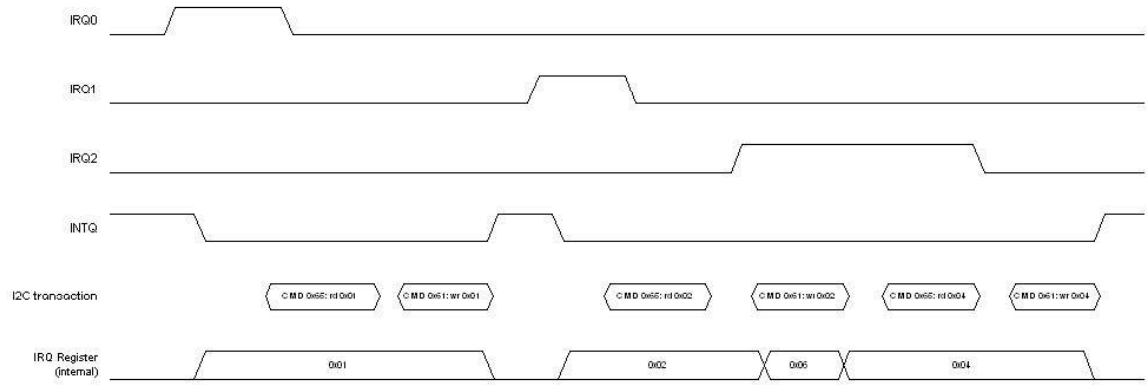

Figure 6.2. Interrupt: Set, Status, Clear

6.3. GPIO Commands

Four single-byte input and output ports can be controlled by the I²C master. All the GPIO commands are three-byte commands. The first byte contains the I²C address and the second byte contains the GPIO port address. Depending upon the read/write operation, the third byte is written or read by the I²C master. GPIO commands are shown in Figure 6.3.

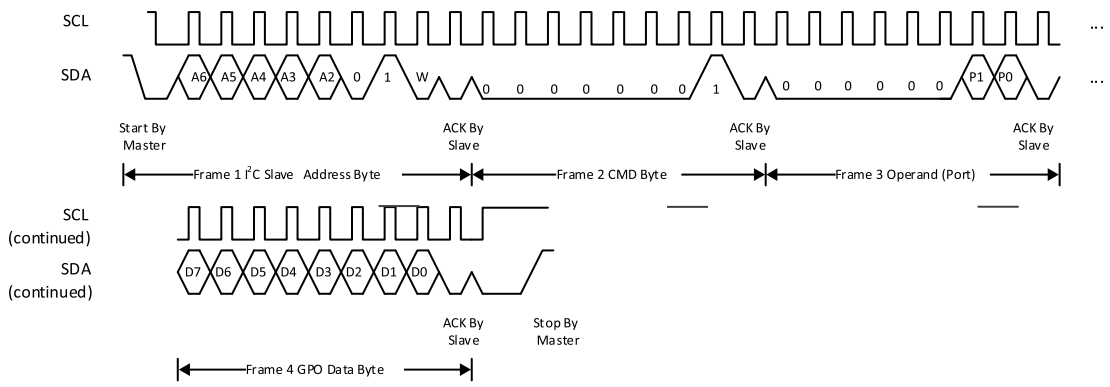

Figure 6.3. Write GPO Transaction

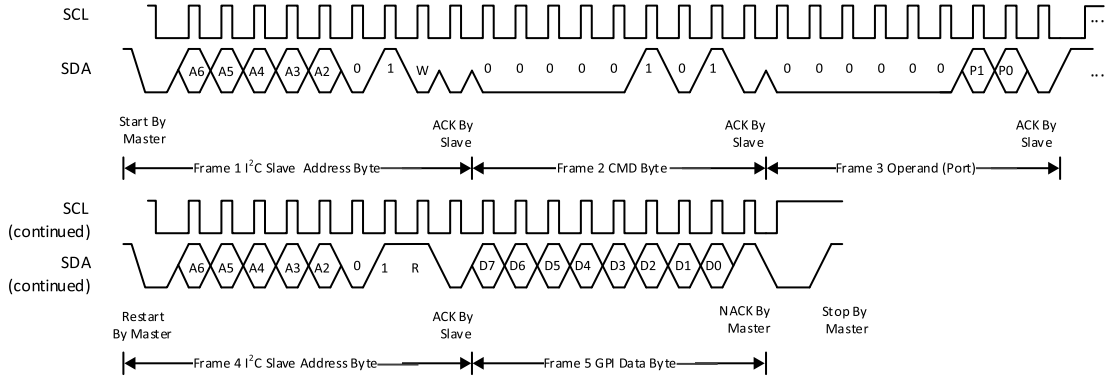

Figure 6.4. Read GPI Transaction

6.4. Memory Commands

An EBR-based memory is interfaced to this design. All memory operations are controlled by the I^{2}C master. The intermediate commands, such as address or write/read, will be generated by the state machine inside the design. A burst of eight bytes are read/written into memory every time. This make the size of the I^{2}C transfer 10 bytes. The enable command will activate the memory block. The memory commands will be similar to the GPIO commands except that they are 10-byte commands. The timing diagram for the memory interface is shown in Figure 6.5.

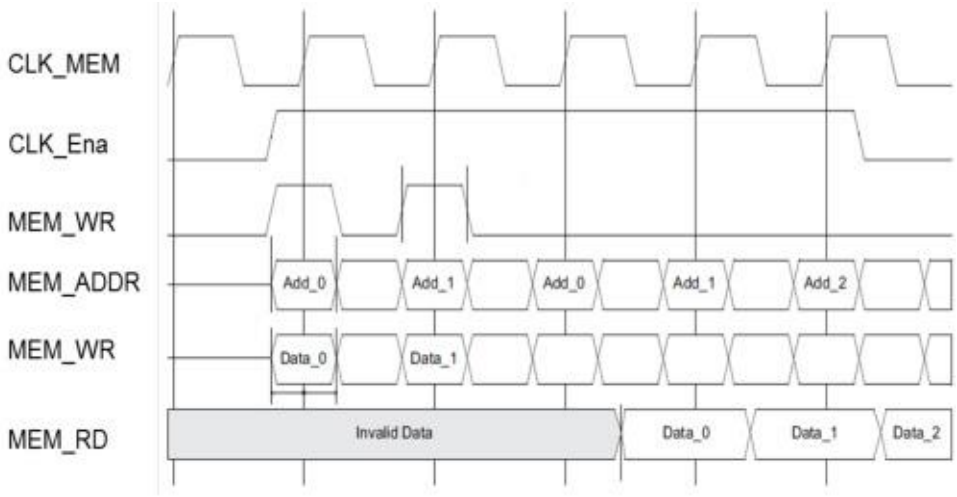

Figure 6.5. Memory Interface Timing Diagram

6.5. Erroneous Commands

This design is capable of detecting error transfers such as incorrect commands, too many bytes, too few bytes etc. The design can respond to different error scenarios. In the case of an incorrect command sent to the design, the design will issue a NACK. If there are more than the required number of bytes present in the transfer, the design will consider the relevant bytes and ignore the extra bytes sent by the master. When there are fewer bytes provided by the master, the slave design will ignore the command and discard the data.

Please note that this design does not support stand-alone read operation.

7. Test Bench Description

The I²C GPIO Memory Interface design is simulated with the aide of I²C Master Bus Functional modules. The test bench for this design consists of the following functional blocks, as shown in Figure 7.2.

- I²C master bus functional model
- EBR-based memory module
- Design under test (I²C slave)
- Back-end interface (clock and reset generation)

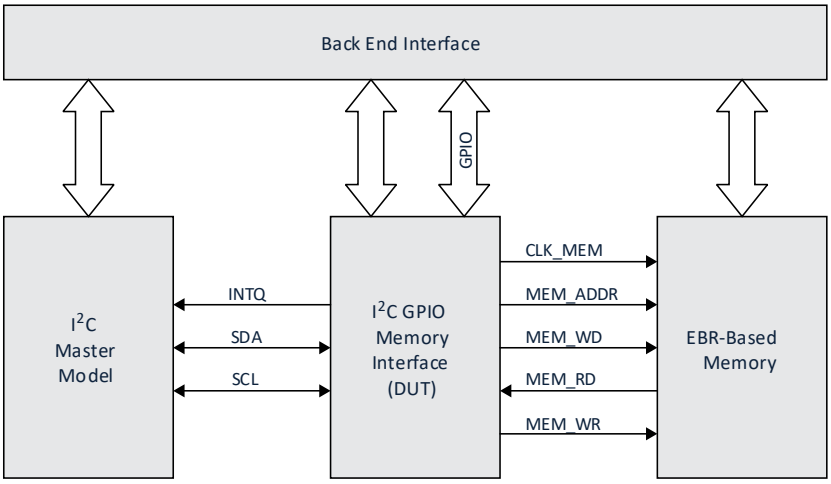

Figure 7.1. Test Bench Architecture

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

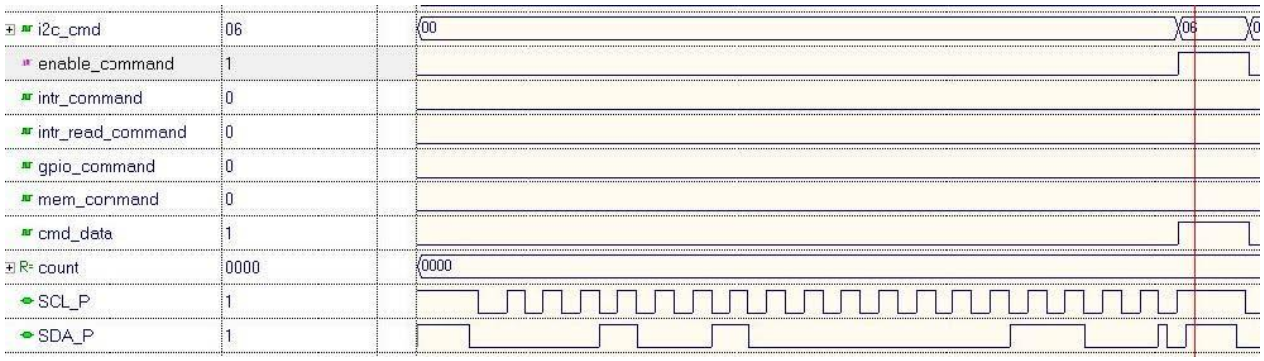

Figure 7.2. Enable Command

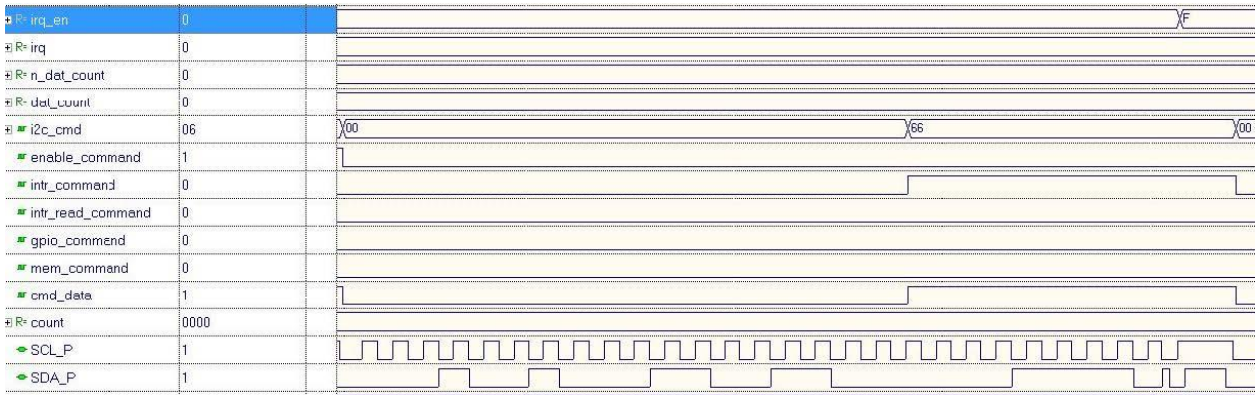

Figure 7.3. Set Interrupt Command

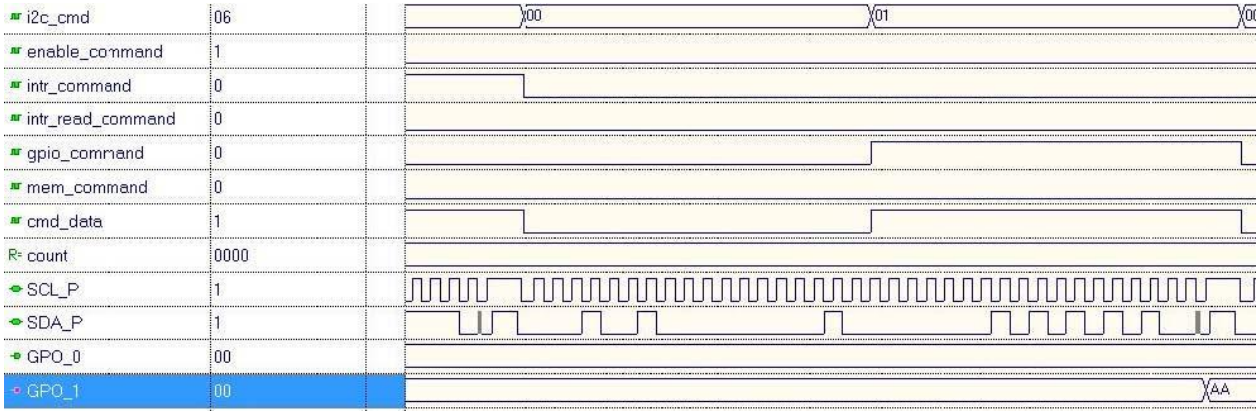

Figure 7.4. Writing GPO Command

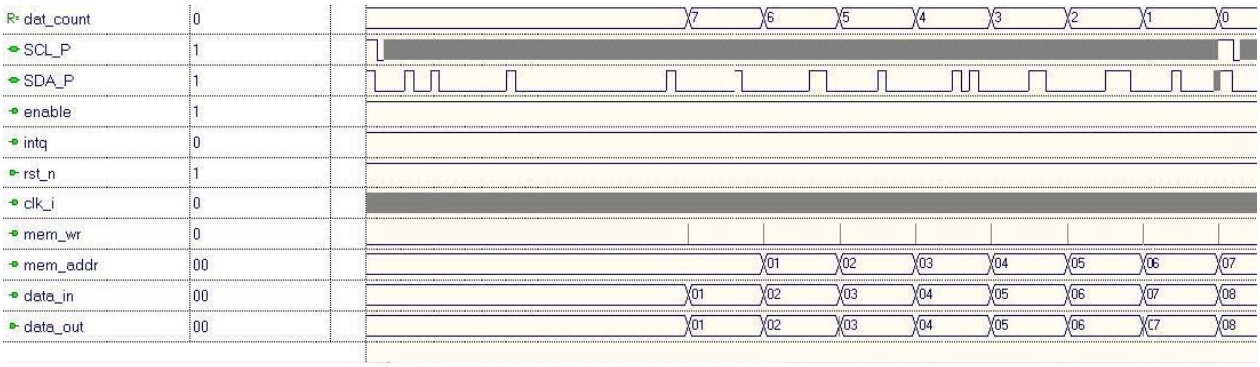

Figure 7.5. Writing into Memory Command

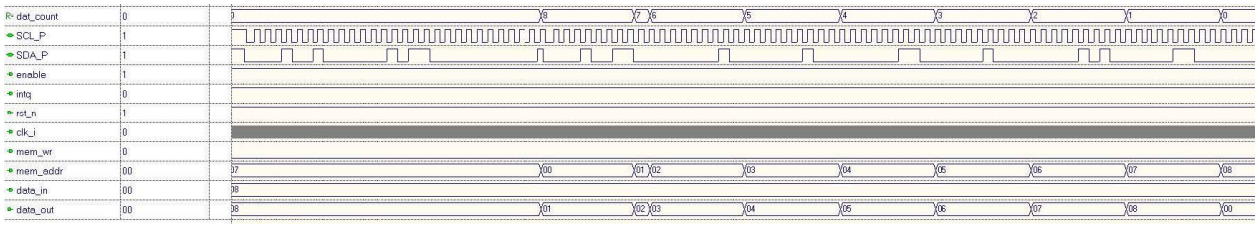

Figure 7.6. Reading from Memory Command

8. Implementation

Table 8.1. Performance and Resource Utilization

Family	Language	Speed Grade	Utilization	fmax (MHz)	I/Os	Architecture Resources
MachXO2[1]	Verilog-LSE	–3	192 LUTs	>40	100	EFB
	Verilog-Syn	–3	179 LUTs	>40	100	EFB
	VHDL-LSE	–3	217 LUTs	>40	100	EFB
	VHDL-Syn	–3	199 LUTs	>40	100	EFB
MachXO3L[2]	Verilog-LSE	–6	192 LUTs	>40	100	EFB
	Verilog-Syn	–6	179 LUTs	>40	100	EFB
	VHDL-LSE	–6	217 LUTs	>40	100	EFB
	VHDL-Syn	–6	199 LUTs	>40	100	EFB

Notes:

1. Performance and utilization characteristics are generated using LCMOX02-1200ZE-3MG132C with Lattice Diamond® 3.3 design software with LSE (Lattice Synthesis Engine) and Synplify Pro®.
2. Performance and utilization characteristics are generated using LCMXO3L-4300C-6BG256C with Lattice Diamond 3.3 design software with LSE and Synplify Pro.

References

- I^2C Specification from Freescale Semiconductor, Inc.
- Using User Flash Memory and Hardened Control Functions in MachXO2 Devices (FPGA-TN-02162)
- Using Hardened Control Functions in MachXO3 Devices Reference Guide (FPGA-TN-02064-1.8)

Technical Support Assistance

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

Revision History

Revision 1.5, July 2021

Section	Change Summary
Functional Description	Updated Figure 3.2, Figure 3.3, and Figure 3.4.

Revision 1.4, November 2019

Section	Change Summary
All	• Changed document number from RD1124 to FPGA-RD-02073. • Updated document template.
Disclaimers	Added this section.

Revision 1.3, October 2014

Section	Change Summary
Introduction	Updated this section. Added MachXO3L support.
Functional Description	Updated this. Revised and added technical note references.
Enable/Disable Command	Updated this section. • Updated Figure 6.1, Enable Command Operation.
GPIO Command	Updated this section. • Updated Figure 6.3, Write GPO Transaction. • Updated Figure 6.4, Read GPI Transaction.
Implementation	Updated Table 8, Performance and Resource Utilization. • Modified Utilization data for Verilog implementation and VHDL implementation. • Provided LSE and Synplify Pro data for MachXO2. • Changed Lattice Diamond software version in footnotes 1 and 2.
References	Updated References section. Modified and added technical note references.

Revision 1.2, February 2014

Section	Change Summary
I²C Slave Interface	Updated this. Changed I²C slave address
Implementation	Updated Table 8.1, Performance and Resource Utilization. • Modified Utilization data for Verilog and VHDL implementation. • Changed device and Lattice Diamond software version in footnote.
All	Added support for MachXO3L device family.

Revision 1.1, September 2013

Section	Change Summary
All	Updated for VHDL implementation.
HDL Parameter Descriptions	Added this section.
Functional Description	Updated Figure 3.2 and Verilog and VHDL code to remove clock stretching support per PCN#10A-13.
Technical Support Assistance	Updated Technical Support Assistance information.

Revision 1.0, April 2012

Section	Change Summary
All	Initial release.

www.latticesemi.com