I^{2}C Controller for Serial EPROMs

January 2015 Reference Design RD1006

Introduction

The I^{2}C bus provides a simple two-wire means of communication. This protocol supports multi-masters and provides a low-speed connection between intelligent control devices, such as microprocessors, and general-purpose circuits, such as memories.

This reference design documents an I^{2}C controller designed to interface with serial EEPROM devices. The design can be used with a microprocessor to read the configuration data from a serial EEPROM that supports an I^{2}C protocol. It is intended to be a simple I^{2}C master using 7-bit addresses and providing random reads cycles only. Typically, serial EEPROMs are programmed at the time of board assembly. They store configuration information which is read by a microprocessor during power-up.

This design is implemented in Verilog and VHDL. Lattice design tools are used for synthesis, place and route and simulation. The design can be targeted to multiple Lattice device families. Its small size makes it portable across different FPGA/CPLD architectures.

This design assumes the user has experience with I^{2}C controllers. Information available in the documents listed below is not repeated in this document.

References

- National Semiconductor NM24C16 16,384-Bit Serial EEPROM
- Philips I^{2}C Specification
- Lattice Semiconductor Data Sheets

Theory of Operation

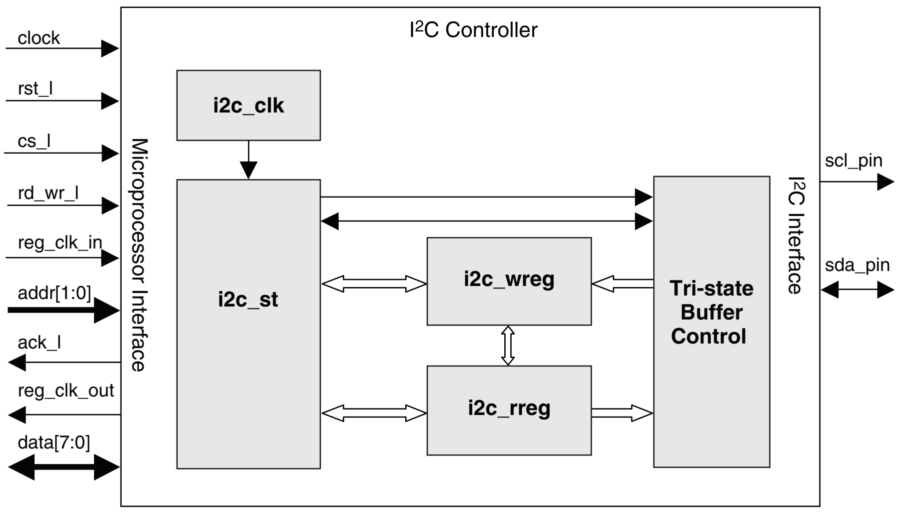
Overview

This I^{2}C controller provides an interface between standard microprocessors and I^{2}C serial EEPROM devices. It acts as an I^{2}C master to support random read cycles from the serial EEPROM. The design consists of the following modules:

• I2c	Top-level module
• I2c_clk	Clock generation module
• I2c_rreg	Read register module
• I2c_st	State machine module
• I2c_wreg	Write register module

Figure 1 provides a top-level block diagram for the I^{2}C controller reference design.

Figure 1. I²C Controller Top-Level Block Diagram



Top-Level Signal Descriptions

Table 1 provides descriptions of the input/output signals of the I²C controller. Signals ending with "_L" indicate an active low signal. This convention is used throughout the design. The chip select signal is assumed synchronous to the clock. Address, data and the read/write signals are assumed valid at the assertion of the chip select.

Table 1. I²C Signal Descriptions

Interface	Signal	Type	Description
Microprocessor Interface	CS_L	Input	I²C chip select from microprocessor
	RD_WR_L	Input	Write pulse from microprocessor
	ADDR[1:0]	Input	Address bus from microprocessor
	DATA[7:0]	I/O	Microprocessor data bus
	CLK	Input	Input clock from microprocessor, 50MHz
	RST_L	Input	Asynchronous reset
	REG_CLK_IN	Input	Clock used to latch word address
	ACK_L	Output	Microprocessor cycle acknowledge
	REG_CLK_OUT	Output	Generated clock to latch word address
I²C Interface	SCL_PIN	Output	I²C clock pin
	SDA_PIN	I/O	I²C data pin

Register Descriptions

Table 2 lists the I²C controller registers.

Table 2. I²C Registers

Register	Address	Type
Word Address	00	Read/Write
Data	01	Read
Status	10	Read

Design Module Description

I2C_CLK Module

The I2C_CLK module provides a one-microprocessor clock tick wide pulse every 5µsec. This is used to control the state machine. The 8-bit counter in this module can be adjusted to meet the needs of any design. If a faster microprocessor clock is used, a bigger counter will be needed. If it is desired to run the I^{2}C bus at a faster speed, the counter can be smaller. This design assumes a microprocessor clock frequency of 50MHz.

I2C_RREG Module

The I2C_RREG module provides a multiplexor for reading data back to the CPU. The word address, the data read from the I^{2}C device and status information can be read back. The status register bits are defined in Table 3.

Table 3. Status Register Bits

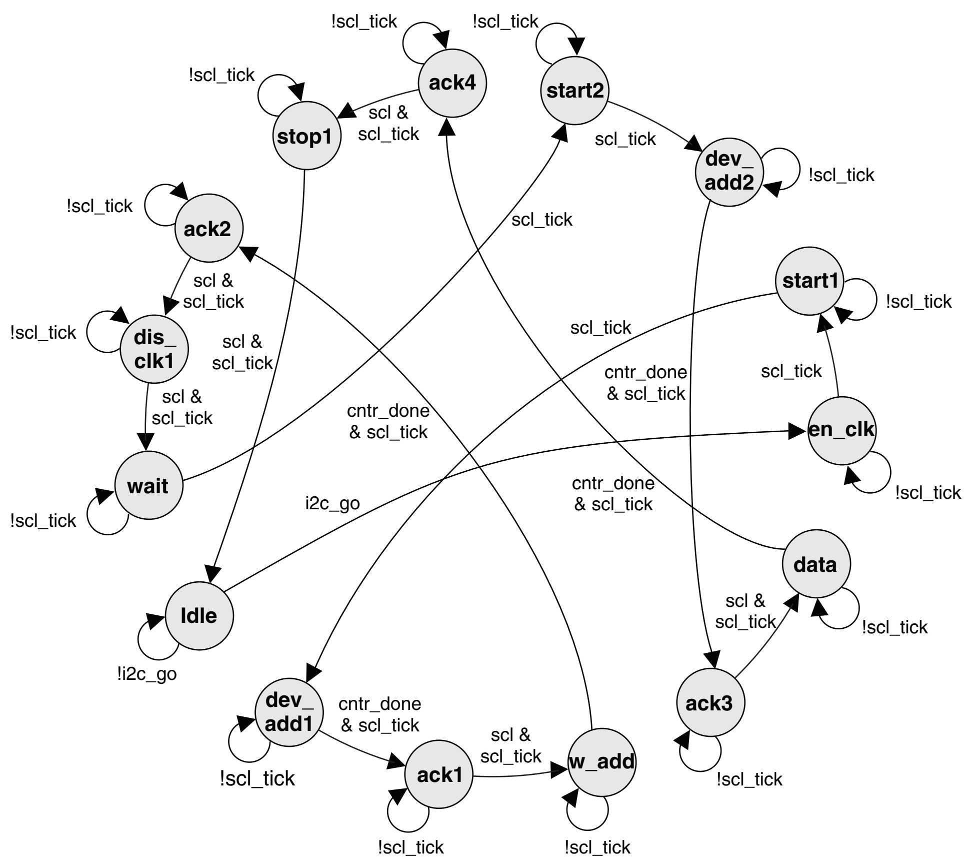
Bit	Signal	Description
7	ready	The I^{2}C device has responded back with read data.
6	ack_err	The I^{2}C device did not acknowledge the current read cycle.
0	Active	An I^{2}C cycle is in progress.

I2C_ST Module

The state machine used in this module is shown in Figure 2. It starts from the IDLE state and waits for the i2c_go signal from the I2C_WREG module. Once i2c_go is asserted, the state machine goes through various i2c states to ensure proper address/data transfers. The ACK signals are inserted by the serial EPROM, which is the slave I^{2}C device in this design. The I2C_st module provides:

- A state machine to control the I^{2}C cycles
- A bit counter to count the bit phases
- I^{2}C clock generation
- I^{2}C data generation
- Ready generation
- Error generation

Figure 2. State Machine



I2C_WREG Module

The I2C_WREG module provides the word address register. When this register is written to, an i2c_go signal is generated. This causes the state machine to start a random read cycle to the address stored in this register.

A clock, reg_clk_out, is generated in this module to allow the use of input registers for the word address register in certain CPLD/FPGA architectures. In order to use the input registers, this clock signal must be routed to a dedicated clock input pin on the board. Please refer to the device data sheet for architectural details.

A microprocessor acknowledge signal is provided in this module. This is the chip select signal delayed by one clock tick. This signal can be omitted from the design if the function is not needed or is performed elsewhere.

I2C Module

The I2C module provides the top-level logic for the design. It links together the individual modules and tri-state buffer for the microprocessor data. It also provides the open drain outputs for the I^{2}C clock and data signals.

Test Bench Description

The test bench for this design includes the following modules:

- CLK_RST
- I2C_SLAVE
- I2C_TB
- MICRO

CLK_RST Module

The CLK_RST module provides the clock and reset signals to the test bench. Editing the CLK_PERIOD parameter changes the clock frequency. Editing the RESET_TIME parameter changes the duration of reset. The reset_recovery parameter delays the clock until a fixed time, allowing all registers to recover from reset.

I2C_SLAVE Module

The I2C_SLAVE module provides a model of a I^2C memory device which features 256 memory locations. The slave module is able to detect START and STOP commands from the I^2C controller. It generates ACK after the word address cycle and leaves the I^2C bus tristated after data read. The data stored in the memory is the same as its address. This provides a convenient way to check the data by comparing “Slave Data Receive on Write” and “Slave Data Transmitted on Read” messages during simulation.

I2C_TB Module

The I2C_TB module is the top-level test bench for the design. It instantiates the I^2C controller as well as the modules in the test bench.

MICRO Module

The MICRO module provides the microprocessor stimulus to the I^2C controller. This module provides tasks to simulate write and read cycles to the I^2C controller. It also performs error checking on the data read back from the I^2C controller.

Design Flow

Lattice design tools are used for synthesis, place and route and simulation. In addition to the place and route/fitter engine, the Lattice ispLEVER® design tool includes Synplify®/SynplifyPro® from Synplicity®, and Active-HDL® from Aldec®. The details of the design flow can be found in the README.txt file that comes with the reference design.

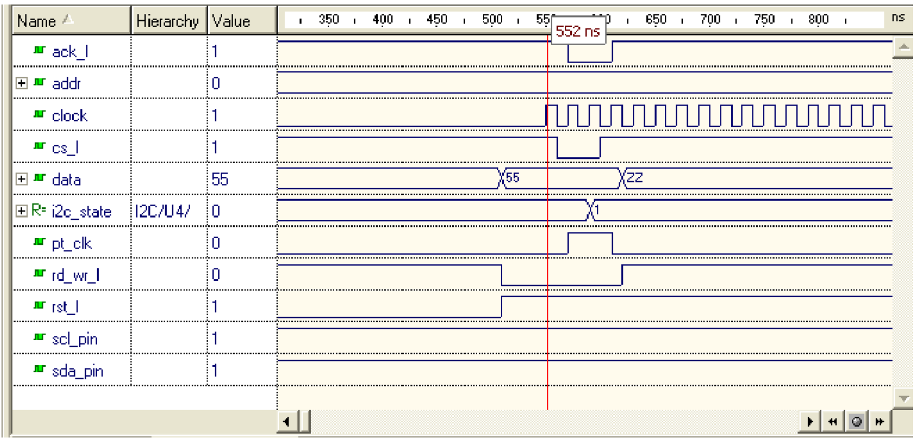
Timing Diagrams

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

Microprocessor Write Cycle

The microprocessor writes a “55” to the word address register. This causes the I^2C controller state machine to start a random read cycle to the I^2C device.

Figure 3. Microprocessor Write Cycle Timing Diagram

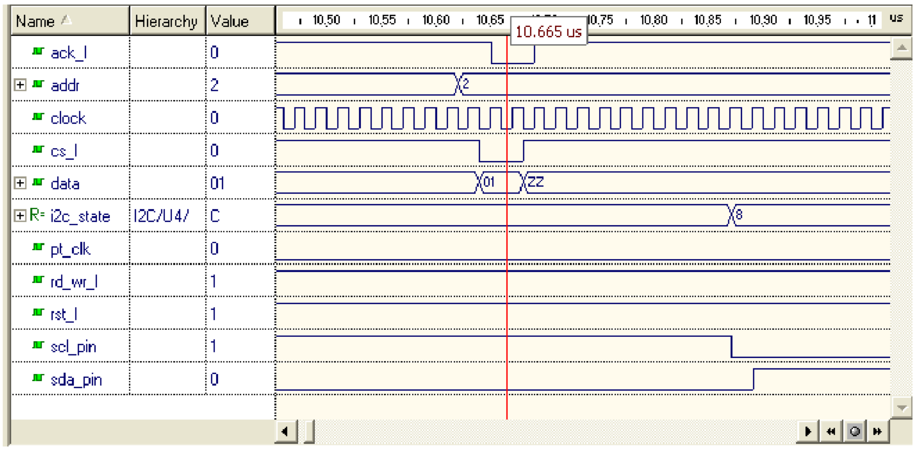


Name	Hierarchy	Value
ack_l		1
addr		0
clock		1
cs_l		1
data		55
i2c_state	I2C/U4/	0
pt_clk		0
rd_wr_l		0
rst_l		1
scl_pin		1
sda_pin		1

Microprocessor Read Cycle

The microprocessor reads the status register. Bit 1 is set indicating an I^2C cycle is in progress. Bits 6 and 7 are cleared. This indicates the cycle has not completed and that an error has not been detected.

Figure 4. Microprocessor Read Cycle Timing Diagram

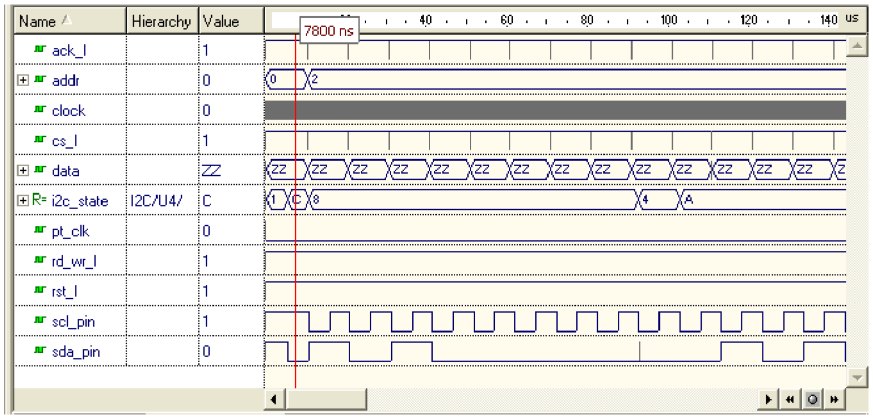


Name	Hierarchy	Value
ack_l		0
addr		2
clock		0
cs_l		0
data		01
i2c_state	I2C/U4/	C
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		1
sda_pin		0

I^{2}C Device Address Write Cycle

An I^{2}C cycle starts, followed by the device address and an acknowledge from the I^{2}C device.

Figure 5. I^{2}C Device Address Write Cycle Timing Diagram

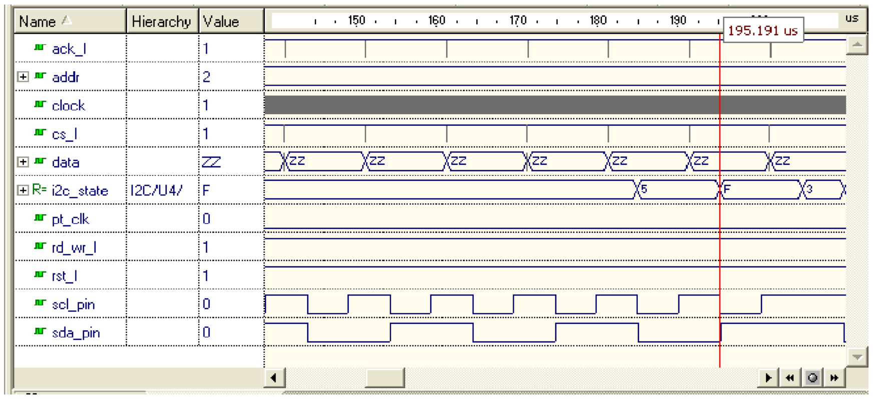


Name	Hierarchy	Value
ack_l		1
addr		0
clock		0
cs_l		1
data		ZZ
i2c_state	I2C/U4/	C
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		1
sda_pin		0

I^{2}C Word Address Cycle

The word address cycle starts, followed by an acknowledge from the I^{2}C device.

Figure 6. I^{2}C Word Address Cycle Timing Diagram

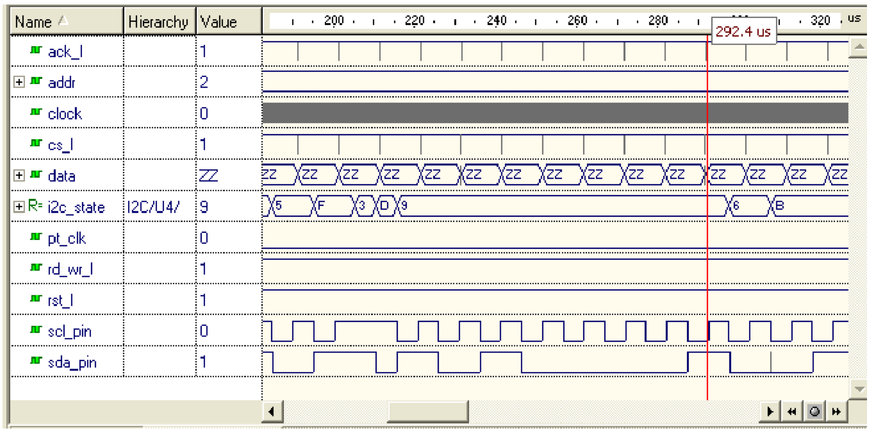


Name	Hierarchy	Value
ack_l		1
addr		2
clock		1
cs_l		1
data		ZZ
i2c_state	I2C/U4/	F
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		0
sda_pin		0

I²C Device Address Read Cycle

The second device address cycle, this time requesting a read.

Figure 7. I²C Device Address Read Cycle Timing Diagram

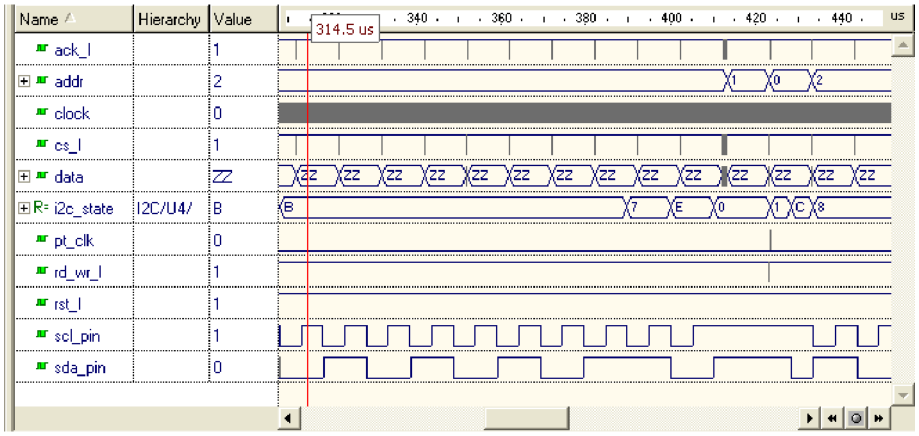


Name	Hierarchy	Value
ack_l		1
addr		2
clock		0
cs_l		1
data		ZZ
i2c_state	I2C/U4/	9
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		0
sda_pin		1

I²C Read Data Cycle

The data is read from the I²C device. This cycle is done without an acknowledge. A stop is asserted by the controller.

Figure 8. I²C Read Data Cycle Timing Diagram

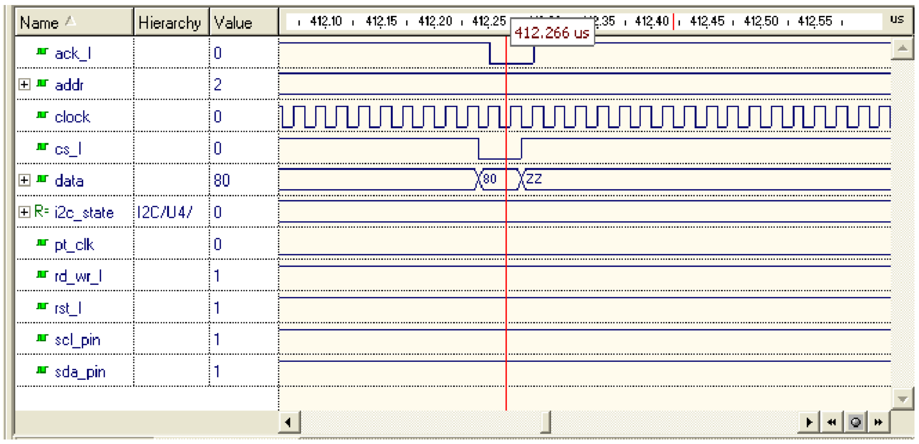


Name	Hierarchy	Value
ack_l		1
addr		2
clock		0
cs_l		1
data		ZZ
i2c_state	I2C/U4/	B
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		1
sda_pin		0

Microprocessor Read Status Cycle

The status register is read indicating the I²C cycle has completed.

Figure 9. Microprocessor Read Status Cycle Timing Diagram

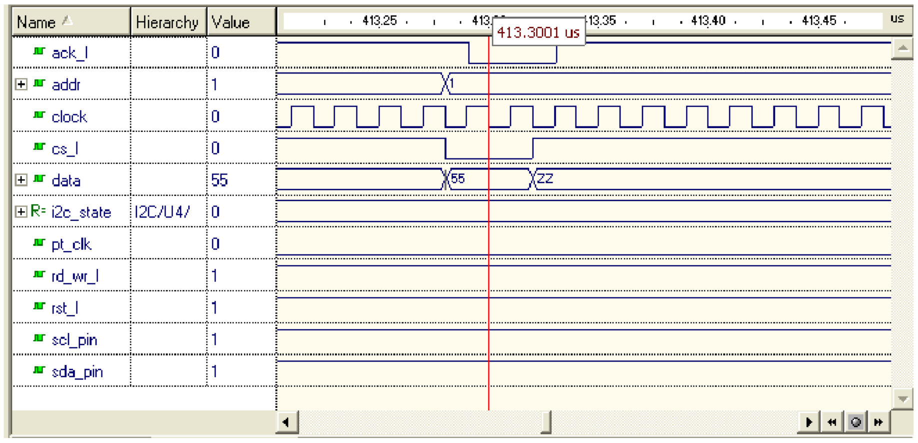


Name	Hierarchy	Value
ack_l		0
addr		2
clock		0
cs_l		0
data		80
i2c_state	I2C/U4/	0
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		1
sda_pin		1

Microprocessor Read Data Cycle

The I²C data is read by the microprocessor.

Figure 10. Microprocessor Read Data Cycle Timing Diagram



Name	Hierarchy	Value
ack_l		0
addr		1
clock		0
cs_l		0
data		55
i2c_state	I2C/U4/	0
pt_clk		0
rd_wr_l		1
rst_l		1
scl_pin		1
sda_pin		1

Implementation

This design is implemented in Verilog and VHDL. When using this design in a different device, density, speed, or grade, performance and utilization may vary. Default settings are used during the fitting of the design.

Table 3. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization	Fmax(MHz)	I/Os	Architecture Resources
ECP5™ [6]	Verilog-LSE	–6	86 LUTs	>50	18	N/A
	Verilog-Syn	–6	87 LUTs	>50	18	N/A
	VHDL-LSE	–6	85 LUTs	>50	18	N/A
	VHDL-Syn	–6	89 LUTs	>50	18	N/A
LatticeECP3™ [3]	Verilog-Syn	–6	96 LUTs	>50	18	N/A
	VHDL-Syn	–6	94 LUTs	>50	18	N/A
MachXO3L™ [7]	Verilog-LSE	–6	85 LUTs	>50	18	N/A
	Verilog-Syn	–6	85 LUTs	>50	18	N/A
	VHDL-LSE	–6	85 LUTs	>50	18	N/A
	VHDL-Syn	–6	86 LUTs	>50	18	N/A
MachXO2™ [1]	Verilog-LSE	–6	85 LUTs	>50	18	N/A
	Verilog-Syn	–6	85 LUTs	>50	18	N/A
	VHDL-LSE	–6	85 LUTs	>50	18	N/A
	VHDL-Syn	–6	86 LUTs	>50	18	N/A
MachXO™ [2]	Verilog-LSE	–3	81 LUTs	>50	18	N/A
	Verilog-Syn	–3	82 LUTs	>50	18	N/A
	VHDL-LSE	–3	81 LUTs	>50	18	N/A
	VHDL-Syn	–3	83 LUTs	>50	18	N/A
LatticeXP2™ [4]	Verilog-Syn	–5	93 LUTs	>50	18	N/A
	VHDL-Syn	–5	93 LUTs	>50	18	N/A
ispMACH® 4000ZE[5]	Verilog	–5ns	64 Macrocells	>50	18	N/A
	VHDL	–5ns	64 Macrocells	>50	18	N/A

1. Performance and utilization characteristics are generated using LCMXO2-1200HC-6TG144C with Lattice Diamond® 3.3 design software with LSE (Lattice Synthesis Engine) and Synplify Pro®.
2. Performance and utilization characteristics are generated using LCMXO256E-3T100C with Lattice Diamond 3.3 design software with LSE and Synplify Pro.
3. Performance and tilization characteristics are genereted using LFE3-17EA-6FTN256C with Lattice Diamond 3.3 design software with Synplify Pro
4. Performance and utilization characteristics are generated using LFXP2-5E-5M132C with Lattice Diamond 3.3 design software with Synplify Pro.
5. Performance and utilization characteristics are generated using LC4256ZE-5TN100C with Lattice Diamond 3.3 design software with LSE.
6. Performance and utilization characteristics are generated using LFE5U-45F-6MG285C with Lattice Diamond 3.3 design software with LSE and Synplify Pro.
7. Performance and utilization characteristics are generated using LCMXO3L-4300C-6BG256C with Lattice Diamond 3.3 design software with LSE and Synplify Pro.

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
January 2015	2.7	Updated Table 3, Performance and Resource Utilization.
		— Added support for Lattice Diamond 3.3 design software.
		— Added support for LSE and Synplify Pro.
March 2014	02.6	Updated Table 3, Performance and Resource Utilization.
		— Added support for ECP5 device family.
		— Added support for MachXO3L device family.
		— Added support for Lattice Diamond 3.1 design software.
		Updated corporate logo.
		Updated Technical Support Assistance information.
November 2010	02.4	Added support for the MachXO2 device family.
		Updated to support Lattice Diamond 1.1 design software.
		Updated to support ispLEVER 8.1 SP1 design software.
December 2009	02.3	Added support for LatticeXP2 device family.
		Updated for ispLEVER 8.0.
September 2009	02.2	Design updated to include ispMACH 4000ZE support. Added VHDL source file and testbench.
February 2009	02.1	Design updated to include MachXO support. Document updates to reflect the design/tool change.
—	—	Previous Lattice releases.