SPI Flash Controller with Wear Leveling

November 2010 Reference Design RD1102

Introduction

Flash memory has been widely used in embedded systems to support various functions in products like consumer electronics. How to effectively manage Flash memory and extend the service cycles of a Flash memory has become the challenge faced by designers.

The maximum number of erase cycles for each sector or block of Flash memory is close to 100,000. For most applications, a master device often accesses and updates a few specific sectors. These sectors can wear out in a short period of time while the rest of the sectors are still valid for applications. Such behavior significantly reduces the lifetime of Flash memory and impacts overall product cost. Wear leveling is a technique to extend the service cycles of Flash memory by averaging the number of accesses to each sector. As a result, the number of erase cycles is distributed among all the sectors, thus extending the life of each sector of the Flash memory.

This reference design implements the wear leveling control of data storage for SPI Flash memory. The CPU stores the number of erases, logic-map-physical table, and the valid page pointers into Embedded Block RAM (EBR) or User Flash Memory (UFM) to keep track of the SPI Flash memory usage. A WISHBONE bus is used to interface between the master device and the wear leveling controller.

Features

This design provides the following features to manage the SPI Flash sector usage:

- Configurable registers for command and data transfer
- Manages number of erases for each sector
- Manages logic-map-physical tables for the Flash memory
- Provides valid page pointers
- Supports SPI interface
- WISHBONE compliant

Functional Description

The wear leveling algorithm is composed of two parts, the dynamic algorithm and the static algorithm. The dynamic algorithm is used to manage frequently-used sectors. The static algorithm is used to manage the read-only sectors or sectors that are seldom updated.

In order to implement the algorithm, the controller must record the number of erases for every sector, provide the logic-map-physical table, keep track of the page valid pointers to the EBR memory, and process the information before the next operation.

This reference design works with a CPU or microcontroller to implement the wear leveling algorithm. The CPU stores the required data of the algorithm into the corresponding memory.

The targeted Flash memory in this design is the Numonyx® Forté™ M25P40 device, a serial Flash memory. The total size of the Flash memory is 4 Mbits. This memory is organized into eight sectors, each contains 256 pages. Each page is 256 bytes in size. This Flash device can be driven by a master controller with a SPI interface. This reference design implements a SPI Master Controller to interface the Flash memory.

The CPU configures the command register to trigger different operations. If a sector erase command is issued, the number of erases for the sector is incremented by 1, and this number is stored in EBR memory. At the same time, the page valid pointers are updated to indicate that all pages of that sector are invalid.

If a bulk-erase command is issued, the number of erases for every sector is read and the new number is recorded.

If the page-program command is issued, the corresponding page pointer bit is updated to indicate a valid page. At the same time, the data is stored in the page-valid-memory. Before the page-program command is sent, the CPU must write the physical address to the corresponding memory.

Before the power is turned off, the CPU reads the information of the SPI sector/page usage from the EBR memory and writes to the UFM memory. At the next power-up, the data contained in the UFM will initialize the EBR memory automatically.

Figure 1 is a block diagram of the design.

Figure 1. SPI Flash Controller with Wear Leveling Block Diagram

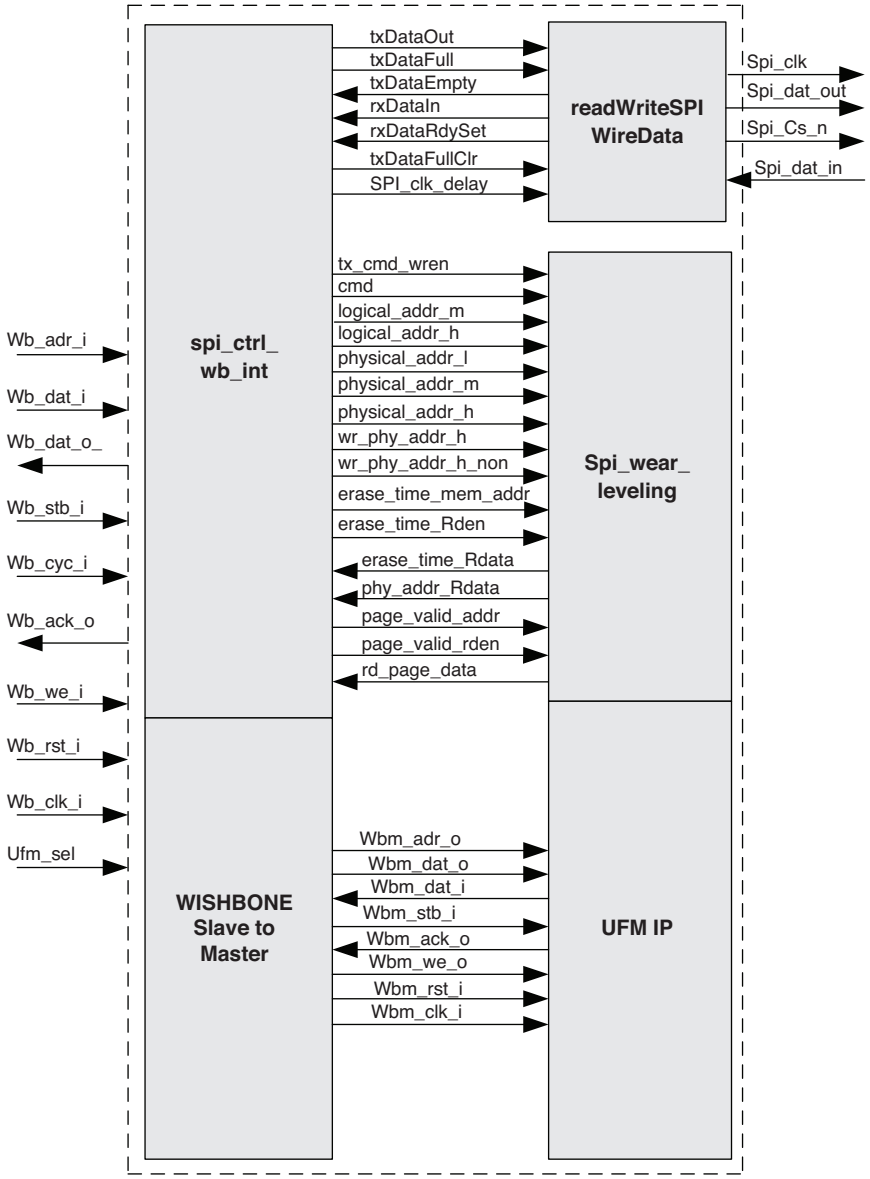


Table 1 provides the pin descriptions for this reference design.

Table 1. Pin Descriptions

Port	direction	Width	Description
WISHBONE Interface			
wb_clk_i	Input	1	WISHBONE clock input
wb_rst_i	Input	1	WISHBONE reset input, active-high synchronous reset signal
wb_adr_i	Input	8	WISHBONE input address bits
wb_dat_i	Input	8	WISHBONE input data to the core
wb_dat_o	Output	8	WISHBONE output data from the core
wb_we_i	Input	1	WISHBONE input write enable signal
wb_stb_i	Input	1	WISHBONE input strobe signal
wb_ack_o	Output	1	WISHBONE output bus cycle acknowledge output
SPI Flash Interface			
spiClkOut	Output	1	SPI clock, clock speed configurable
spiDataIn	Input	1	SPI serial data from slave
spiDataOut	Output	1	SPI serial data to slave
spiCsn	Output	1	SPI Flash device chip select
Generic Functions			
ufm_sel	Input	1	Switch signal used to select the UFM IP. Must be set to ‘1’ to write UFM memory.

Module Descriptions

This reference design can be divided into four modules:

- spi_ctrl_wb_int
- spi_wear_leveling
- readWriteSPIWireData
- UFM

spi_ctrl_wb_int Module

The spi_ctrl_wb_int module is used for the reading and writing of all user registers. According to the configuration command, this module can generate different control signals to trigger different SPI Flash operations. The CPU reads the status register and the corresponding data registers to determine the wear leveling of each sector. This reference design includes 20 registers. Table 2 provides a list of the register descriptions.

Table 2. User Register Descriptions

Register Name	Address	Width	Access	Description
command	0x77	8	Read/Write	Command register. Users can write to this register to transmit command code.
logical_addr_m	0x78	8	Read/Write	Logical address for the middle byte register.
logical_addr_h	0x79	8	Read/Write	Logical address for the high byte register. This register and the logical_addr_m register form one logical page address.
physical_addr_l	0x80	8	Read/Write	Physical address for low byte registers. This register, physical_addr_m and physical_addr_h comprise one byte accession address.
physical_addr_m	0x81	8	Read/Write	Physical address for middle byte registers.
physical_addr_h	0x82	8	Read/write	Physical address for high byte registers. This register and the physical_addr_m register comprise one physical page address.

Table 2. User Register Descriptions (Continued)

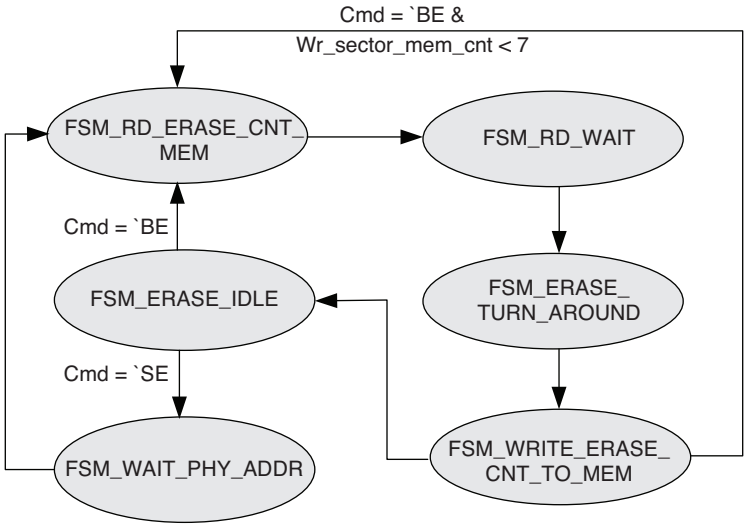
Register Name	Address	Width	Access	Description
write_data	0x83	8	Read/Write	Write data registers. The CPU can configure this register to transfer a data byte to SPI Flash memory.
erase_times_mem_rd	0x084	8	Read/Write	Erasing sector memory reading register. The user can configure this register to generate a read address and read enable signal for erasing number memory. The eighth bit is the read enable bit. The first, second and third bits comprise the read address for erasing memory.
page_valid_addr	0x85	8	Read/Write	This register is configured to generate reading address of page-valid-pointer memory. **0x00~0x1F**: These addresses are used for reading pages of the first sector. The lowest bit in the address, 0x00, denotes the valid first page of the first sector. The highest bit in the address, 0x1F, denotes the valid of last page of the first sector. … **0xE0~0xFF**: These addresses are used for reading pages of the eighth sector. The lowest bit in the address, 0xE0, denotes the valid first page of the eighth sector. The highest bit in the address, 0xFF, denotes the valid last page of the eighth sector.
page_valid_rden	0x86	8	Read/Write	This register is used to generate the read enable signal of the page valid pointer.
SPI_Clk_delay	0x87	8	Read/Write	This register is used to generate the SPI clock.
RxDataOut	0x88	8	Read	This register is used for sending the SPI data to the CPU.
erase_mem_rdata	0x89	8	Read	This register is used for sending the low byte number of erases to the CPU.
erase_mem_rdata	0x90	8	Read	This register is used for sending the middle byte number of erases to the user.
erase_mem_rdata	0x91	8	Read	This register is used for sending the high byte number of erases to the user.
command_valid	0x92	8	Read/Write	This register is used to indicate whether the current command should be transmitted to the SPI Flash.
phy_addr_rdata	0x93,0x94	8	Read	This register is used for sending the low byte of the physical page address to the user.
rd_page _data	0x95	8	Read	This register is used for sending the page valid pointer to the user.
status	0xFF	8	Read	This is the read-only register that indicates the status of activities between the CPU and the SPI Flash.

spi_wear_leveling Module

The spi_wear_leveling module manages three EBR memories. The erase numbers for all sectors are stored in sector_erase_cnt_memory. The logical-map-physical table is stored in phy_addr_mem. The page valid pointer of every sector is stored in page_valid_mem.

The READ/WRITE operation of sector_erase_cnt_memory is triggered by the erase command. The address of this memory corresponds to the sector number of the M25P40 device. The number of erases is stored into the corresponding address. The flow of these operations shown in Figure 2.

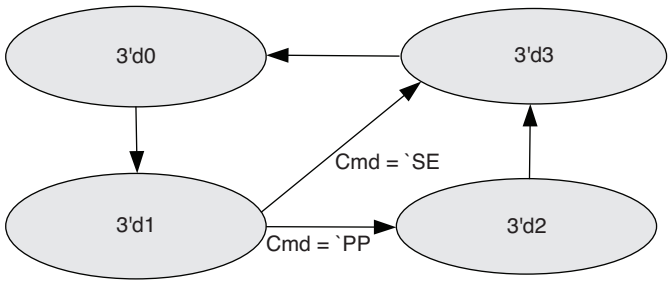
Figure 2. READ/WRITE of sector_erase_cnt_memory



There are 2,048 pages in the selected SPI Flash device. The physical page address is 11 bits in length and is stored as two bytes of data. The address of this memory is the logical address. The physical address is the data to be stored in this memory.

The page_valid_mem register is used to store the page-valid-pointer data. Once a page is written by the user, the corresponding page-valid-pointer bit is updated to '1'. Once the sector is erased, all pages of that sector are invalid and the corresponding page pointers are updated to all '0'. The user can read this memory to learn about the status of every page. The flow of these operations is shown in Figure 3.

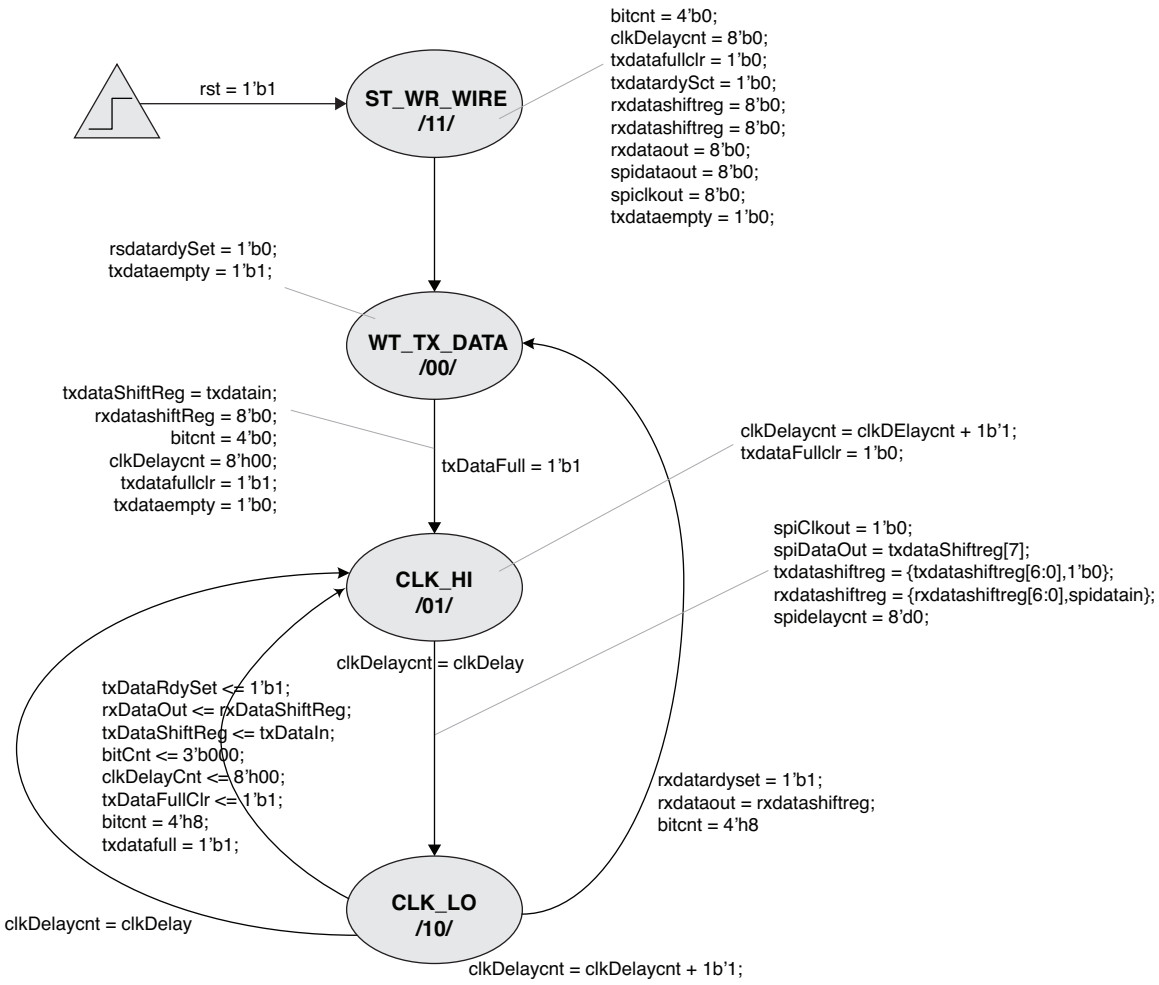
Figure 3. WRITE page_valid_pointer



ReadWriteSPIWireData Module

This module acts as a SPI Flash Memory Controller. The serial interface transmits and receives the serial bit to/from the SPI Flash. Figure 4 is the readWriteSPIWireData module state machine.

Figure 4. Figure 4 ReadWriteSPIWireData Operation State Machine



UFM Module

UFM-hardened IP is built into Lattice MachXO2™ devices. The WISHBONE interface is used to control the transfer of commands to the UFM memory.

Operation Sequence

1. **Sector Erase Operation**
   a. Write the command_valid register (8'h14) with 0x01.
   b. Write the command register with 0xD8.
   c. Read the status register to judge whether the 0xD8 instruction has been transmitted.
   d. Write the middle byte of the physical address with the expected address.
   e. Read the status register to judge whether the mid byte address has been transmitted.
   f. Write the high byte of the physical address with the expected address.
   g. Read the status register to judge whether the high byte address has been transmitted.
   h. Update the page valid pointer of the erased sector.
   i. Read the erased number of the latest sector to determine whether the data has been updated.
   j. Write the command_valid register (8'h14) with 0x00.

2. **Bulk Erase Operation**
   a. Write the command_valid register (8'h14) with 0x01.
   b. Write the command register with 0xC7.
   c. Read the status register to judge whether the 0xC7 instruction has been transmitted.
   d. Read the erased number of all sectors to determine whether the data has been updated.
   e. Write the command_valid register (8'h14) with 0x00.

3. **Page Program Operation**
   a. Update the logical-map-physical table.
   b. Write the command_valid register (8'h14) with 0x01.
   c. Write the command register with 0x02.
   d. Write the low byte address of the expected physical address.
   e. Read the status register to determine whether the 0x02 instruction has been transmitted.
   f. Write the mid byte address of the expected physical address.
   g. Read the status register to determine whether the 0x02 instruction has been transmitted.
   h. Write the high byte address of the expected physical address.
   i. Read the status register to determine whether the 0x02 instruction has been transmitted.
   j. Update the page valid memory.
   k. Write the data register with the expected data.
   l. Read the page-valid memory to determine the status of the latest written pages.
   m. Write the command_valid register (8'h14) with 0x00.

4. **UFM Operation**
   For information on the operation of the UFM IP, refer to TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices.

Timing Specifications

Figure 5. Erase Number Memory READ/WRITE Waveform

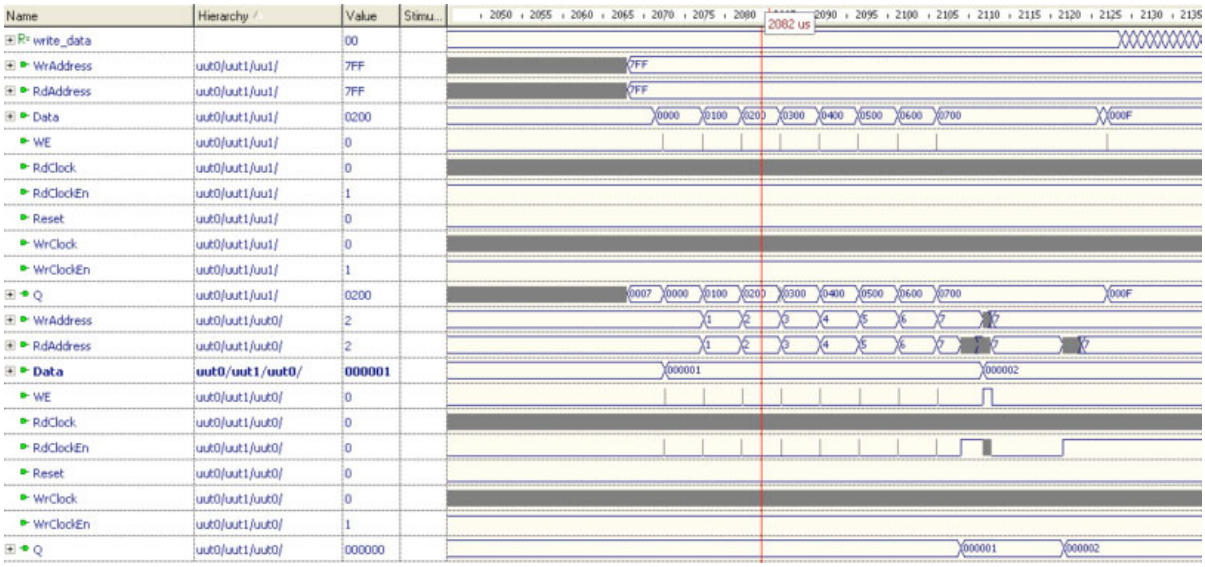


Figure 6. Logical-Physical Table Memory READ/WRITE Waveform

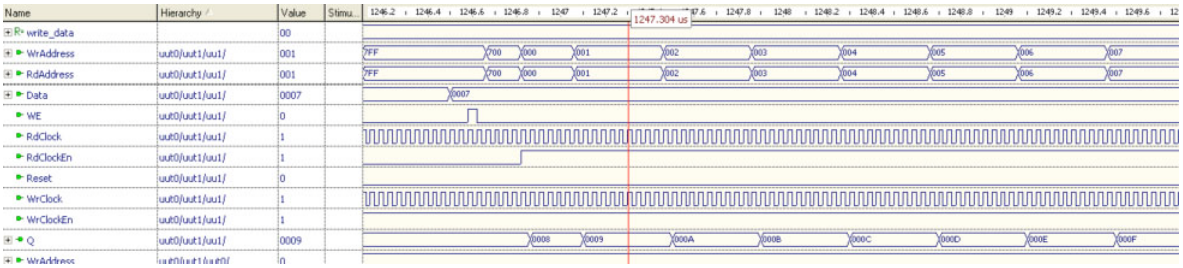


Figure 7. Page-Valid-Pointer Memory READ/WRITE Waveform

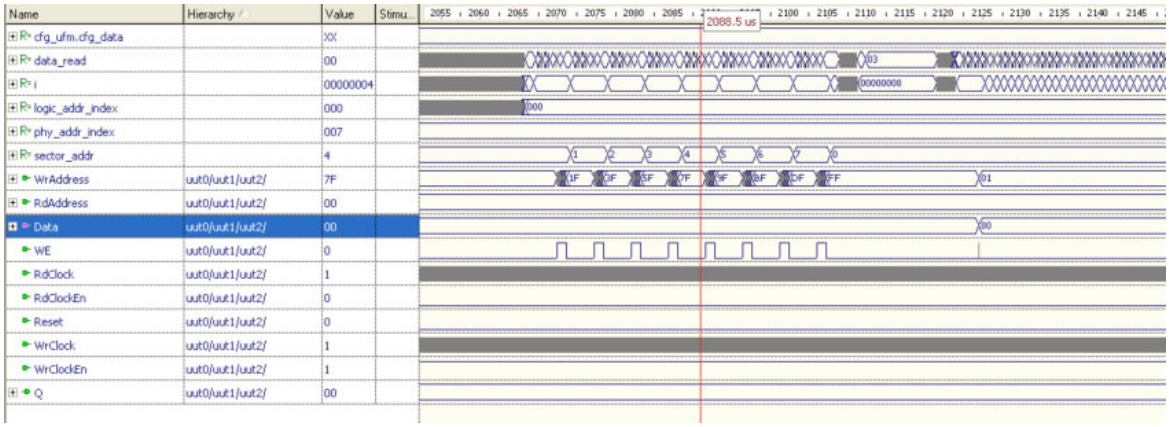
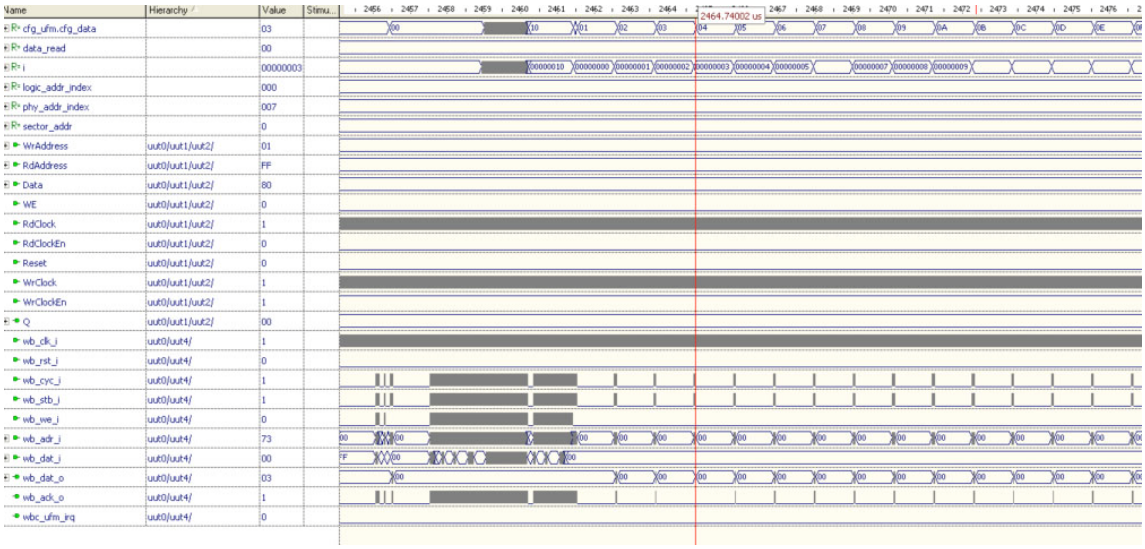


Figure 8. UFM READ/WRITE Waveform



Implementation

Table 3. Performance and Resource Utilization

Device Family	Language	Speed Grade	I/Os	f_{MAX}(MHz)	Utilization (LUTs)	Architecture Resources
MachXO2[1]	Verilog	-5	37	> 24	359	7 EBRs

1. Performance and utilization characteristics are generated using LCMXO2-1200HC-5TG100CES, with Lattice Diamond™ 1.1 and isp-LEVER® 8.1 SP1 design software. When using this design in a different device, density, speed, or grade, performance and utilization may vary.

References

- TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices
- Numonyx Forté Serial Flash Memory M25P40 Data Sheet
- RD1048, SD Flash Controller

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
November 2010	01.0	Initial release.