Design and FPGA Validation of SHAKTI SPI controller for ADCs, Flash-Memories, and Spi-RAM

A Project Report

submitted by

KAUSTUBH GHORMADE

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Design and FPGA Validation of SHAKTI SPI controller for ADCs, Flash-Memories, and Spi-RAM**, submitted by **KAUSTUBH GHORMADE**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master Of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V. Kamakoti
Research Guide
Professor
Dept. of Physics
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to Prof. V. Kamakoti for providing me with all the necessary facilities for the project.

I place on record, my sincere thank you to Prof. V. Janankiraman for the continuous encouragement.

I am also grateful to SHAKTI hardware and software team. I am extremely thankful and indebted to them for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

ABSTRACT

In this thesis will present a serial peripheral interface (SPI) controller designed to work with SHAKTI class of processors. The SPI interface is used to communicate with external devices like ADC, Potentiometer, SPI Flash etc... The thesis describes the design of SPI controller in Bluespec System Verilog (BSV). Bluespec is high level hardware description language based on IEEE system verilog HDL. It also describes the steps to configure and use the programmable serial peripheral controller.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The purpose of this application note is to describe the serial peripheral interface bus controller (SPI). The SPI interface can be used to communicate with external devices using the SPI protocol. It describes how to configure and use the programmable serial peripheral interface.

1.1 Definition of the SPI

The Serial Peripheral Interface (SPI) protocol is asynchronous serial data standard, primarily used to allow a microprocessor to communicate with other microprocessors or ICs such as memories (like flash, serial-RAM), liquid crystal diodes (LCD), analog-to-digital converter subsystems (ADC), etc.
The SPI is a very simple synchronous serial data, master/slave protocol based on four lines:

- Clock Line (SCLK)
- Serial Output (MOSI)
- Serial Input (MISO)
- Slave Select (NSS)

Every SPI system consists of one master and one or more slaves, where a master initiates the communication by asserting the NSS line. When a slave device is selected, the master starts clocking out the data through the MOSI line to the selected slave device. The master sends and receives one bit for every clock edge. One byte can be exchanged in eight clock cycles. The master finishes communication by de-asserting the NSS line. The SPI is a primitive protocol without an acknowledgement mechanism for checking received or sent data. For safe communication, a flow control has to be implemented in the communications protocol on s a higher level.

1.2 SPI main features

- Master or slave operation
- Full-duplex synchronous transfers on three lines
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- SPI bus busy status flag
- 8 master mode baud rate prescalers up to $f_{CLK}/2$.
- Dedicated transmission and reception flags with interrupt capability
- Master mode fault, overrun flags and alarm flag for both FIFO with interrupt capability

CHAPTER 2

DESIGN OF SPI CONTROLLER

2.1 Block diagram of SPI

The SPI allows synchronous, serial communication between the MCU and external devices. Application software can manage the communication by polling the status flag or using dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram Figure 2.1.

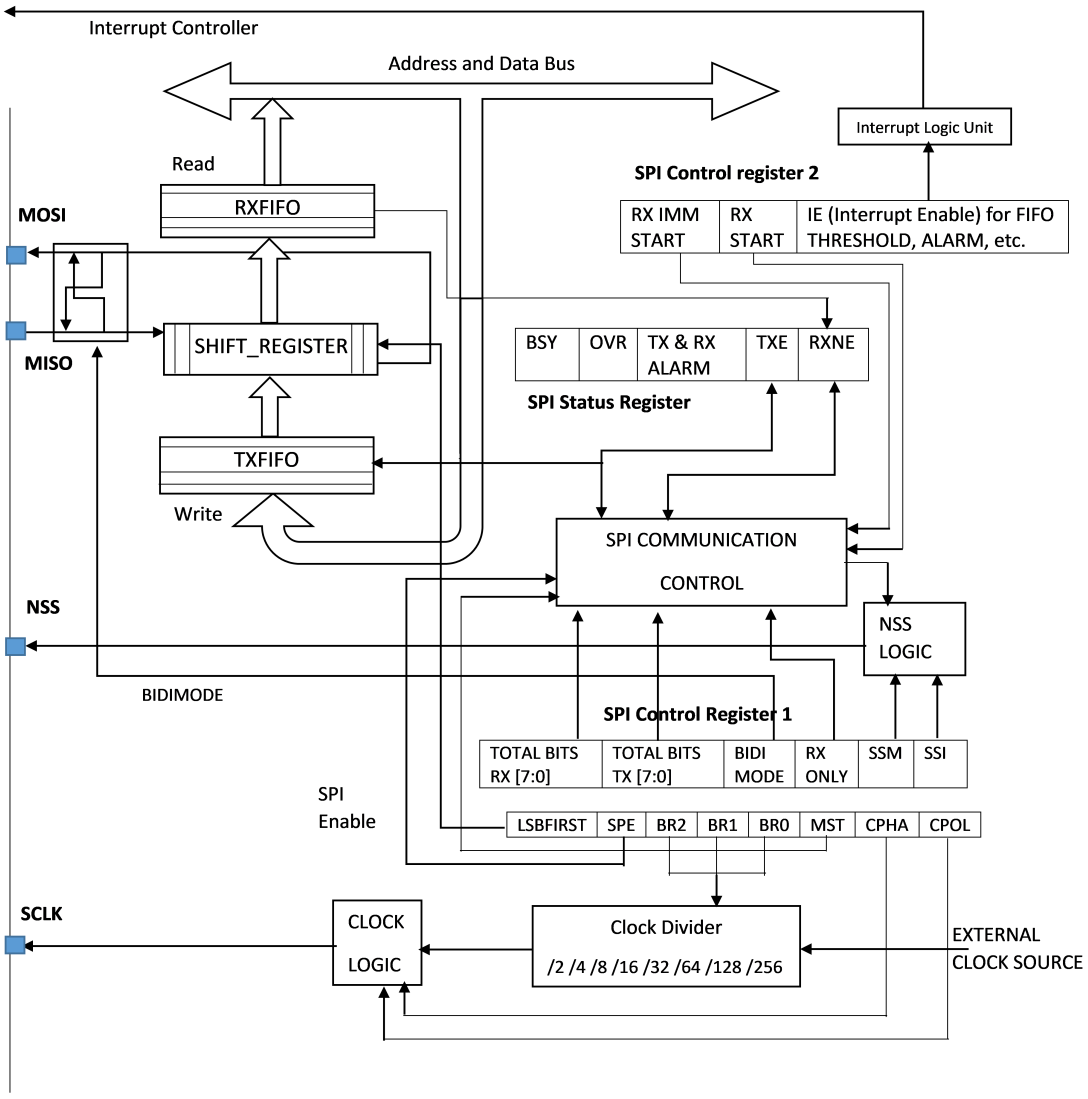

Figure 2.1: Block Diagram of SPI

2.2 Description of spi module

The SPI is capable of master and receive only mode . The SPI have two embedded FIFO for Transmit and Receive each.The size of each FIFO slice is fully user-programmable, depending on the need. The BSV code for SPI controller can be found on here. The E-class Shakti SOC aardonyx uses the SPI controller.

2.2.1 Signal description and connection scheme

The serial peripheral interface bus has four external lines. Four I/O pins are dedicated to SPI communication with external devices.

- **MISO**: Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI**: Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK**: Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **NSS**: Slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame or
  - detect a conflict between multiple masters

Figure 1 shows how the slave device is connected to the master in the single master, single slave SPI implementation.Multiple slave SPI implementation is not supported by the SPI without additional external hardware.

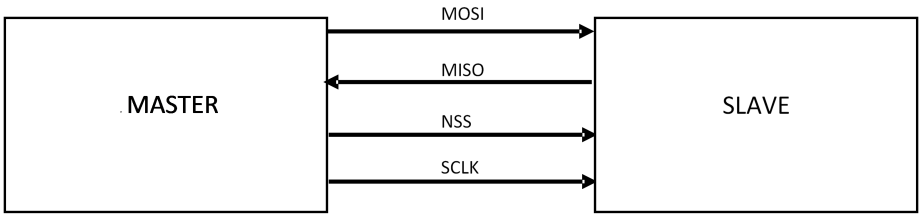

Figure 2.2: Single master, single slave SPI implementation

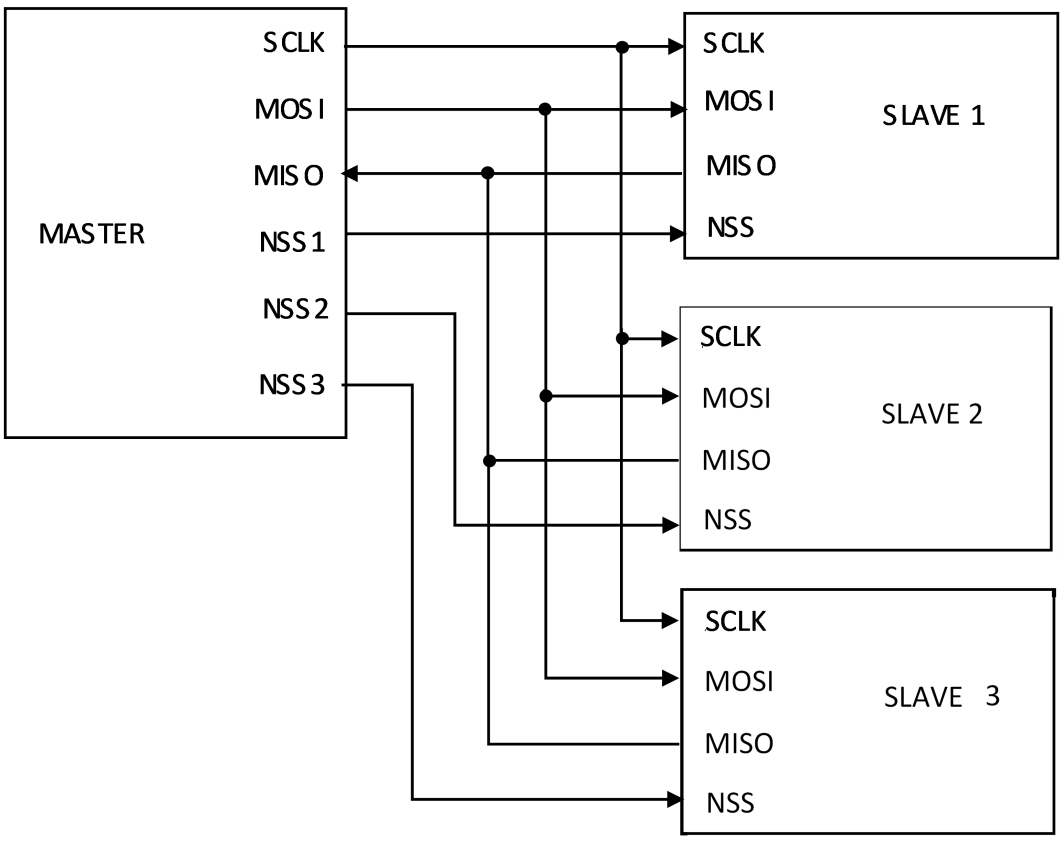

Figure 2.3: Multiple slave SPI implementations

2.2.2 Clock divider and clock logic

The clock logic component selected the frequency of the clock with the help of clock divider module. The polarity of the clock was selected with the help of CPOL and the phase of the clock used the CPHA bit. The clock phase and polarity could be modified for SPI data transfers. To be able to communicate together,the master and slaves devices must follow the same communication format.

Clock divider/Baud rate generator

The clock frequency was changed by dividing down the clock input signal. Three bits BR[2:0] in the SPI_CR1 register establish the value of which the bus clock is divided. This provides the end-user ample choices for baud rate generation with divisors ranging from 2 to 256. The baud rate generator is active only if the SPI is operating in the master mode.

The frequency of communication can be calculated with the following equation:

$$f_{SCLK} = \frac{f_{CLK}}{2^{BR[2:0]+1}}$$

Clock phase and polarity control

The clock can be set to one of the four basic configuration defined in Motorola spi specification. Any one timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPIx_CR1 register.

- The **CPOL** bit sets the polarity of the clock signal during the idle state.
- The **CPHA** bit controls the edge on which the SCK pin captures the first data bit transacted

Data are latched on each occurrence of this clock transition type.

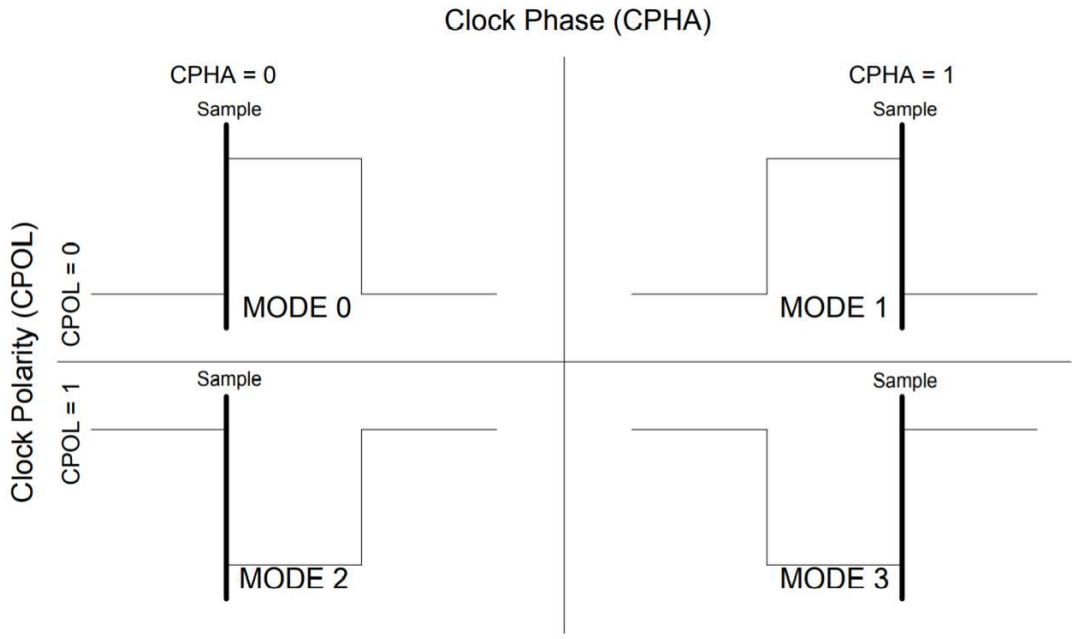

Figure 2.4: The four SPI-modes corresponding to the states of CPHA and CPOL parameters.

2.2.3 Shift-register and FIFO

The transmit shift register controlled the shift and load operations of the SPI. It also monitored the SPI bus and determined when a byte transfer was complete.Each direction has its own FIFO called TXFIFO and RXFIFO.

Shift-register

The SPI transmit shift register was an 8-bit loadable shift register. This shift register was loaded from the TX FIFO, via a load signal generated by the SPI communication control state machine. In case receiving data from slave, it transfer the data to RX FIFO in 8 bit format.

FIFOs

These FIFOs are used in all SPI modes. Both FIFOs are 8 bit wide and the depth of TX FIFO and RX FIFO both are user-programmable. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short. Each FIFO has an ALARM bit set when FIFO reaches 80% of total DEPTH defined by the user. For monitoring the status of FIFO, FRLVL[1:0] and FTLVL[1:0] bits are described in the status register.

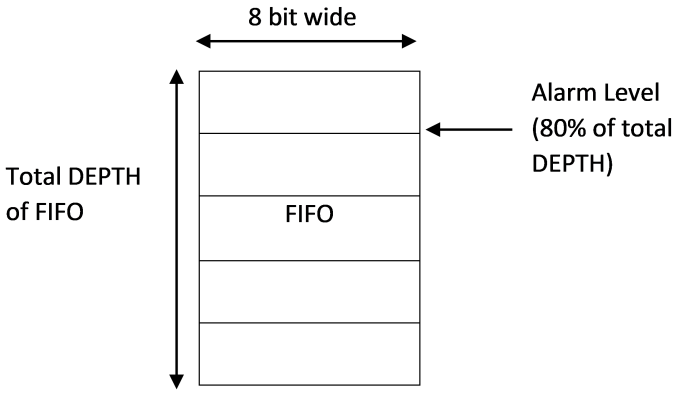

Figure 2.5: FIFO and its parameter

2.2.4 SPI communication control

This is one of the main blocks in the SPI Module. It is a state machine which produces all the control signal for SPI. The user can choose how to get the data from the memory to the FIFO or how to put data in the memory from the FIFO. It also generate all the status flag for communication like BSY, TXE, RXNE etc. The SPI-protocol is highly flexible regarding the length of the data word, transmission length and transmission periodization.

Slave select (NSS) pin management

In master mode, NSS can be used either as output or input. As an input it can prevent multi master bus collision, and as an output it can drive a slave select signal of a single slave. In slave mode, the NSS works as a standard "chip select" input and lets the slave communicate with the master. Hardware or software slave select management can be set using the SSM bit in the SPIx_CR1 register:

- Hardware NSS management (SSM = 0): The NSS pin is managed by the hardware. The NSS signal is driven low as soon as the SPI is enabled in master mode, and is kept low until the SPI is disabled.
- Software NSS management (SSM = 1): in this configuration, slave select information is driven internally by the SSI bit value in register SPIx_CR1. The external NSS pin is free for other application uses.

Status flags

Different status flags are provided for the application to completely monitor the state of the SPI bus:

- **TXE**: This bit is set when transmission TXFIFO has finished loading the data from data register. Now, data register can be loaded again to send more data. An interrupt can be generated if the TXE_IE bit in the SPI_CR2 register is set. The bit is cleared automatically when the transmission is completed.
- **RXNE**: This bit is set when receive RXFIFO has finished sending the data to data register. Now data register can be read. An interrupt can be generated if the RXNE_IE bit in the SPI_CR2 register is set. The RXNE is cleared by hardware automatically when the above conditions are no longer true.
- **BSY**: This flag is set and cleared by hardware. When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy).It can be used in

certain modes to detect the end of a transfer and it is also useful for preventing write collisions in a multi-master system.

Error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the ERR_IE bit.

- **Overrun flag (OVR)**: An overrun condition occurs when data is transferred/received by a master or slave and the FIFO has not enough space to store this transferred/received data. Both FIFO have separate flags as TX_OVR and RX_OVR for this error. When an overrun condition occurs, the newly received value does not overwrite the previous one in the FIFO. The newly received value is discarded and all data transmitted subsequently is lost.
- **Mode fault (MODF)**: Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. This automatically sets the MODF bit.
- **TI mode frame format error (FRE)**: A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the FRE flag is set in the SPI_SR register.

Data frame format

The data byte is parallel-loaded into the shift register (from the internal bus) during the first-bit transmission and then shifted out serially to the MOSI pin MSB first or LSB first depending on the LSB FIRST bit in the SPI_CR1 register.The data frame size is chosen by using the TOTAL_BITS_TX[7:0] and TOTAL_BITS_RX[7:0] bits. Both are 8 bit wide, so the number of bits to be transmit or receive goes up to maximum of 255 bits.

Interrupt Events

Interrupts are enabled by writing one to the proper position in the SPIx_CR2 register. Interrupts can be generated for following events:

Interrupt Event	Enable Control Bit	Description
Transmit TXFIFO ready to be loaded	TXE_IE	TXFIFO has finished loading the data from data register
Data received in Receive RXFIFO	RXNE_IE	RXFIFO has finished sending the data to data register
Over-run, mode fault and Frame format error	ERR_IE	The SPI generates an error interrupt when error flag is set.
Alarm Interrupt	TX_ALARM_IE, RX_ALARM_IE	The SPI generates an alarm interrupt when the amount of data in the FIFO reaches the alarm level.
Full interrupt	TX_FIFO_FULL, RX_FIFO_FULL	the SPI generates a full interrupt when the FIFO is full
Empty interrupt	TX_FIFO_EMPT, RX_FIFO_EMPT	the SPI generates a empty interrupt when the FIFO is empty

Table 2.1: SPI interrupt requests

2.2.5 SPI registers

SPI has a set of control register, status register, and various data registers. The control register one used for setting up the spi communication. The control register two is mainly used for interrupt enable. The status register provides the various flags for monitoring SPI. The data register is used for sending/loading data to/from FIFO. The detailed explanation of them is in chapter 3.

Address offset	Data Width	Permission	Description
0x00	4 bytes	RW	SPI Configuration Register 1
0x04	4 bytes	RW	SPI Configuration Register 2
0x08	4 bytes	RO	SPI Status Register
0x0C	4 bytes	RW	SPI Data Register 1
0x10	4 bytes	RW	SPI Data Register 2
0x14	4 bytes	RW	SPI Data Register 3
0x18	4 bytes	RW	SPI Data Register 4
0x1C	4 bytes	RW	SPI Data Register 5

Table 2.2: SPI register memory map

2.3 Modes of operation

The user can choose between active polling and interrupt mode. The main differences, advantages, and disadvantages are:

- **Polling mode**: core periodically checks status of TX and RX FIFO and writes data to TX FIFO or reads data from Rx FIFO. The main disadvantage of this mode is core loading.
- **Interrupt mode**: lower core loading compared to active polling mode. The core is not used for periodical status checks of the TX and RX FIFO, but it is used for handling interrupt events (sending and receiving data). The number of interrupts depends on the user application. Usage of FIFO alarm level interrupts can decrease the number of requested interrupts.

Note: When data is less, the polling loop wastes less time per transfer than an interrupt call overhead.

2.3.1 Polling Mode

This mode is explained with the help of figure 2.6. In polling mode, to check the status of SPI, the core has to generate read requests to Status register, which unnecessarily stall the core. For example, for reading the receive data from the data register, RXNE should set. To check if RXNE is one or zero, read request has to be generated for SPIx_SR register and keep generating until it is one.

2.3.2 Interrupt Mode

Interrupt mode is explained in figure 2.7. In this mode, instead of keep checking the status register, we check it once the interrupt is there. By reading the SPRx_SR register, we know due to which flag interrupt is generated, and appropriate action can be taken to serve the interrupt. This ensures that the core generates read requests to the status register, only when needed.

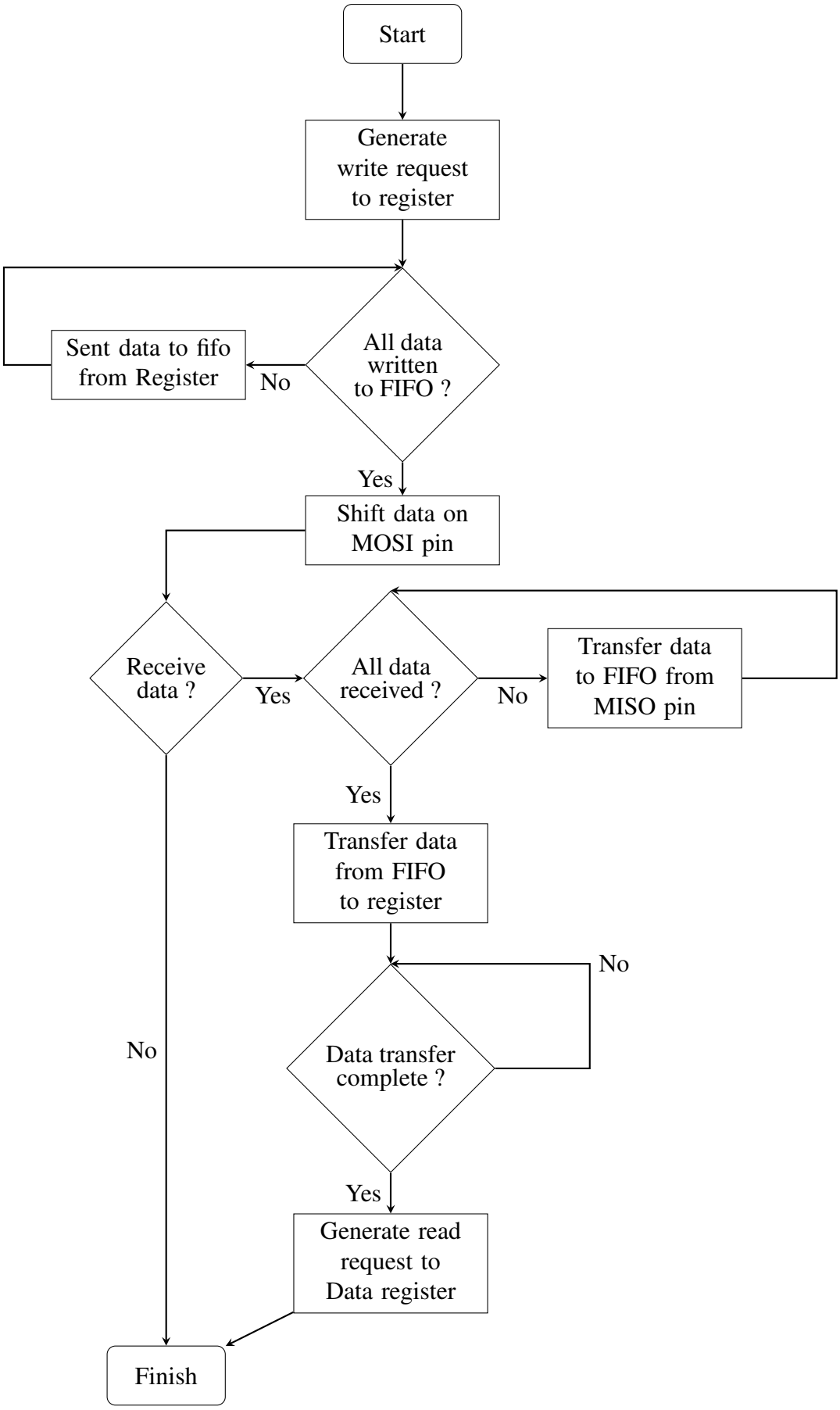

Figure 2.6: polling mode

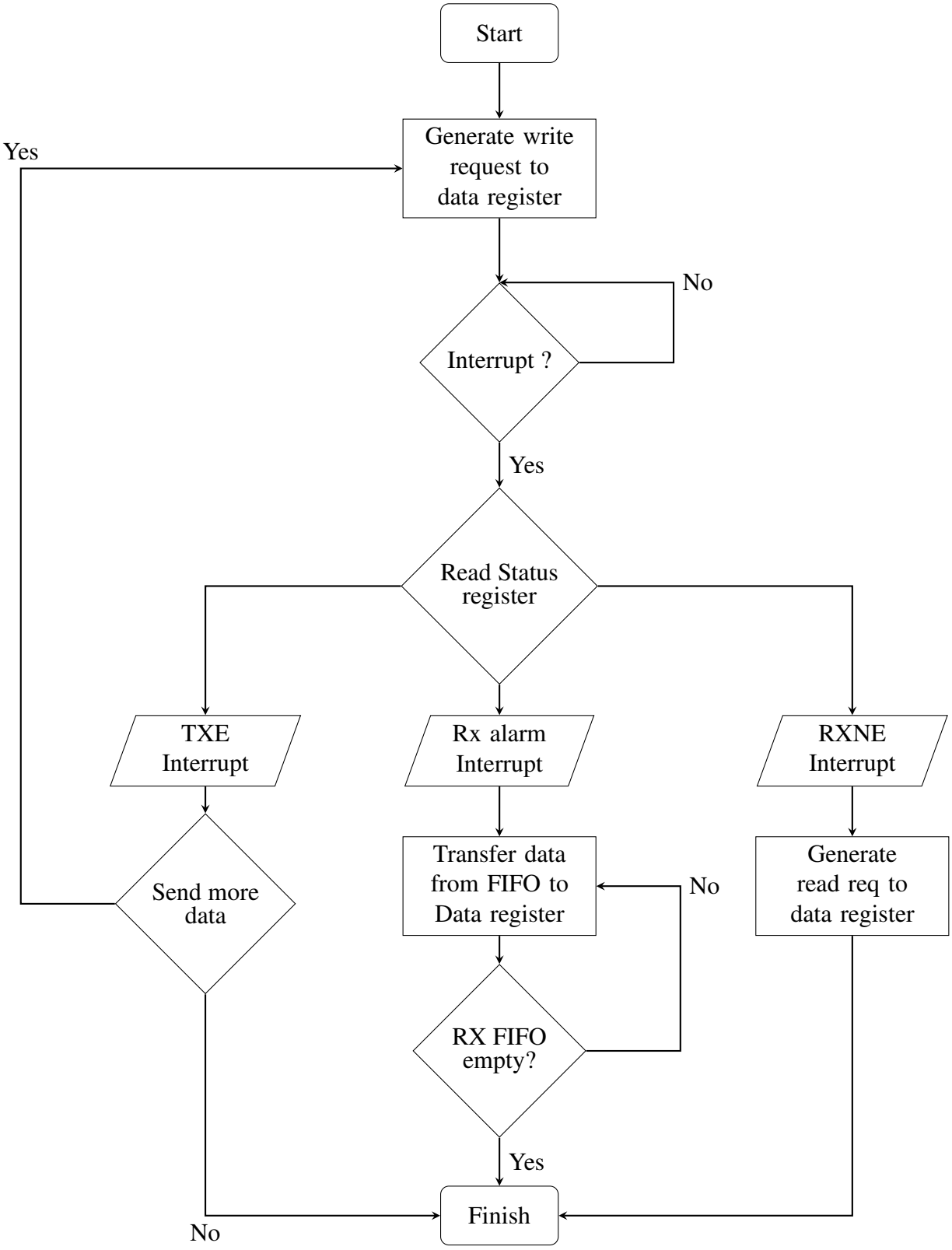

Figure 2.7: Interrupt mode

CHAPTER 3

SPI REGISTERS AND CONFIGURATION

The peripheral registers can be accessed by words (32-bit).

3.1 SPI control register 1

Address offset: 0x00

Reset value: 0x00000000

31 - 24	23 - 16	15	14	13	12	11	10
TOTAL BITS_RX	TOTAL BITS_TX	BIDI MODE	BIDI OE	CRC EN	CRC NEXT	CRCL	Rx ONLY
rw	rw	rw	rw	rw	rw	rw	rw

9	8	7	6	5 - 3	2	1	0
SSM	SSI	LSB FIRST	SPE	BR [2:0]	MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw

Table 3.1: SPI CR1

Bits [31:24]	TOTAL BITS_RX	Expected Total number of bits to be received. Receive state will go to idle after sending this many bits. One can receive maximum 1280 bits. (Reading status register, Device ID and data, etc).
Bits [32:16]	TOTAL BITS_TX	Total number of bits to transmitted. After transmitting this many bits it Transmit state will got to idle. One can send maximum 1280 bits. It include the instruction, read/write address and write data.

Bit 15	BIDI MODE	Bidirectional data mode enable. This bit enables half-duplex communication using common single bidirectional data line. 0: 2-line unidirectional data mode selected 1: 1-line unidirectional data mode selected
Bit 14	BIDI OE	Output enable in bidirectional mode This bit combined with the BIDI-MODE bit selects the direction of transfer in bi-direction mode. 0: receive-only mode (Output Disabled) 1: transmit-only mode (Output Enabled)
Bit 13	CRC EN	Hardware CRC calculation Enable. 0: CRC calculation disable 1: CRC calculation enable
Bit 12	CRC NEXT	Transmit CRC Next. 0: Next Transmit value is from Tx buffer 1: Next Transmit value is from Rx buffer
Bit 11	CRCL	CRC Length bit is set and cleared by software to select the CRC length.
Bit 10	RX ONLY	Receive only mode enabled. This bit enables simplex communication using a single unidirectional line to receive data exclusively. Keep BIDIMODE bit clear when receiving the only mode is active. This bit is also useful in a multi slave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.
Bit 9	SSM	Software Slave Management. When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit. 0: Software slave management disabled 1: Software slave management enabled

Bit 8	SSI	Internal Slave Select.This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored
Bit 7	LSB FIRST	Frame Format. 0: data is transmitted/received with the MSB first 1: data is transmitted/received with the LSB first *Note* This bit should not be changed when communication is ongoing
Bit 6	SPE	SPI Enables 0: Peripheral disabled 1: Peripheral Enabled
Bits [5:3]	BR [2:0]	Baud Rate Control. 000: fCLK/2 001: fCLK/4 010: fCLK/8 011: fCLK/16 100: fCLK/32 101: fCLK/64 110: fCLK/128 111: fCLK/256 *Note* This bit should not be changed when communication is ongoing
Bit 2	MSTR	Master selection 0: Slave Configuration 1: Master Configuration *Note* This bit should not be changed when communication is ongoing
Bit 1	CPOL	Clock polarity 0: CLK is 0 when idle

1: CLK is 1 when idle

Note This bit should not be changed when communication is ongoing

Bit 0 CPHA Clock Phase

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Note This bit should not be changed when communication is ongoing

3.2 SPI Control Register 2

Address offset: 0x04

Reset value: 0x00007000

16	15	14	13	12	11	10	9
RX IMM START	RX START	FRXTH	TX FIFO FULL	-	-	TX FIFO EMPT	RX FIFO FULL
rw	rw	rw	rw	rw	rw	rw	rw

8	7	6	5	4	3	2	1	0
-	-	RX FIFO EMPT	ERR IE	TX ALARM IE	RX ALARM IE	SSOE	TXE IE	RXNE IE
rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 3.3: SPI CR2

Bits [31:17] Reserved Must be kept at reset value

Bit 16 RX_IMMA START Set this bit to receive data immediately after transmission finished. This bit should be set before configuring the control register 1.

0: Immediate Receive data disabled

1: Immediate Receive data enabled

Bit 15	RX START	Set this bit to start receive data only. This is similar to RX_ONLY bit in control register 1. 0: Receive only disabled 1: Receive only enabled
Bit 14	FRXTH	FIFO reception threshold bit is used to set the threshold of the RXFIFO that triggers an RXNE event. 0: RXNE event is generated if the FIFO level is greater than or equal to 1/2 (16-bit) 1: RXNE event is generated if the FIFO level is greater than or equal to 1/4 (8-bit)
BIt 13	TX FIFO FULL	Tx buffer full interrupt enable. 0: TX buffer full interrupt masked 1: TX buffer full interrupt not masked.
Bit 10	TX FIFO EMPT	TX buffer empty interrupt enable. 0: TX buffer empty interrupt masked 1: TX buffer empty interrupt not masked.
Bit 9	RX FIFO FULL	RX buffer full interrupt enable. 0: RX buffer full interrupt masked 1: RX buffer full interrupt not masked.
Bit 6	RX FIFO EMPT	RX buffer empty interrupt enable. 0: RX buffer empty interrupt masked 1: RX buffer empty interrupt not masked.
Bit 5	ERR IE	Error interrupt enable bit controls the generation of interrupt when an error condition occurs. 0: Error interrupt masked 1: Error interrupt not masked.

Bit 4	TX ALARM IE	TX alarm interrupt enable 0: TX ALARM interrupt masked. 1: TX ALARM interrupt not masked. Used to generate an interrupt request when the TX_ALARM flag is set.
Bit 3	RX ALARM IE	RX alarm interrupt enable 0: RX ALARM interrupt masked. 1: RX ALARM interrupt not masked. Used to generate an interrupt request when the RX_ALARM flag is set.
Bit 2	SSOE	SS output enable 0: SS output is disabled in master mode and the SPI interface can work in a multi-master configuration 1: SS output is enabled in master mode and when the SPI interface is enabled. The SPI interface cannot work in a multi-master environment.
Bit 1	TXE_IE	Interrupt enable for TXE event. 0: TXE interrupt masked 1: TXE interrupt is not interrupt masked
Bit 0	RXNE_IE	Interrupt enable for RXNE event 0: RXNE interrupt masked 1: RXNE interrupt is not interrupt masked

3.3 SPI Status Register

Address offset: 0x08

Reset value: 0x00000002

12-11	10 - 9	8	7	6	5	4	3	2	1	0
FT LVL	FR LVL	FRE	BSY	TX OVR	RX OVR	MOD F	TX ALRM	RX ALRM	TXE	RXNE
r	r	r	r	r	r	r	r	r	r	r

Table 3.5: SPI SR

Bits [31:13]	RSVD	Reserved, must be kept at reset value.
Bits [11:12]	FTLVL [1:0]	FIFO Transmission Level. These bits are set and cleared by hardware. 00: FIFO empty 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full
Bits [9:10]	FRLVL [1:0]	FIFO reception level.These bits are set and cleared by hardware. 00: FIFO empty 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full
Bit 8	FRE	Frame format error. This flag is used for SPI in the TI slave mode. Refer to Section 38.4.11: SPI error flags. This flag is set by hardware and reset when SPIx SR is read by software. 0: No frame format error 1: A frame format error occurred

Bit 7	BSY	The BSY flag is set and cleared by hardware (writing to this flag has no effect). When BSY is set, it indicates that a data transfer is in progress on the SPI (the SPI bus is busy). 0: SPI not busy 1: SPI is busy in communication or Tx buffer is not empty
Bit 6	TX OVR	An overrun condition occurs when a master or slave transmits data, and the TXFIFO does not have enough space to store this data. 0: No overrun occurred 1: Overrun occurred
Bit 5	RX OVR	An overrun condition occurs when data is received by a master or slave and the RXFIFO has not enough space to store this received data. 0: No overrun occurred 1: Overrun occurred When an overrun condition occurs, the newly received value does not overwrite the previous one in the RXFIFO. The newly received value is discarded and all data transmitted subsequently is lost
Bit 4	MODF	Mode fault occurs when the master device has its internal NSS signal (NSS pin in NSS hardware mode, or SSI bit in NSS software mode) pulled low. 0: No mode fault occurred 1: Mode fault occurred
Bit 3	TX ALARM	When TX buffer reaches certain threshold, TX ALARM bit is set. FIFO threshold is user programmable. 0: TX buffer has not reached the alarm level 1: TX buffer reached the alarm threshold

Bit 3	RX ALARM	When RX buffer reaches certain threshold, RX ALARM bit is set. FIFO threshold is user programmable. 0: RX buffer has not reached the alarm level 1: RX buffer reached the alarm threshold
Bit 1	TXE	This flag is set when transmission TXFIFO has send all the data.
Bit 0	RXNE	The RXNE flag is set when RX buffer finishes transferring data to register

3.4 SPI Data Registers

There are total five data registers. All the write request and read request are done by the data register. When user want to send data on MOSI line, write request to these data register must be generated. When user want read data from MISO pin, read request must be generated to these register.

Data Registers	Address offset
SPIx_DR1	0x0C
SPIx_DR2	0x10
SPIx_DR3	0x14
SPIx_DR4	0x18
SPIx_DR5	0x1C

Table 3.7: SPI Data register list

3.5 Configuration of SPI Registers

The configuration procedure is almost the same for master and slave. For specific mode setups, follow the dedicated sections. When a standard communication is to be initialized, perform these steps:

1. Write to the SPIx_CR1 register:

    - Configure the serial clock baud rate using the BR[2:0] bits.
    - Configure the CPOL and CPHA bits combination to define one of the four relationships between the data transfer and the serial clock
    - Configure the LSB_FIRST bit to define the frame format
    - Configure SSM and SSI
    - Configure the MSTR bit
    - Select the number of bits to transmit and receive using TOTAL_BITS_TX and TOTAL_BITS_RX.

2. Write to SPI_CR2 register:

    - Select simplex or half-duplex mode by configuring RX_IMM_START and RX_ONLY.
    - For interrupt mode select the required interrupt enable.

3. Write data to DR register
    - Generate write request with proper data on DR registers.
    - After communication is finished, one can read data for DR register.

CHAPTER 4

SPI DRIVER AND VALIDATION

4.1 SPI Flash

Serial Flash which are used for validation are S2FL128L by cypress and Micron N25Q128A. Both 3 Volt, 128Mb serial flash. Main difference between them is Micron only support 24 bit addressing mode while Cypress flash support both 24 bit and 32 bit addressing mode. Programming the flash can take as long as four to five minutes, which is mostly due to the lengthy erase process inherent to the memory technology. Once written, however, FPGA configuration can be very fast—less than a second. The other pins are in pulled high.

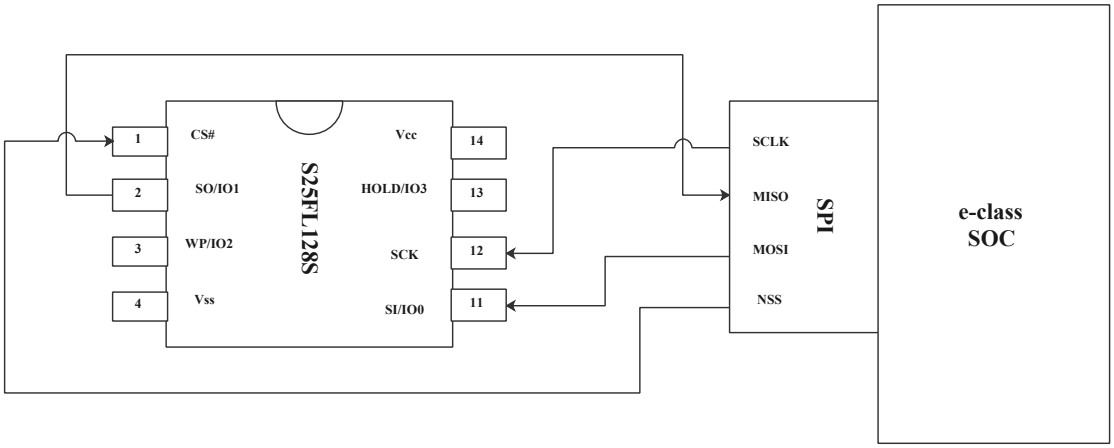

Figure 4.1: Flash pin diagram and connection scheme

4.1.1 Flash driver

Status Register in Flash

Registers are small groups of memory cells used to configure how the S25FL128S memory device operates or to report the status of device operations.

Bits	Field Name	Function	Default value
7	SRWD	Status Register Write Disable	0
6	P_ERR	Programming Error Occurred	0
5	E_ERR	Erase Error Occurred	0
4-2	BP[2:0]	Block Protection	1
1	WEL	Write Enable Latch	0
0	WIP	Write In Progress	0

Table 4.1: Status Register 1 (SR1)

- Write Enable Latch (WEL) SR1[1]: The WEL bit must be set to 1 to enable program, write, or erase operations as a means to provide protection against inadvertent changes to memory or register values. The Write Enable (WREN) command execution sets the Write Enable Latch to a 1 to allow any program, erase, or write commands to execute afterwards
- Write In Progress (WIP) SR1[0]: Indicates whether the device is performing a program, write, erase operation, or any other operation, during which a new operation command will be ignored. When the bit is set to a 1 the device is busy performing an operation.

Flash Programming Instruction Set

4.1.2 Command Sequence taken from S2FL128L Data-sheet

RDID (0x9F)

Provide read access to manufacturer identification, device identification

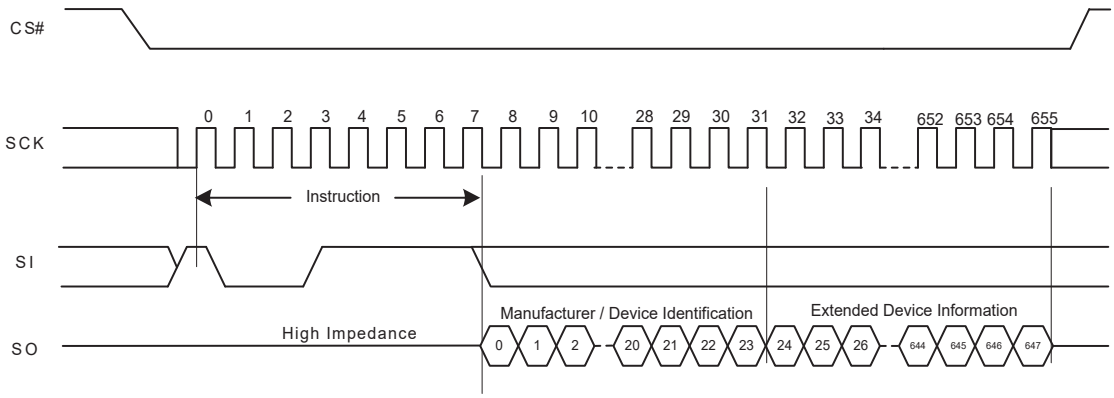

Figure 4.2: Read ID Command sequence

WREN (0x06)

Sets the Write Enable Latch (WEL) bit of the Status Register. Before using Page Program / Erase, WEL bit should be 1.

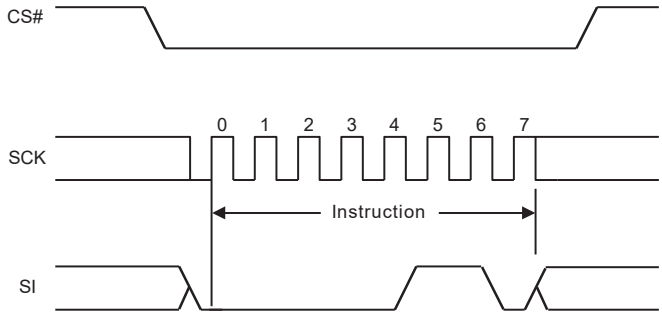

Figure 4.3: Write Enable Command sequence

RDSR (0x05)

Allows the Status Register contents to be read from SO.

By monitoring the WIP bit in SR1, user can check if flash is busy or not.

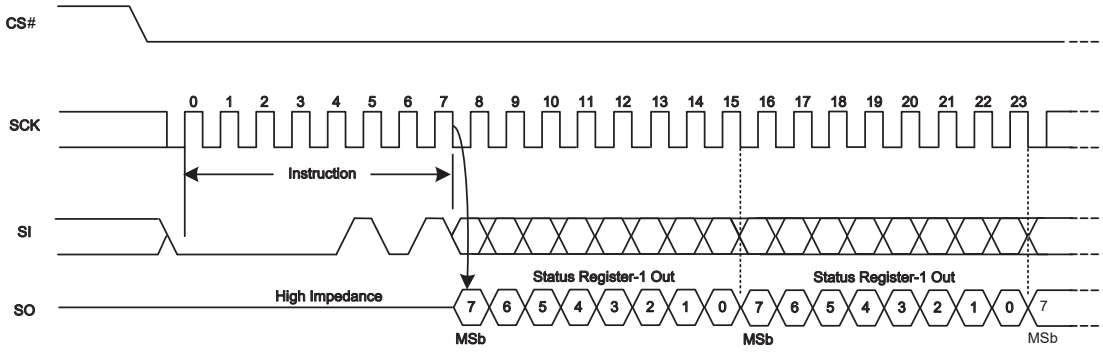

Figure 4.4: Read Status Register Command sequence

4SE (0xDC)

Sets all bits in the addressed sector to1 or Erase the memory array from selected address. The command is 4-byte addressable.

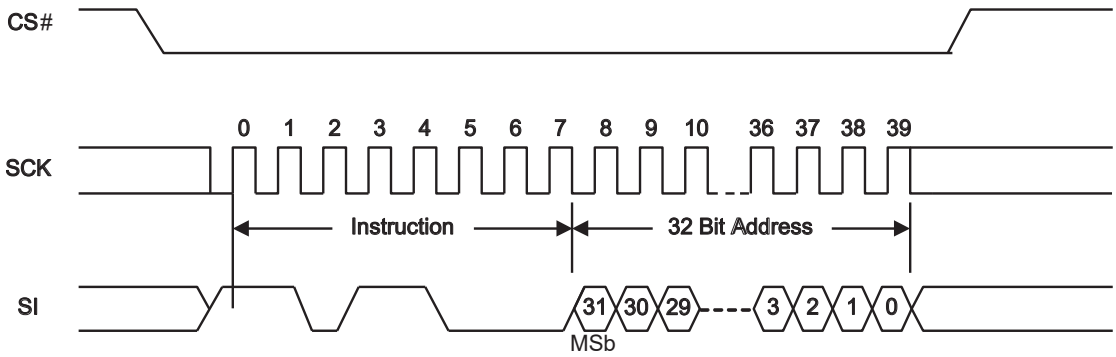

Figure 4.5: Sector erase Command sequence

PP (0x12)

Write data to memory array beginning at selected address.

The address is of 4-byte.

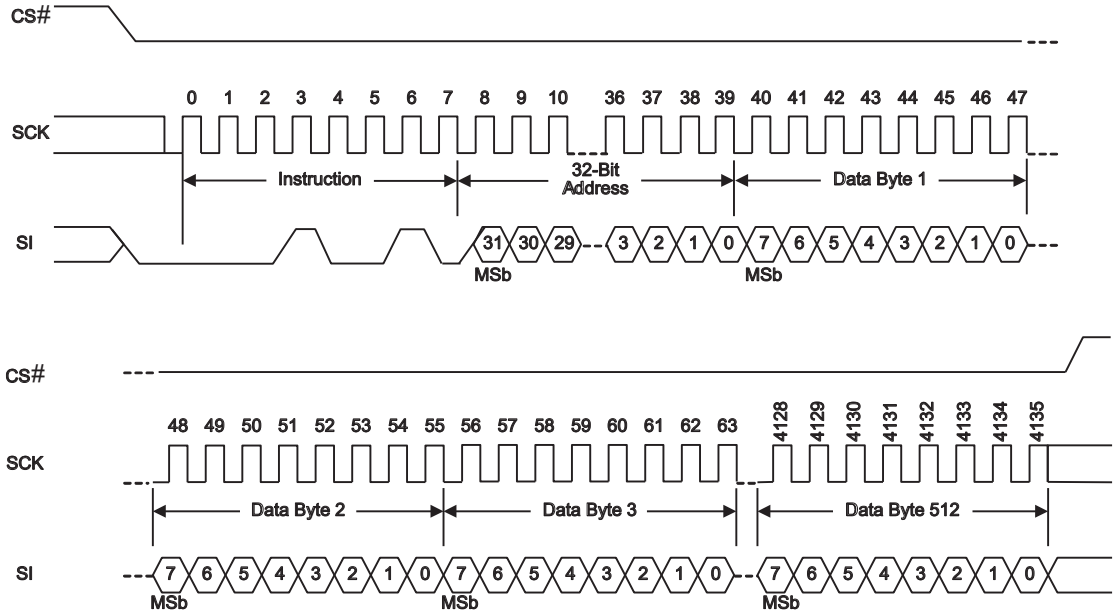

Figure 4.6: Page programming Command sequence

FAST_READ (0x0C)

Read data from memory array beginning at selected address. After 1-byte instruction, 4-byte address and 1-byte of dummy data must be given. The dummy bits should give flash enough time to prepare the data.

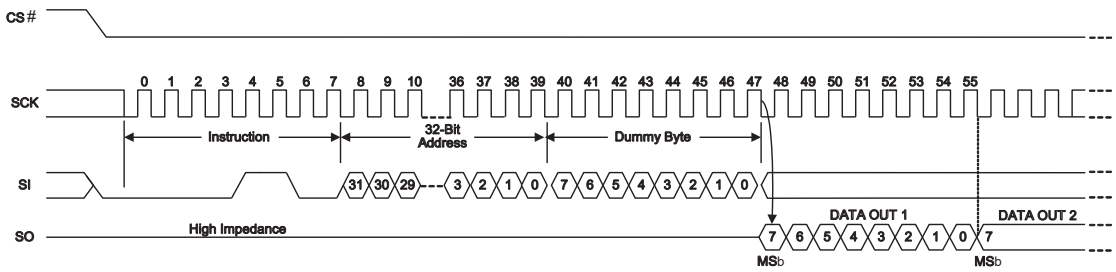

Figure 4.7: Read Command sequence

4.1.3 Flash driver Data-flow

Following flow chart explains the SPI driver for flash.

- SPI initialization setup the CPOL, CPHA and BR[2:0] bits in SPIx_CR1.
- Read ID, If id match go ahead with required operation else finish.
- Before Writing or Erase action, Write enable command must be given
- To check if erase or write still in progress, check WIP bit in SR of flash
- All Erase, Page Program and read are 4-byte addressable.
- If transmit only mode is selected then before sending next set of data check if SPI is busy or not using BSY bit in SPIx_SR register.
- Read id finished when required data is received and RXNE bit is set.
- When RXNE = 1, generate read request to data register.

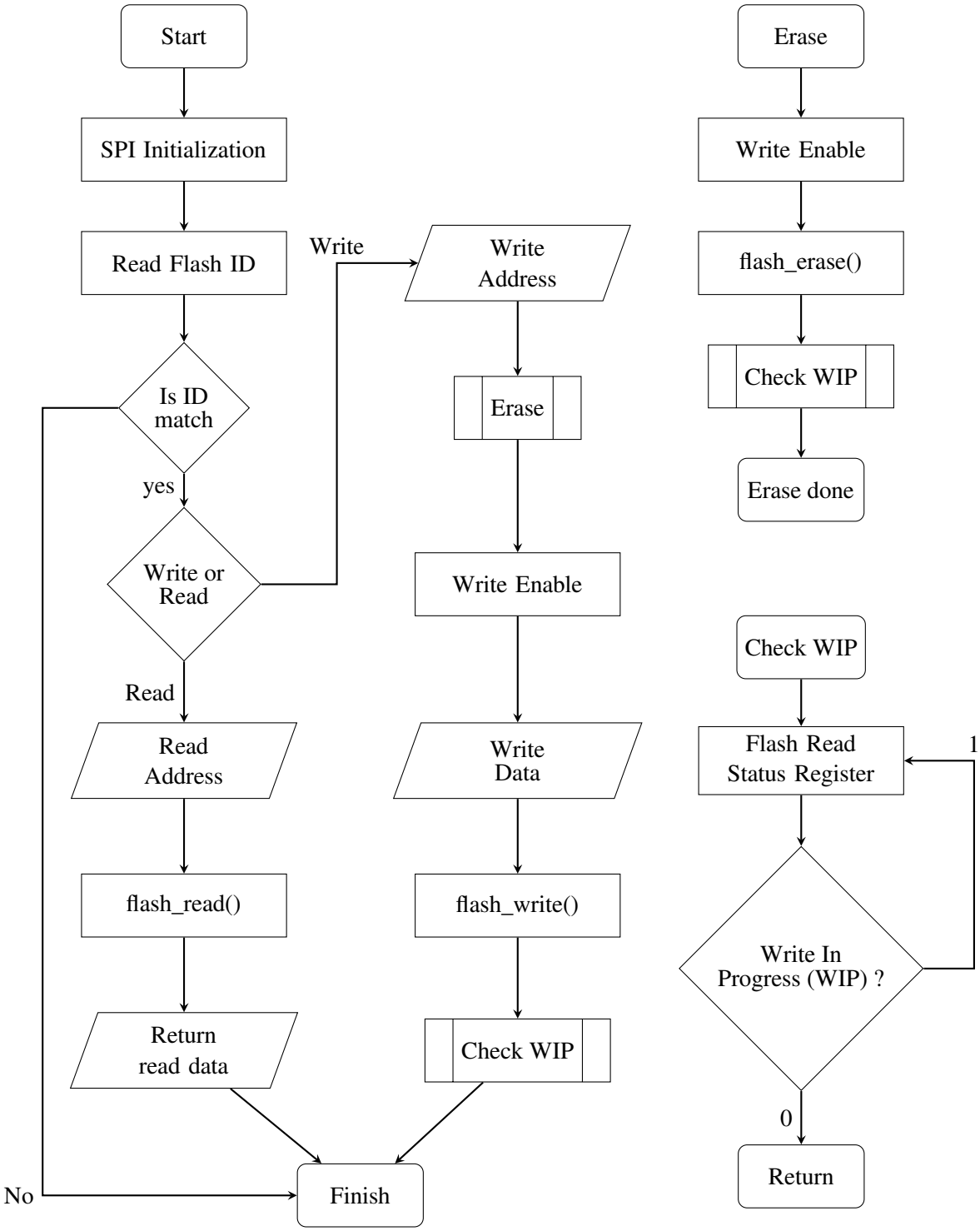

Figure 4.8: SPI driver for Flash

4.1.4 Result and conclusion

The E-Class Soc with SPI Peripheral imported to Arty-35T FPGA. The SPI is tested with CYPRESS and ISSI Flash. Arty-35T has built in 128Mb Cypress Flash.
The following waveform capture using Logic analyzer from SALEAE with ARTY-35T internal **Cypress** Flash.

Reading Manufacturing and Device ID

Total 24 bits which are received are 0x012018 which implied that

- 0x01 - Manufacturer ID for Cypress
- 0x20 - Device ID MSB (Memory Interface Type)
- 0x18 - Device ID LSB (Memory Density is 128Mb)

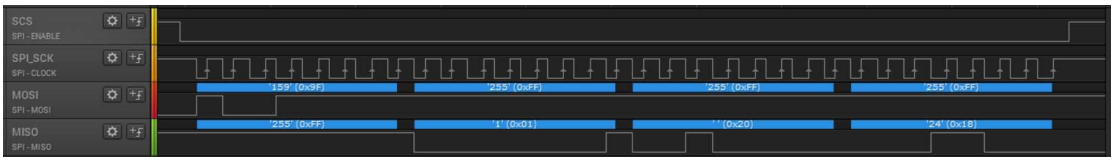

Figure 4.9: RDID Command with Cypress flash

Write Enable before Erase/Page program

Only 8 bit instruction as 0x06 is sent.

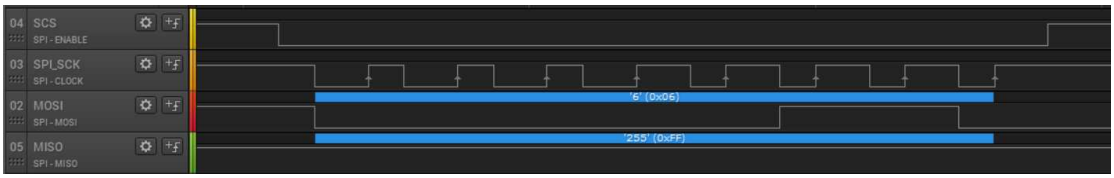

Figure 4.10: WREN Command with Cypress flash

Erase command

32 bit address is expected after 8-bit instruction. Starting Address = 0x00B00000 all bits set to 1.

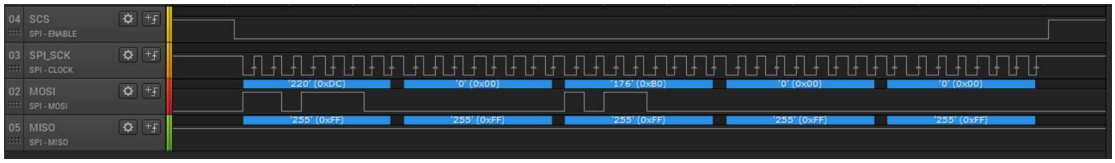

Figure 4.11: S4E Command with Cypress flash

Reading Status Register

After giving Erase or Page program command, the user should read the Status Register. If the WIP bit is set, representing an Erase/Page program is still in progress and not completed, the user must wait before giving the next command.

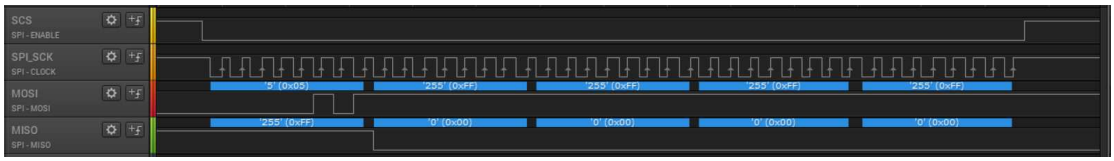

Figure 4.12: RDSR Command with Cypress flash

As the SR returns the value 0x00000000, which represents the WIP bit is reset to 0. Flash can accept the next command.

Page Program

Figure 4.13: PP Command with Cypress flash

After erasing, flash is ready for programming. Before giving the Page program command WEL bit should be set by using WREN command. The 32-bit address, followed by write data, should be transmitted after 8-bit write instruction.
Four consecutive writes were performed, starting with start address as 0x00B00000. Each PP command writes 32-bit of data into flash with an increasing address. Data which is written into the flash - 0x41014081, 0x42014181, 0x43014281, and 0x44014381.

Reading the Flash memory array

32-bit address and 8-bits dummy bits must be transmitted. The read address is 0x00B00000. After writing data into flash, four simultaneous reads are performed.As you can see, after 48 bits, transmission finished and receive started. MISO pin returns the data corresponds to a provided address. The receive data are 0x41014081, 0x42014181, 0x43014281, and 0x44014381.

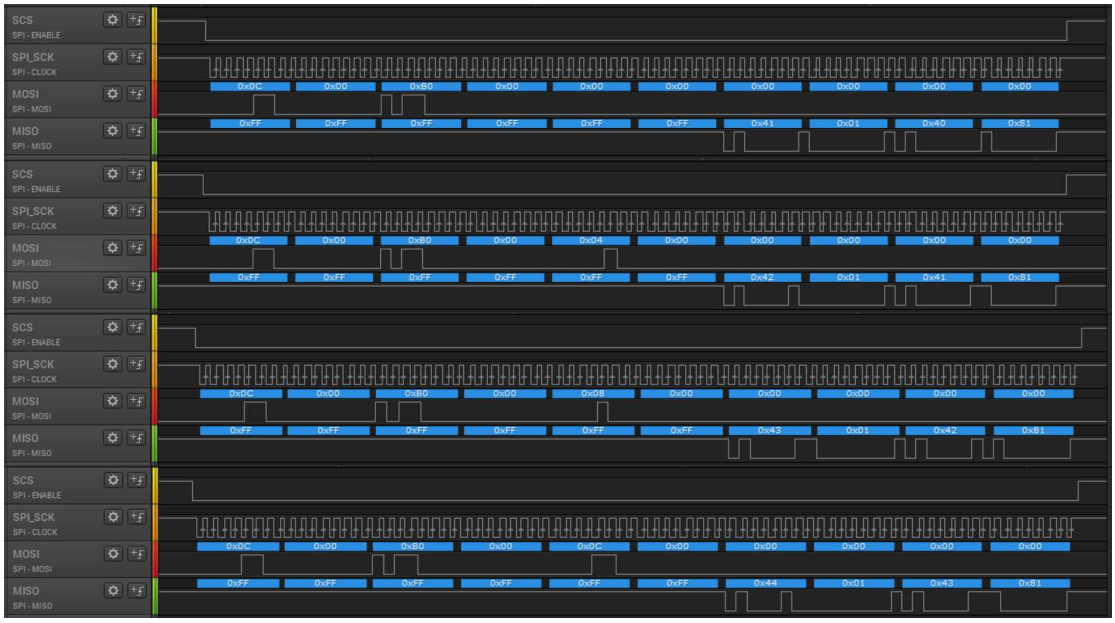

Figure 4.14: FAST_READ Command with Cypress flash

ISSI Flash

The next set of waveform is captured with **ISSI Flash (S25PL256D)** with help of Logic Analyzer from SALEAE.

Reading Manufacturing and Device ID

Total 24 bits which are received are 0x9D6019 which implied that

- 0x9D - Manufacturer ID for ISSI
- 0x6019 - Memory Type + Capacity for IS25LP256D

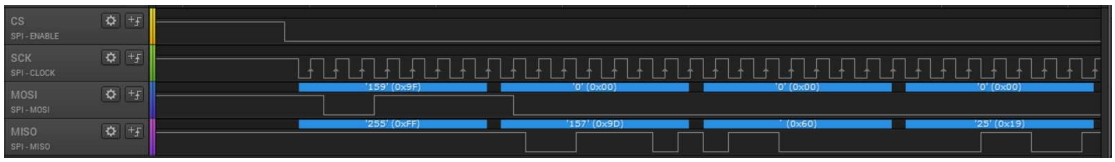

Figure 4.15: RDID Command with ISSI flash

Erase command

Here instead of 0xDC, ISSI used 0x21 for erase instruction.

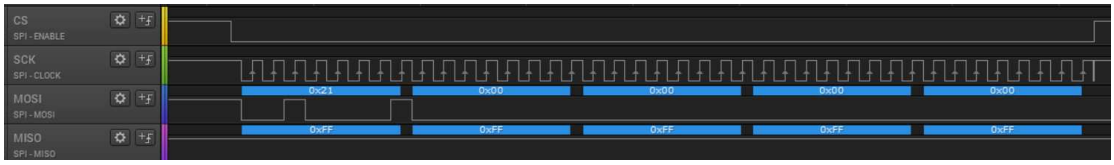

Figure 4.16: S4E Command with ISSI flash

Page Program

Here total 16-bytes of data are written in single page program instruction. Data 0x01010101, 0x02020202, 0x03030303and 0x04040404 from address 0x00000000 are written into SRAM.

Figure 4.17: PP Command with ISSI flash

Reading the Flash memory array

After writing data into flash, four simultaneous reads are performed starting from address 0x00000000. Read data 0x01010101, 0x02020202, 0x03030303and 0x04040404

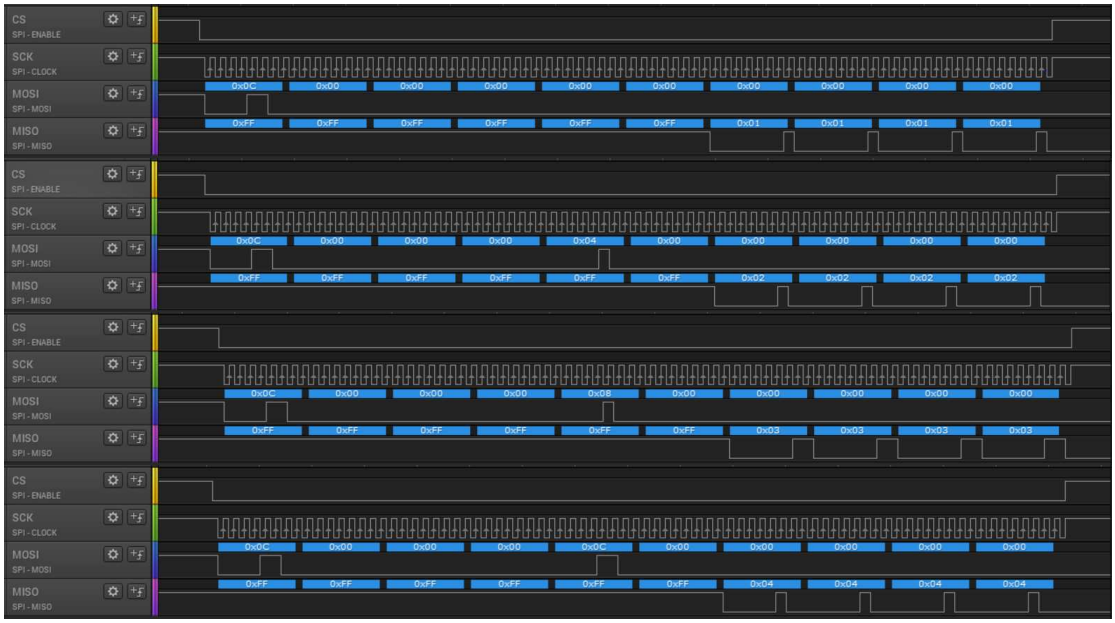

Figure 4.18: FAST_READ Command with ISSI flash

Conclusion

The SPI is working perfectly with Cypress and ISSI flash. All the operations such as reading ID, reading a status register, erasing the memory array, page programming and reading the memory are working correctly.

4.2 SPI Sram

The ISSI IS62/65WVS5128GALL/GBLL are 4Mb Fast Serial static RAMs organized as 512K bytes by 8 bits. The device is accessed via a simple Serial Peripheral Interface (SPI) compatible serial bus. The bus signals required are a clock input (SCK) plus separate data in (SI) and data out (SO) lines. Access the device is controlled through a Chip Select (CS).

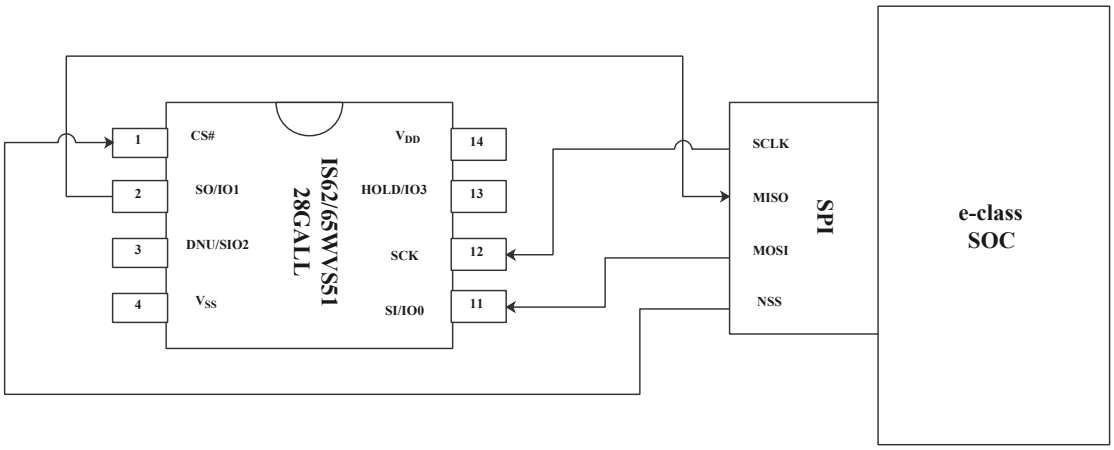

Figure 4.19: Connection Scheme and pin diagram of SRAM

4.2.1 SRAM driver

Operating Modes

The device has three modes of operation that are selected by setting bits 7 and 6 in the MODE register. The modes of operation are Byte, Page and Sequential. The possible

7	6	5	4	3	2	1	0
W / R		-	-	-	-	-	-
MODE		Reserved					
W / R = writable / readable							

Table 4.2: Bits in MODE register of SRAM

modes of operation are:

- 00 = Byte mode: the read/ write operations are limited to only one byte
- 10 = Page mode: the read and write operations are limited to within the addressed page (the address is automatically incremented internally)

- 01 = Sequential Mode (Default Mode): Sequential operation allows the entire array to be written to and read from. The internal address counter is automatically incremented until reaches the end of die boundary
- 11 = Reserved and Bits 0 through 5 are reserved and should always be set to '0'.

SRAM Programming Instruction Set

4.2.2 Command Sequence taken from IS62/65WVS512 Datasheet

WRMR (0x01)

The Write Mode Register instruction allows the user to write to the bits in the MODE register. This allows for setting of the Device operating mode.

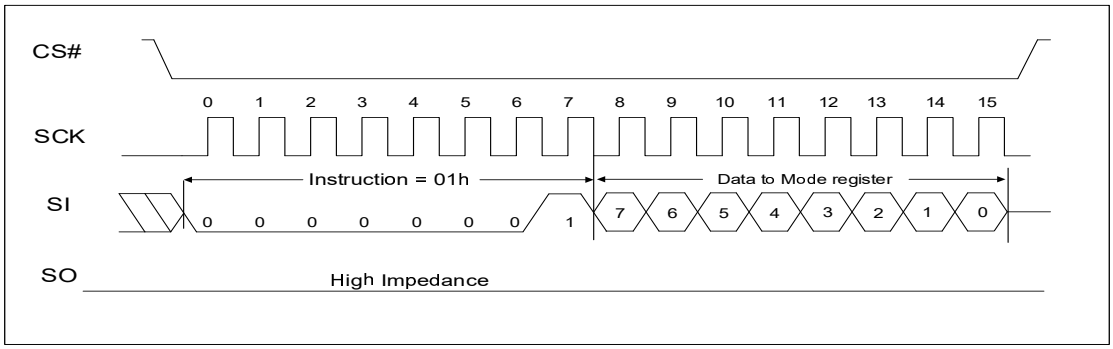

Figure 4.20: WRMR Command sequence

RDMR (0x05)

The Read Mode Register instruction provides access to the MODE register. The MODE register may be read at any time.

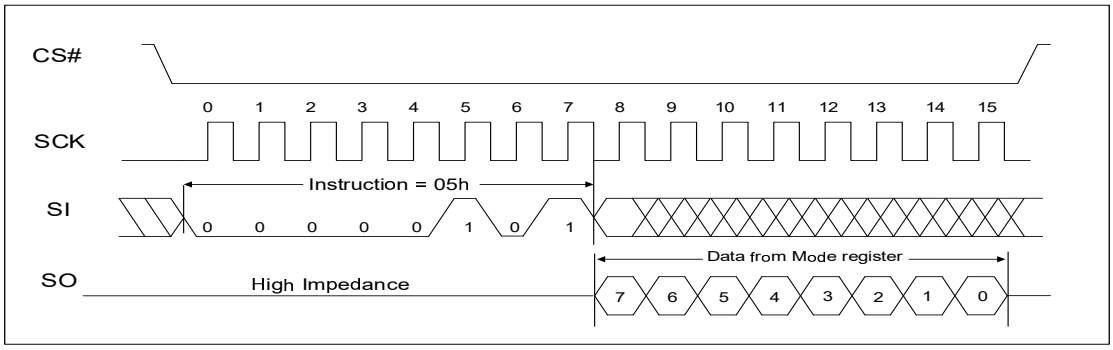

Figure 4.21: RDMR Command sequence

WRITE (0x02)

Write data to memory array beginning at selected address. It support 3-byte addressing mode only.

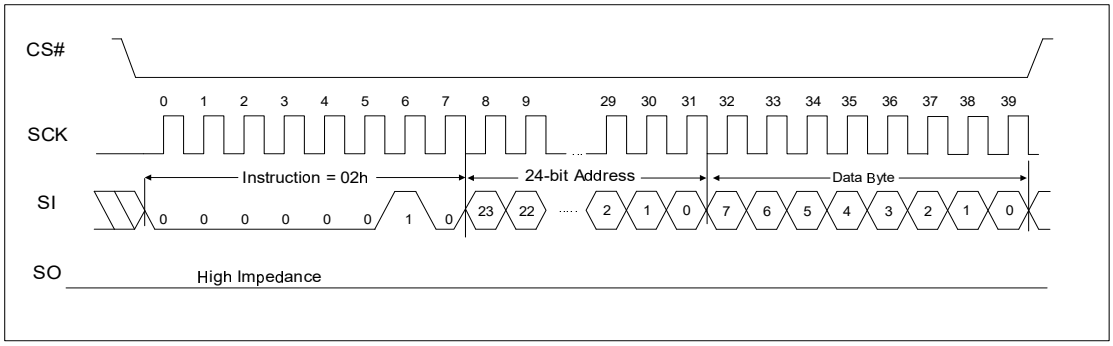

Figure 4.22: Byte WRITE Command sequence

READ (0x03)

Read data from memory array beginning at selected address. After 1-byte instruction, 24-bit address and 8-bits of dummy data must be given. The dummy bits should give sram enough time to prepare the data.

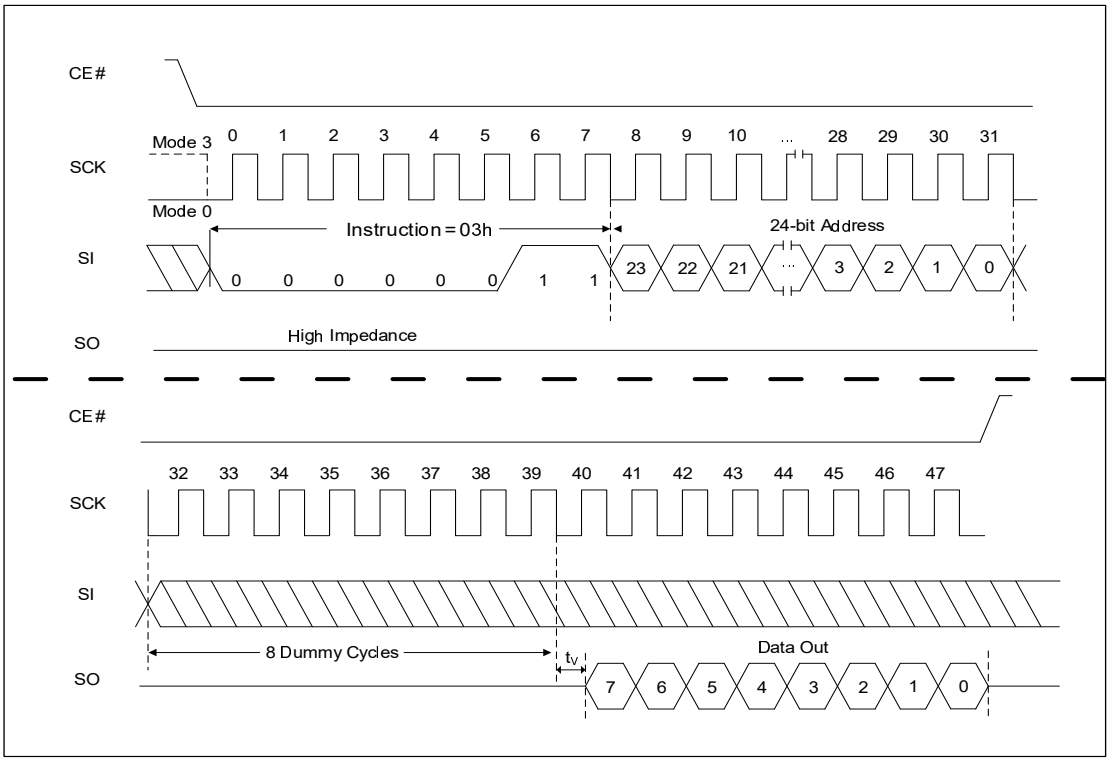

Figure 4.23: Byte READ Command sequence

4.2.3 SRAM driver flow diagram

Following flow chart explains the SPI driver for SRAM.

1.
2. SPI initialization setup the CPOL, CPHA and BR[2:0] bits in SPIx_CR1.
3. Write request are generated on Data register with required data.
4. SPI start when SPIE bit is set in SPIx_CR1.
5. Select the operating mode between byte mode, page mode and sequential mode(default) using WDMR command.
6. Generate the read request to data register when RXNE = 1 (received finish).

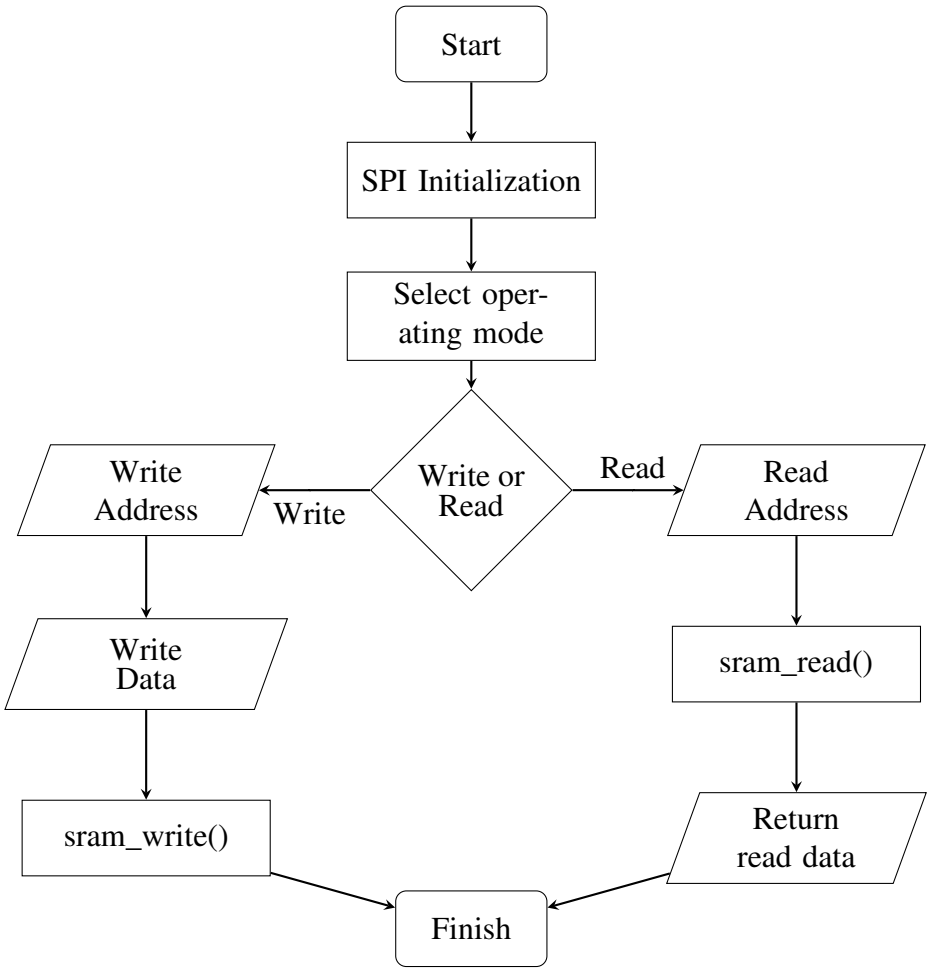

Figure 4.24: SPI driver for SRAM

4.2.4 Simulation Result

The E-Class Soc with SPI Peripheral is simulated with help of NCVerilog. The SPI is tested with ISSI SRAM bfm.
The following waveforms are capture using GTKwave.

Write Mode Register Instruction

The operating mode can be selected by changing 7 and 6 bit. Page mode is selected by setting [7:6] as 2'b10.

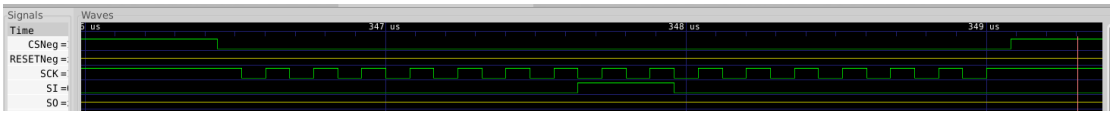

Figure 4.25: WREN Command with Cypress flash

Read Mode Register Instruction

Total 8 bits which are received. As [7:6] bit are 10 which implied that SRAM is operating in page mode.

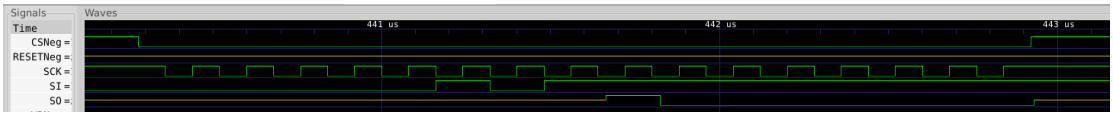

Figure 4.26: RDMR command

Page Write

The 24-bit address, followed by write data, should be transmitted after 8-bit instruction. Two consecutive writes were performed, starting with start address as 0x000100. Each Write command writes 32-bit of data into SRAM with an increasing address. Data which is written into the SRAM - 0xAA9955AA, 0x54FEAC52.

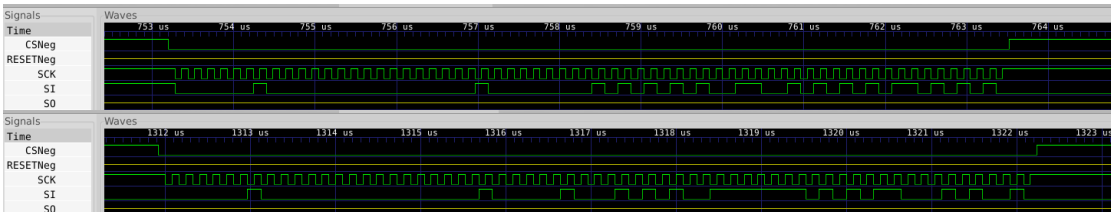

Figure 4.27: Writing DATA into SRAM

Reading the SRAM memory array

24-bit address and 8-bits dummy bits must be transmitted. The read address is 0x000100. As you can see, after 40 bits, transmission finished and receive started. MISO pin returns the data corresponds to a provided address. The receive data are 0xAA9955AA and 0X54FEAC52.

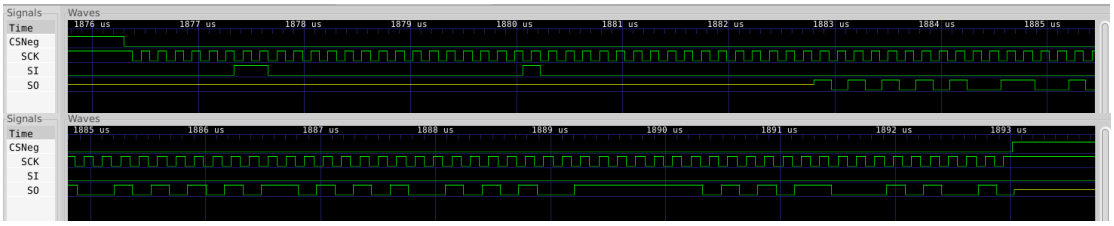

Figure 4.28: Reading data from SRAM

The simulation log is attached for reference.

```
SPI SRAM START
SPI init done
Sram Mode: Page Mode
write data done: aa9955aa
write data done: 54feac52
Reading Value aa
Reading Value 99
Reading Value 55
Reading Value aa
Reading Value 54
Reading Value fe
Reading Value ac
Reading Value 52
sram read done 1:
```

Figure 4.29: Simulation Log os SRAM

Conclusion

The Read and Write is done successfully on SRAM using SPI controller.

4.3 Analog to Digital Converter

The ADC, which is used for SPI validation, is MCP3302. It is 13-bit A/D converter features fully differential inputs. It is user-programmable to provide two differential input pairs or four single-ended inputs. It has four channels. The temperature sensor (LM35) is connected to CH0, and the potentiometer is connected to CH3 of the MCP3202.

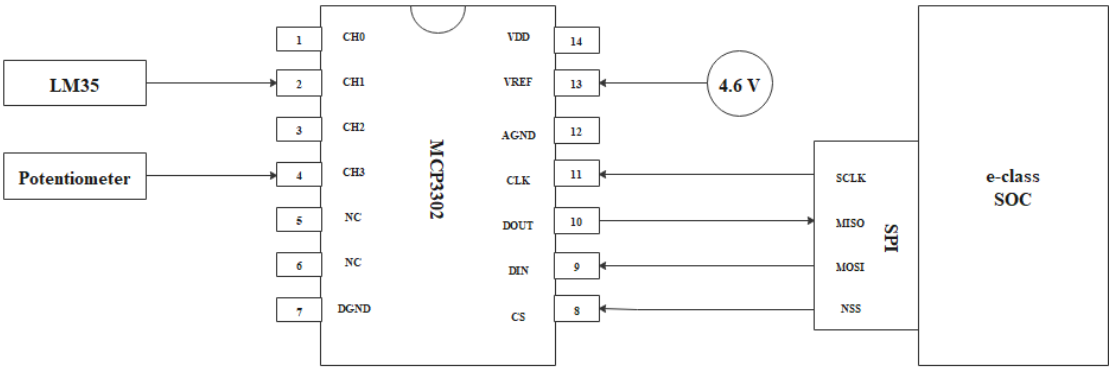

Figure 4.30: ADC pin diagram and connection scheme

.

ADC MCP 3302

The constant voltage source of 4.6v is connected to the Vref port. This input pin provides the reference voltage for the device, which determines the maximum range of the analog input signal and the LSB size. The LSB size is determined according to the equation shown below.

$$LSBSize = \frac{2 \times V_{REF}}{8192}$$

We get *LSBSize* as 0.001123.

LM35 Temperature sensor

It is an integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature.

LM35 Transfer Function:

$$V_{OUT} = 10mv\ ^0C \times T$$

4.3.1 ADC driver

SPI Communication with MCP3302

The ADC takes 4 bit as input and return the 13 bit output on SPI lines. D2 bit is dont care for MCP3302. The four bits can configure as shown below:

Configuration bits selection				Input configuration	Channel selection
Single/ D̃iff	D2	D1	D0		
1	X	0	0	Single ended	CH0
1	X	0	1	Single ended	CH1
1	X	1	0	Single ended	CH2
1	X	1	1	Single ended	CH3
0	X	0	0	Differential	CH0=IN+, CH1=IN-
0	X	0	1	Differential	CH0=IN-, CH1=IN+
0	X	1	0	Differential	CH2=IN+, CH3=IN-
0	X	1	1	Differential	CH2=IN-, CH3=IN+

Table 4.3: Cofiguration bits for MCP3302

MCP3302 is a 12 bit analog to digital converter. Out of received 13 bits in MSB_FIRST format, MSB is discarded, and later 12 bits are taken as output. The output is multiplied by *LSBSize* to get output voltage. The following figure is taken from the MCP3302 datasheet, which explains the spi communicating sequence with ADC. .

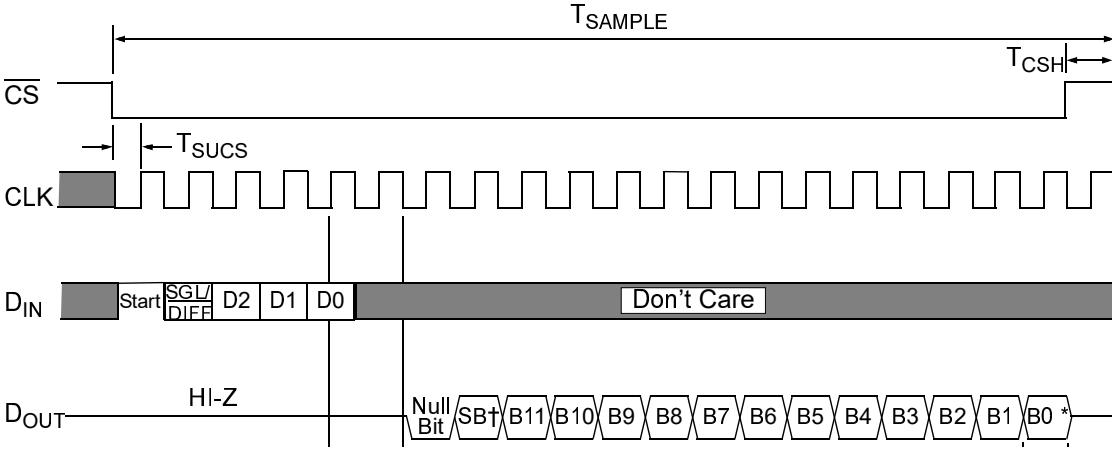

Figure 4.31: Communicating sequence with ADC

4.3.2 ADC driver flow diagram

Following flow chart explains the SPI driver for ADC.

1. SPI initialization setup the CPOL, CPHA and BR[2:0] bits in SPIx_CR1.
2. Select the operating mode and Channel using configuration bits.
3. Write request are generated on Data register with required data.
4. SPI start when SPIE bit is set in SPIx_CR1.
5. SPI generate read request to data register when transaction is finished.

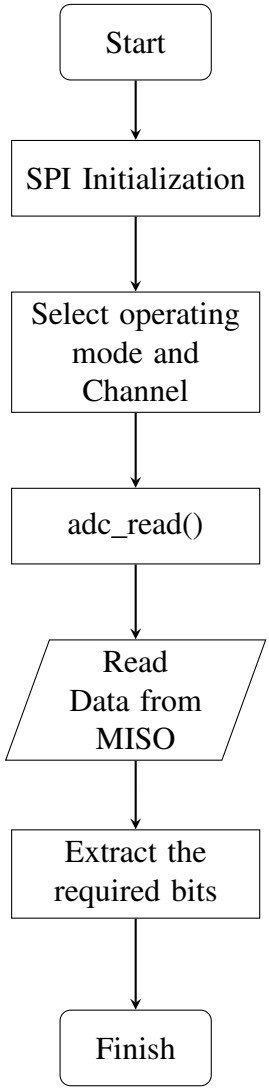

Figure 4.32: SPI driver for ADC

4.3.3 Result and conclusion

Several readings were taken By varying the input voltage using a potentiometer and temperature near the sensor.

Temperature Sensor LM35 on CH1 of MCP3302

Following reading is taken when room temperature is around 30-32 ° C. Using SPI in the loop, users can continuously monitor the room temperature.
As 4'b1101 is sent to ADC, CH1 is selected and 13 bits are receive which are 0x011B. If MSB bit is neglected we get 12 bits 0x11B ('283'). When multiplied by *LSBSize = 0.001123* ,we get 0.317822, which is equivalent of 31.7 ° C.

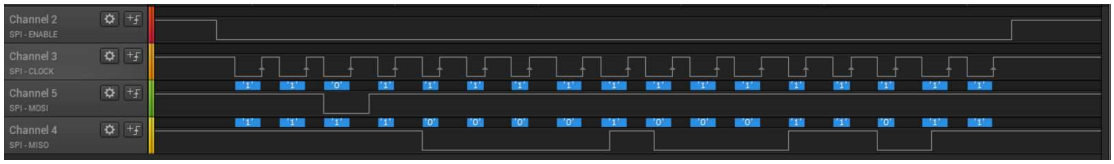

Figure 4.33: SPI with ADC and LM35 in mode3

In the next set of readings, a bag of cold water is placed near a sensor to lower its temperature.
Here the received bits in first case are 0x00E9 and second case are 0x0075. If MSB bit is neglected we get 12 bits 0x0E9 ('233') and 0x075 ('113'). The corresponding voltages when multiplied by *LSBSize = 0.001123* are 0.261659 V and 0.131391 V. Using LM35 linear relation between voltage and temperature, we get 26.16 ° C and 13.13 ° C.

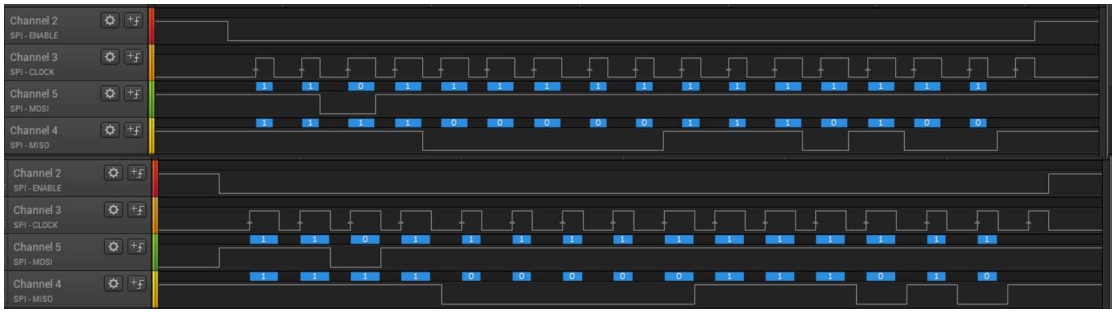

Figure 4.34: SPI with ADC and LM35 in mode 0

Temperature is decreasing with time, and SPI is successfully able to communicate it with core.

Potentiometer on CH3 of MCP3302

Following readings are taken by varying input voltage of ADC with the help of three terminal potentiometer.

Input voltage at CH3 as 2.9V

As 4'b1111 is sent to ADC, CH3 is selected and 13 bits are receive which are 0x0A7B. If MSB bit is neglected we get 12 bits 0xA7B ('2683'). When multiplied by *LSBSize = 0.001123* , we get 3.013 V.

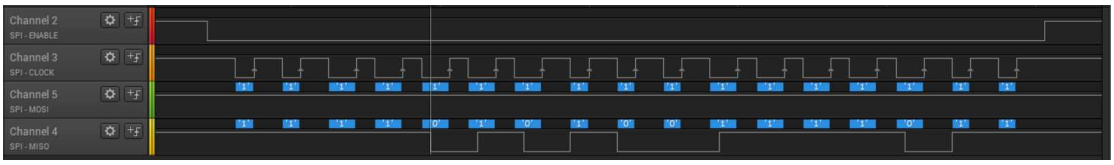

Figure 4.35: SPI with ADC with Vin as 2.9V

Input voltage at CH3 as 2.35 V

Here the received bits are 0x084A. If MSB bit is neglected we get 12 bits 0x84A ('2122'). When multiplied by *LSBSize = 0.001123* , we get 2.383 V.

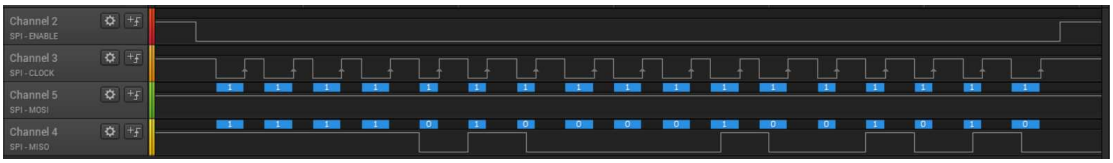

Figure 4.36: SPI with ADC with Vin as 2.35 V

Conclusion

SPI Controller is functioning correctly with Analog to digital converter.

REFERENCES

[1] Reference manual for STM32F030x4/x6/x8/xC and STM32F070x6/xB advanced ARM®-based 32-bit MCUs

[2] MPC5121e Serial Peripheral Interface (SPI) by NXP

[3] The Cypress S25FL128S and S25FL256S datasheet

[4] The ISSI IS62/65WVS512 datasheet

[5] The TI MCP3302 datasheet