SRAM for IoT Applications in 0.6μm CMOS

Subhra S. Mahapatra[1], Aditya Singh[1], Manamohan Nath[2], Prachi Mrudula[2], Senha Kumari[2], Santunu Sarangi[1], and Saroj Rout[1,*]

[1]Silicon Institute of Technology, Bhubaneswar, india
[2]Sevya Pvt. Ltd., Greater Noida, India
[*]Corresponding author: saroj.rout@silicon.ac.in

Abstract

This article presents the detailed design and implementation of a Static Random-Access Memory (SRAM) in CMOS technology. The data access (read/write) is done through Serial Peripheral Interface (SPI), an industry-standard serial protocol. This SRAM is specifically suitable for Internet-of-Things (IoT) applications with slow access rates and low power consumption. For the purpose of demonstration, a 32-byte SRAM was designed and fabricated in 0.6μm CMOS technology and successfully tested for its full functionality after fabrication.

1 Introduction

A typical IoT sensor contains four main devices as shown in Figure 1: sensor, microcontroller (MCU), a wireless device (Bluetooth, WiFi, LoRa) and SRAM. Since sensor nodes work on low data rates and low power, the devices communicate among themselves over a serial bus such as the Serial Peripheral Interface (SPI) or Inter IC Communication (I2C) protocol. The MCU serves as the master controller for the IoT node to negotiate all transactions between devices. The wireless device is responsible for sending sensor data to the nearest gateway and receiving control instructions from the gateway as well. The sensor measures environmental parameters (temperature/humidity/etc) and the data is digitized is transmitted over the SPI/ I2C serial communication bus. The SRAM is used to store local data when a gateway is not available for communication. This article describes the design and implementation of a low-power SPI accessible SRAM implemented in a 0.6µm CMOS technology. The innovation of this project lies in creating and controlling all the signals required for the SRAM without use of internal clocks, deriving all control signals from the SPI signals themselves instead, thereby making it a simple and scalable circuit that can be made to consume very little power with slower access time.

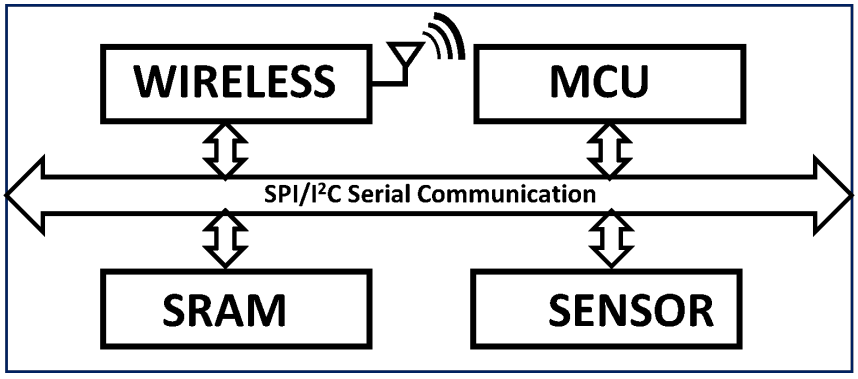

Figure 1: Block diagram of a typical IoT sensor node.

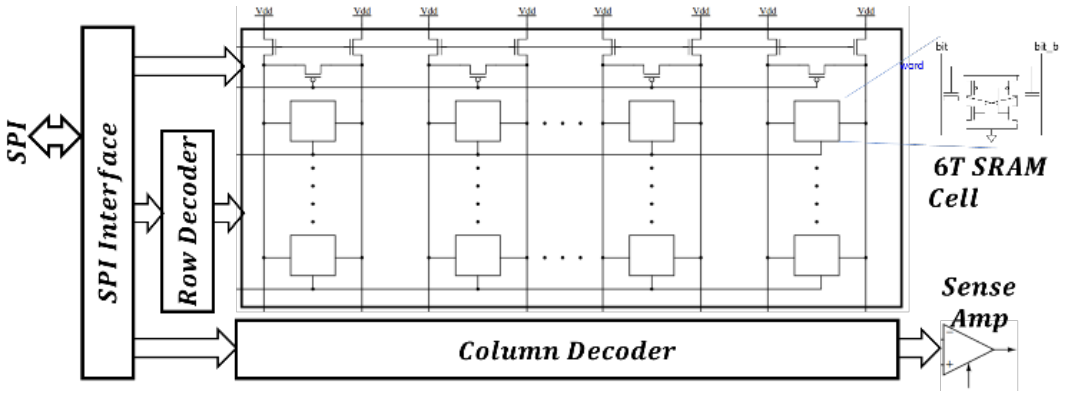

Figure 2: SRAM Architecture.

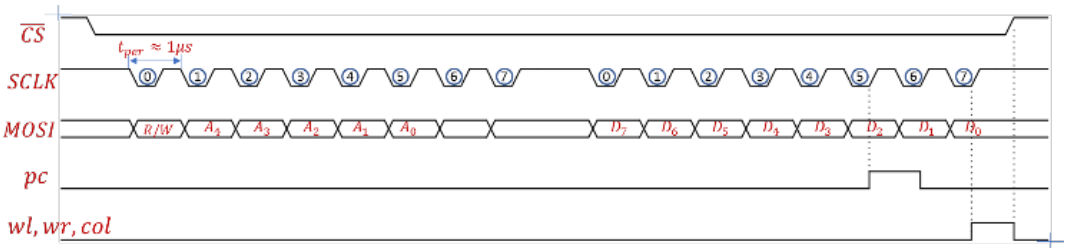

Figure 3: .Timing diagram for the write operation.

2 SRAM Architecture

The architecture of the SRAM with the serial interface is shown in Figure 2. The SRAM consists of the array of the core SRAM cells along with the row decoders, column decoders, and the sense amplifiers. The SPI interface converts the incoming serial signals to their appropriate control signals to read and write the data from the SRAM using the serial bus. The SRAM core consists of 32-bytes of 8-bit words laid out as a 16x16 array of six-transistor (6T) CMOS memory cells (Leblebici et al., 2014; Weste and Harris, 2011). Each cell is individually accessed by selecting the corresponding row and column using their respective decoders. The word lines are selected using the 4-bit LSB of the SRAM address $A_{3:0}$ and two columns are selected using the MSB of the address A_4. After selecting the individual cells, the corresponding bit lines are used to read from the cell or write to it. During the read operation the bit lines are first pre-charged to a pre-determined value $(V_{DD} - V_{TN})$, then the pre-charge circuit is disabled, and the SRAM cell is selected after that. The difference in the voltage between the bit lines represents the data in the SRAM cell and typically this differential voltage is a fraction of the supply voltage, not enough to drive a digital gate directly. A differential sense amplifier is used to amplify this small differential voltage to the supply rails (V_{DD} or 0) and then the output of the sense amplifier can drive digital gates directly. The output is fed back to the SPI controller which then converts the parallel 8-bit data to a serial output complying with the SPI protocol. During the write operation, the SPI controller converts the serial input data to 8-bit parallel data and the corresponding SRAM cells area selected, as in the read operation, but the column decoders now connect the bit lines to the 8-bit drivers to write the data instead of sensing it. Once the data is written, all the cells are de-selected and the data remain stored until the power is applied to the IC or overwritten by another write operation. The next two sections detail the architecture of the timing to keep everything synchronous to the master clock ($SLCK$). The SPI protocol has 4 modes of operation (0-3) (Leens, 2009) and this SPI controller is designed for Mode-2 where the idle clock is high and the masters sends data at the negative-edge of the clock and samples at the positive edge of the clock. To keep the design simple and avoid setup/hold time issues, the SRAM sends and samples data opposite to the master ie. negative-edge and positive-edge respectively.

2.1 Write Operation

Figure 3 shows the timing diagram of the SPI write operation. After the master (eg. MCU) selects the SRAM chip by pulling Chip-Select (CS) low, 2 bytes are transferred serially on the Master-Out-Slave-In

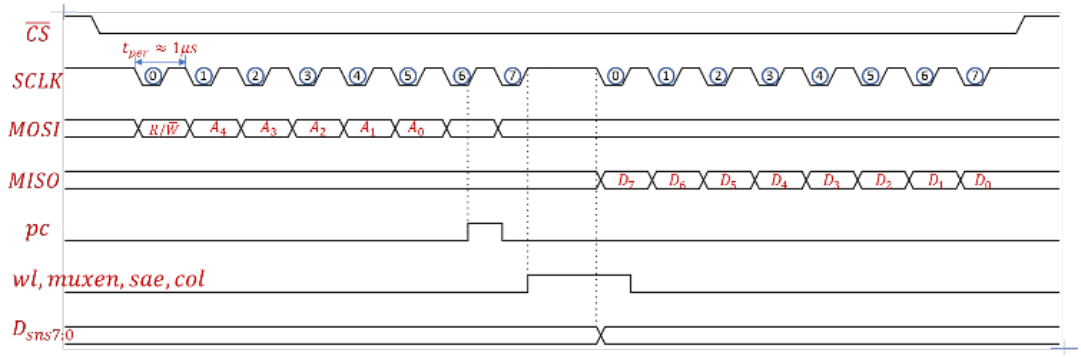

Figure 4: .Timing diagram for the read operation.

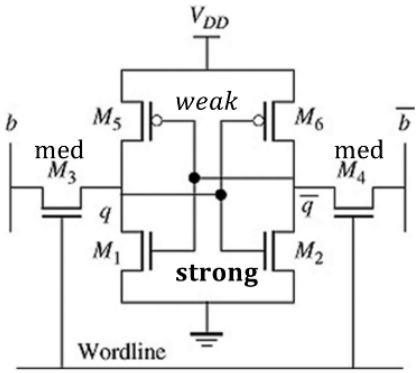

Figure 5: 6T SRAM Cell.

($MOSI$) port with data changing synchronously every negative edge of the clock input ($SCLK$). The first bit denotes if the operation is read (1) or write (0). The next 5 bits are assigned for the address of the SRAM and the last two bits are unused. The second byte contains the 8-bit data that needs to be written to the corresponding address. The controller latches the read/write bit after the first clock cycle and latches the address to the decoders at the end of the 6^{th} clock cycle. Then the controller generates a pre-charge signal (pc) at the positive edge of the 14^{th} cycle for one clock period. The data is latched for write at the positive edge of the last clock cycle (16^{th}) and write control signals (wl, wr, col) are also asserted at the same edge and pulled low when the chip is de-selected (CS is high).

2.2 Read Operation

Figure 4 shows the timing diagram for the read operation. First byte of the read operation is the same as the write operation except, the pre-charge signal (pc) is asserted at the positive edge of the 7^{th} clock cycle and the read control signals ($wl, col, muxen, sae$) are asserted from one clock cycle at the positive edge of the 8^{th} clock cycle during which the data from the SRAM ($D_{sns[7:0]}$) is latched by a shift-register at the negative edge of the 9^{th} clock cycle. Once the data is latched, the shift register shifts each bit into the Master-Input-Slave-Output ($MISO$) port at the negative edge of the clock for the next does clock cycles.

2.3 Six-Transistor (6T) SRAM Cell

The core of the design is the SRAM cell itself which stores one bit of information. The area and the power of each cell is very critical since it decides the area and power of the entire chip. In this design, a standard 6T SRAM Cell is used as shown in Figure 5 (Leblebici et al., 2014; Weste and Harris, 2011). In principle, it is a back-to-back inverter (M_1, M_2, M_5, M_6) to store data indefinitely if power is provided to the cell. The access transistors (M_3, M_4) are used to read and write data into the cell. The sizing of the devices is decided by three main factors: (1) smallest size (2) does not destroy the data when reading and (3) able to overwrite the stored data. On analyzing with the given constraints and the appropriate large-signal equations

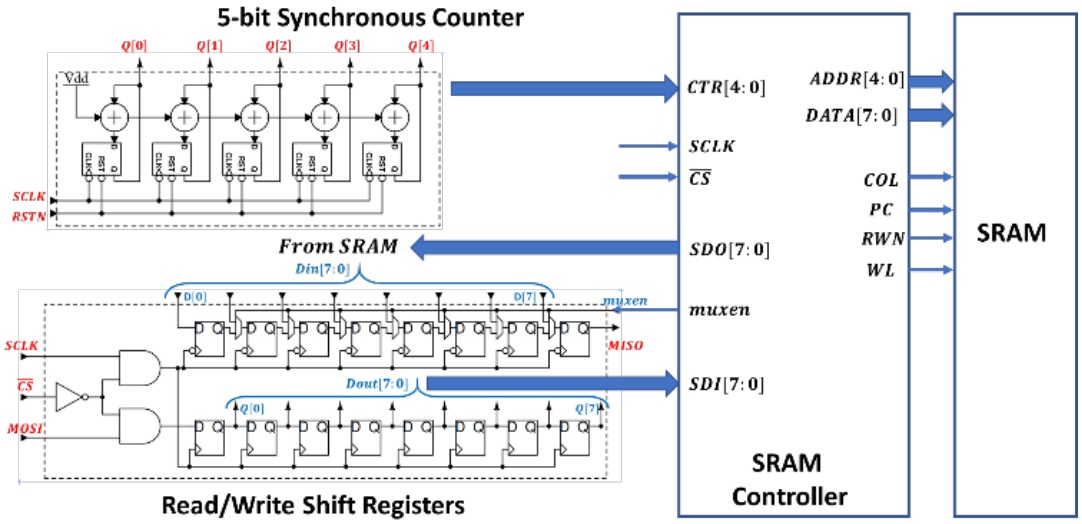

Figure 6: Block diagram of the SRAM Controller.

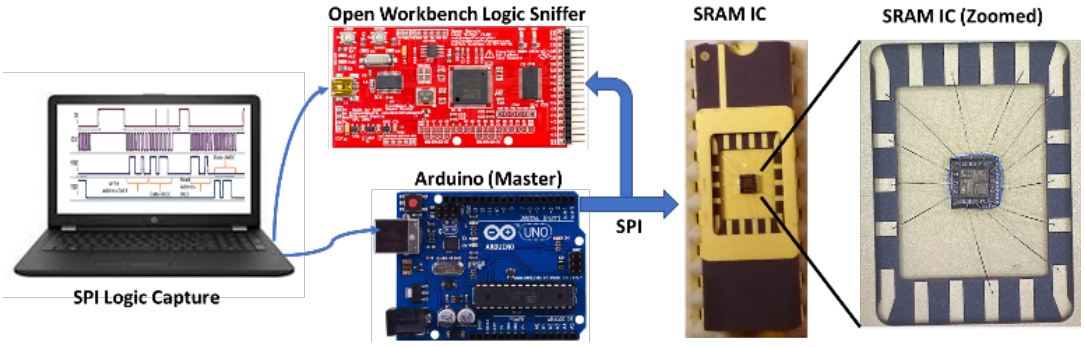

Figure 7: Measurement setup.

(saturation/linear) (Leblebici et al., 2014), it can be shown that M_3 needs to stronger M_5 and M_1 needs to be stronger than M_3. Since the structure is symmetric, the same constraints for M_6, M_4, M_2.

2.4 SRAM Controller

The core contribution of this work is to create all the control signals for SRAM from the master clock ($SCLK$). Figure 6 shows the block diagram of the overall architecture of the controller. A 5-bit synchronous counter is used to keep track of the number of cycles after CS goes low to create the appropriate synchronous timing signals as shown in Fig. 3 and Fig. 4. Two 8-bit shift-registers are used to convert serial data to parallel and vice-versa. The SRAM controller uses a combination of comparators and flip-flops to generate the required control signals for the SRAM.

3 Measurement Setup and Results

A picture-schematic of the measurement setup is shown in Figure 7. An Arduino development was used as the master to communicate with the SRAM using the SPI protocol. An Open Workbench Logic Sniffer was used to probe the SPI signals and plot them to verify the functionality of the fabricated SRAM IC. Figure 8 shows the Logic Sniffer output for a SPI transaction writing location `0x19` with `0xDC` and reading the same location. As seen from the annotation, the first bit on $MOSI$ is 0 denoting a write, the next five is the address `0x19` and the second word is `0xDC`. The last two byte is a read operation of the same location and seen from the $MISO$ output of the SRAM it is `0xDC`.

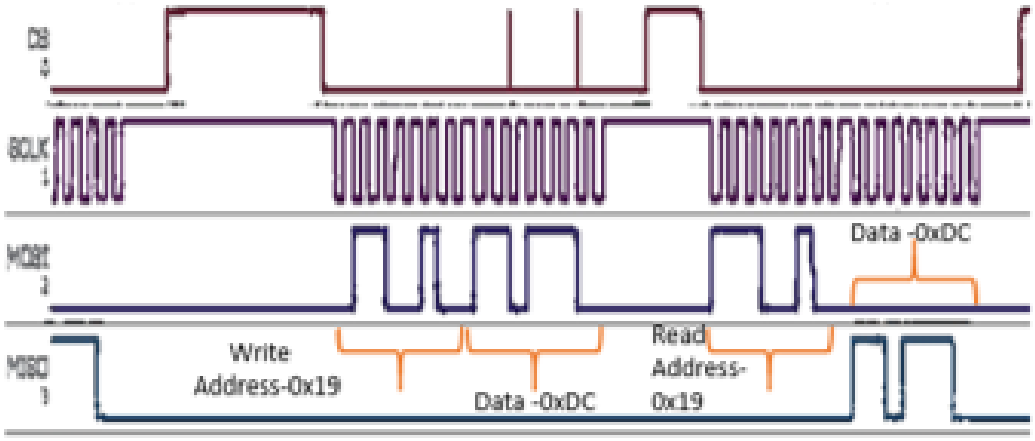

Figure 8: Measurement result.

4 Conclusion

A SRAM design in $0.6\mu m$ CMOS technology is demonstrated to be a viable candidate for IoT-based sensor nodes where power consumption is the top priority and the access rates are slow which can be done over a serial bus such as the SPI protocol.

5 Acknowledgement

We acknowledge the support of Deepika Kumari, Gautam Kumar, Pragya Tiwari, Sameer Namdeo, Shiva Prasad Das, Suruchi Kumari and Unnati Kumari Gupta for their help in simulation and layout.

References

Leblebici, Y., Kim, C. W., and Kang, S.-M. S. (2014). *CMOS Digital Integrated Circuits Analysis & Design*. McGraw-Hill Education, 4 edition, ISBN: 978-0-07-338062-9.

Leens, F. (2009). An introduction to I2c and SPI protocols. *IEEE Instrumentation Measurement Magazine*, 12(1):8–13, ISSN: 1941-0123, DOI: 10.1109/MIM.2009.4762946.

Weste, N. and Harris, D. (2011). *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education, ISBN: 978-0-13-300147-1.