Michael Eller

ECE 3430

Lab 6

28 September 2015

SPI

Goals:

The purpose of this lab was to read and write to the status register using SPI and bit banging. AAI was used to test the read and write functions designed in this lab.

Report:

Collaborators: Mike Verdicchio, Brendon Koch, Fran Clifton

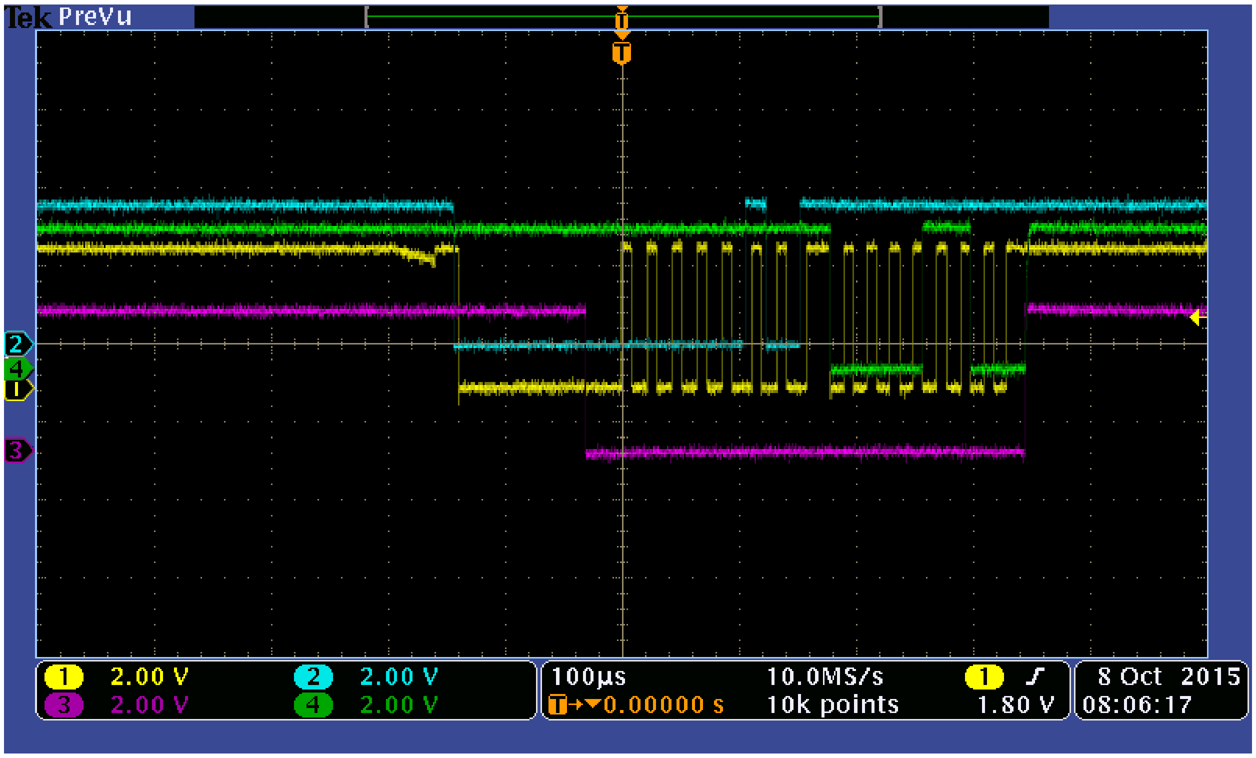

Figure 1: Clock -> yellow, Chip Enable -> purple, SI -> blue, SO -> green

The scope shot above shows the read status register function. The op-code, shown on the blue line, 0x05 (read-status-register) was sent over the first eight clock cycles. The next eight clock cycles were used to read the values from the status register, shown on the green line, in single bits. The chip enable (purple line) was driven low for the duration of the function.

Important Functions:

- sendData():
  This function sends eight individual bits over eight clock cycles. The most significant bit is sent and then the number is bit shifted.

- readData():
  This function builds an eight bit number over eight clock cycles. The most significant bit is received from the flash first. Then the number is shifted and the process is repeated until the number is completely received.

- wait_while_busy():
  This function checks the busy bit continuously and only returns when the bit is LOW.

- writeSR():
  Writing to the status register was enabled first. Then the write-status-register code was sent and then sendData() is called with the desired data passed to it.

- readSR():
  The only difference between this function and the readData() function is that this function sends the read-status-register op-code first.

Program Behavior:

The expected behavior of my program is simple: read and write to the status register and write values to the entire flash memory using AAI pseudo-random number generation. This program is unique because the clock functionality is completely created by software. Each operation that the flash has utilizes a specific series of synchronous tasks.

To read and write to the status-register, data had to be sent bit-by-bit over the software simulated clock cycles. The clock was created using an output pin on the MSP430. The biggest problem in this section was that the HOLD pin has to be set to 1 for any operations to be completed. To read the status register, the RSR (read-status-register) operation code was sent to the flash, which told it to send the status register bits over the next eight clock cycles. To write to the status register, the WSR (write-status-register) operation code had to be sent and then the data had to be sent over the next eight clock cycles.

The AAI function was supposed to write predictable ‘random’ numbers to all of the flash memory. The program was then supposed to check all of those values using the LEDs as indicators. Unfortunately the program does not successfully do this. The process completes but all the values in memory are not what was generated by the AAI() function. Reading and writing to the status register was successful, so the problem must be isolated to the AAI() functions.