Lab 5 – Memory Test

ECE531 Digital TPE

In this lab, you will be designing tests for the flash program memory and RAM of the pic16f883. The flash program memory of the device can be accessed in Program/Verify mode, so this test could be completed entirely with our test program. The PIC RAM can only be accessed during runtime, so testing of the RAM would require creation of another testing application to run on the PIC. You will not need to complete this step, but you will need to design the test procedure with pseudocode for the test application which would run on the PIC.

Part I – RAM Memory test

The volatile memory of the pic16f883 can only be accessed at runtime, and therefore requires a test application running on the PIC. As will be discussed in class, memory is generally tested through March algorithms. These algorithms are intended to cover common memory faults with linear time complexity, such as stuck-at faults, transition faults and coupling faults. The traditional memory March tests are based on bit-oriented tests. However, since most memory is read and written in blocks of words, a modification is needed for the March algorithms.

If the bit-oriented March algorithms were used without modification, individual cell faults and coupling faults between cells in different words (or inter-word faults) would still be detected. However, coupling faults between cells in the same word (or intra-word faults) would not be detected, since they would not be accessed in the sequential pattern expected, but simultaneously. The pic16f883 has 8 bit, or one byte, words in RAM. A word-oriented March test based off of the bit-oriented March test March-LR is shown below (hex values are given for brevity). Please see the following article for more details on the differences of bit-oriented and word-oriented tests, and how this modified March test was constructed:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.5.2095&rep=rep1&type=pdf

$$\left\{\begin{array}{c} \Updownarrow (w0x00); \Downarrow (r0x00, w0xFF); \Uparrow (r0xFF, w0x00, r0x00, r0x00, w0xFF); \Uparrow (r0xFF, w0x00); \\ \Uparrow (r0x00, w0xFF, r0xFF, r0xFF, w0x00); \Uparrow (r0x00, w0x55, w0xAA, r0xAA); \\ \Downarrow (r0xAA, w0x55, r0x55); \Uparrow (r0x55, w0x33, w0xCC, r0xCC); \Downarrow (r0xCC, w0x33, r0x33); \\ \Uparrow (r0x33, w0x0F, w0xF0, r0xF0); \Downarrow (r0xF0, w0x0F, r0x0F); \Uparrow (r0x0F) \end{array}\right\}$$

A word-oriented March test should be used to test the RAM memory of the pic16f883. Since the RAM memory is only accessible at run-time, a new PIC application would be needed which can execute this March test and provide some feedback on the success or failure of the test. For instance, you could set one output pin high on completion and use one or more ports to report the result. You may also find it useful to have an input pin that allows you to control execution from your test program. Implementing the entire March test shown above would result in a large number of opcodes for the test application. For simplicity, consider a word-oriented version of the MATS test, shown below. This makes use of what is known as a checkerboard pattern (alternating 1's and 0's). This test could be easily extended to implement MATS+ or MATS++ if desired.

$$\{\Updownarrow (w0x55); \Updownarrow (r0x55, w0xAA); \Updownarrow (r0xAA); \}$$

Create a test plan for testing the RAM memory. This should include all necessary connections, and pseudocode of the testing procedure for both the test program and the PIC application. One issue that we must contend with when testing the RAM, the Special Function Registers (SFR) of the PIC are in RAM address space. These registers affect functionality, and some are read-only, so it is not feasible to test the entire address space. You may limit your test to General Purpose RAM. You do not need to implement the RAM test, however if you did you would need to develop the test application in assembly to provide the necessary control over timing and register usage.

Part II – Flash Memory test

As will be discussed in class, memory is generally tested through March algorithms. These algorithms are intended to cover common memory faults with linear time complexity, such as stuck-at faults, transition faults and coupling faults. Traditional memory March tests do not work for flash memory, however. Flash memory is non-volatile memory which uses floating-gate transistors to store bits. Unlike traditional RAM which can read or write with random-access, flash memory can only read with random-access. Writing involves either programming or erasing a floating-gate. Programming can be performed with random-access, but erasing can only be performed in blocks.

Erasing a block ("flashing") sets all of the bits in the block to logic 1. Programming a cell turns that cell to a logic 0. Therefore, in flash memory, we can write a "0", but we cannot write a "1". Thus, traditional March tests which include w1 commands are not possible. Additionally, the different hardware of flash memory introduces new coupling effects and potential faults between cells. Testing on the PIC is further complicated by the fact that the flash memory is NAND structure and the program counter can only be incremented by 1 when in Program/Verify mode. This prevents true random-access for read or programming operations.

For more details on the differences in testing flash memory, you may reference the article at http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04167999. An example of an algorithm which can be used for testing flash memory is the Cocktail-March algorithm shown below. This is a word-oriented algorithm, meaning that we will write to one word at a time rather than having to issue a write for each individual bit. The notation for the algorithm is similar to standard March algorithms with two additions: *f* refers to a flash operation (erase entire block, which sets all bits to 1), and *p* refers to program operation (replaces w0). The algorithm is given below for 4-bit words, this can be extended for the 14-bit words of the program flash memory.

$$\left\{\begin{array}{c}(f);\Uparrow(r1111,p0000,r0000);\Updownarrow(r0000);\\(f);\Downarrow(r1111,p0000,r0000);\Updownarrow(r0000);\\(f);\Updownarrow(p0011);\Updownarrow(r0011);(f);\Updownarrow(p1100);\Updownarrow(r1100);\\(f);\Updownarrow(p0101);\Updownarrow(r0101);(f);\Updownarrow(p1010);\Updownarrow(r1010)\end{array}\right\}$$

As may be seen, the number of steps or marches when using 14-bit words can become very lengthy. For simplicity, we will only execute the steps in the 4th row shown above. The patterns in the 4th step are referred to as standard backgrounds or checkerboard patterns. These help to identify intraword faults due to coupling defects. It is a common practice to simply perform a checkerboard test of memory. Datalog your results by storing the address of any word that fails (you only need to detect one failure). An address outside of addressable space can be used to indicate success, such as 0x2000. If you would like to capture more detailed information, such as *all* failed addresses, it would be possible to execute a directed test, where the ATE program is providing a synchronizing control signal, and then capturing a result signal to identify which addresses succeed and fail, but this is not required.

Write-up your results

Part I – submit test plan for RAM test, including schematic, and pseudocode for test program and PIC test application.

Due: Wednesday, May 4, 2016

Part II – Record your test time. Turn in your final code and datalogs. Discuss your test results.

Due: Wednesday, May 11, 2016