Self-testing in embedded systems: Hardware failure

Colin Walls (/user/Colin Walls)

FEBRUARY 09, 2016

Hard faults are permanent malfunctions and show up in three forms:

1. Memory not responding to being addressed at all.
2. One or more bits are stuck on 0 or 1.
3. There is cross talk; addressing one bit has an effect on one or more others.

(1) results in a trap in the same way as the aforementioned peripheral failure. Ensuring that a suitable trap handler is implemented addresses this issue.

The other forms of hard memory failure, (2) and (3), may be detected using self-test code. This testing may be run at start-up and can also be executed as a background task.

Start-up memory testing

There are a couple of reasons why testing memory on start-up makes a lot of sense. As with most electronic devices, the time when memory is most likely to fail is on power-up. So, testing before use is logical. It is also possible to do more comprehensive testing on memory that does not yet contain meaningful data.

A common power-up memory test is called “moving ones”. This tests for cross-talk – i.e. whether setting or clearing one bit affects any others. Here is the logic of a moving ones test:

```
set every bit of memory to 0
for each bit of memory
{
    verify that all bits are 0
    set the bit under test to 1
    verify that it is 1
    verify all other bits are 0
    set the bit under test to 0
}
```


MOST COMMENTED

07.06.2017
Want a robot to iron your shirts? (/electronics-blogs/max-unleashed-and-unfettered/4458601/Want-a-robot-to-iron-your-shirts-)

06.19.2017
Saving power with relays and solenoids (/electronics-blogs/without-a-paddle/4458541/Saving-power-with-relays-and-solenoids)

07.10.2017
A solution looking for a problem (/electronics-blogs/without-a-paddle/4458611/A-solution-looking-for-a-problem)

07.14.2017
The Write Stuff: Tutorial template in high-visibility format (/electronics-blogs/the-write-stuff/4458625/The-Write-Stuff--Tutorial-template-in-high-visibility-format)

07.03.2017
Public key infrastructure for the IIoT (/electronics-blogs/say-what-/4458599/Public-key-infrastructure-for-the-IIoT)

The same idea may be applied to implement a moving zeros test. Ideally, both tests should be used in succession. Coding these tests needs care. The process should not, itself, use any memory – code should be executed from flash and all working data must be stored in CPU registers.

With increasingly large amounts of memory, the time taken to perform these tests escalates exponentially and could result in an unacceptable delay in the start-up time for a device. Knowledge of memory architecture can enable optimization. For example, cross talk is more likely within a given memory array. So, if there are multiple arrays, the test can be performed individually on each one. Afterwards, a quick check can be performed to verify that there is no cross-talk between arrays, thus:

```
fill all of memory with 0s
for each memory array
{
    fill array with 1s
    verify that other arrays still contain just 0s
    fill array with 0s
}
```

This can then be repeated with all of memory starting full of ones.

Background memory testing

Once “real” code is running, comprehensive memory testing is no longer possible. However, testing of individual bytes/words of memory is possible, so long as tiny interruptions in software execution can be tolerated. Most embedded systems have some idle time or run a background task, when there is no real work to be done. This may be an opportunity to run a memory test.

A simple approach is to write, read and verify a series of bit patterns: all ones, all zeros and alternate one/zero patterns. Here is the logic:

```
for each byte of memory
{
    turn off interrupts
    save memory byte contents
    for values 0x00, 0xff, 0xaa, 0x55
    {
        write value to byte under test
        verify value of byte
    }
    restore byte data
    turn on interrupts
}
```

Implementing this code requires a little care, as an optimizing compiler is likely to conclude that some or all of the memory accesses are redundant and optimize them away. The compiler is optimistic about memory integrity.

In part two of this two-part series, we'll look at self-testing approaches for mitigating software failures.

Colin Walls (http://www.embedded.com/user/Colin%20Walls) has over thirty years experience in the electronics industry, largely dedicated to embedded software. A frequent presenter at conferences and seminars and author of numerous technical articles and two books on embedded software, Colin is an embedded software technologist with Mentor Embedded [the Mentor Graphics Embedded Software Division], and is based in the UK. His regular blog is located at: http://blogs.mentor.com/colinwalls (http://blogs.mentor.com/colinwalls). He may be reached by email at colin_walls@mentor.com

RELATED CONTENT

02.23.2016 | DESIGN

Self-testing in embedded systems: Software failure (/design/debug-and-optimization/4441376/Self-testing-in-embedded-systems--Software-failure)

02.09.2016 | DESIGN

Self-testing in embedded systems: Hardware failure (/design/debug-and-optimization/4441375/Self-testing-in-embedded-systems--Hardware-failure)

04.16.2015 | DESIGN

Testing code that uses an RTOS API (/design/prototyping-and-development/4439214/Testing-code-that-uses-an-RTOS-API)

11.25.2013 | DESIGN

Vintage multicore (/design/mcus-processors-and-socs/4424833/Vintage-multicore)

03.15.2014 | DESIGN

Multicore basics: AMP and SMP (/design/mcus-processors-and-socs/4429496/Multicore-basics)

< Previous (/design/debug-and-optimization/4441375/1/Self-testing-in-embedded-systems--Hardware-failure) Page 2 of 2 Next >

7 COMMENTS

WRITE A COMMENT

SteveMerrick (/User/SteveMerrick) **POSTED: MAR 24, 2016 1:20 PM EDT**

Great post! Focussing on resilience (tolerance of failure) as well as error-avoidance!

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Colin Walls (/User/Colin%20Walls) **POSTED: MAR 24, 2016 1:41 PM EDT**

Thank you.

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Cdhmanning (/User/Cdhmanning) **POSTED: FEB 12, 2016 6:42 PM EST**

The bit that often stumps people is this:

Ok, we've detected a problem. Now what do we do?

Throwing an exception or rebooting just immediately converts a defect into a failure.

Far too often I've seen code with asserts or other measures that change a very trivial issue into something tragic.

Get bad reading from temperature sensor over I2C bus triggers assert triggers reset the system triggers catastrophic failure of mission (be that wash some clothing, cook food or a space launch).

That's essentially what caused the Ariane 5 crash in 1996.

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Colin Walls (/User/Colin%20Walls) **POSTED: FEB 13, 2016 3:43 AM EST**

@Cdhmanning - You are correct. This area will be covered in Part 2 of the article. But I'll admit that it won't be totally comprehensive, as so much is application-specific.

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Cdhmanning (/User/Cdhmanning) **POSTED: FEB 13, 2016 9:33 PM EST**

A couple of years back my car had a failure of its Mass Airflow Sensor. Without MAS numbers, the whole fuel mix/ignition system fails.

I dug into this and found out a bit more about this. It turns out that the ECU also models the expected MAS numbers based on other sensor values. If the MAS number is within bounds, then the MAS sensor is used. If the MAS number goes out of bounds, then the modelled number is substituted.

The engine runs a bit rougher, but it continues to run effectively.

Not sure what you can do if something like RAM fails. Eagerly awaiting your thoughts....

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

MWagner_MA (/User/MWagner_MA) **POSTED: FEB 10, 2016 7:26 AM EST**

Don't forget some of your peripherals may have a built-in self test such as MEMS accelerometers. We have had a failure where the core of the device (accelerometer) communicated with the CPU, yet an axis failed to provide live data (output was fixed). Exercising the built-in test would have detected that failure mode.

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Colin Walls (/User/Colin%20Walls) **POSTED: FEB 10, 2016 9:19 AM EST**

@MWagner - Absolutely. Part of the point of the article is to make you think about these possibilities. Smart peripheral devices are very likely to have such facilities.

REPLY (/?SCREENTOVIEW=LOGIN&REDIRECTTO=HTTP://WWW.EMBEDDED.COM/DESIGN/DEBUG-AND-OPTIMIZATION/4441375/2/SELF-TESTING-IN-EMBEDDED-SYSTEMS--HARDWARE-FAILURE)

Real-world Applications (/development/real-world-applications)

Safety & Security (/development/safety-and-security)

System Integration (/development/system-integration)

ASPENCORE NETWORK

EBN (http://www.ebnonline.com/) EDN (http://www.edn.com/) EE Times (http://www.eetimes.com/) EEWeb (https://www.eeweb.com/) Electronic Products (http://www.electronicproducts.com/) Electronics-Tutorials (http://www.electronics-tutorials.ws/) Embedded (http://www.embedded.com/) Planet Analog (http://www.planetanalog.com/) ElectroSchematics (http://www.electroschematics.com/) Power Electronics News (http://www.powerelectronicsnews.com/) TechOnline (http://www.techonline.com/) Datasheets.com (http://www.datasheets.com/) Embedded Control Europe (http://www.embedded-control-europe.com/) Embedded Know How (http://www.embedded-know-how.com/) Embedded News (http://embedded-news.tv/) IOT Design Zone (http://iot-design-zone.com/) Motor Control Design (http://motor-control-design.com/) Electronics Know How (http://electronics-know-how.com/)

GLOBAL NETWORK

EE Times Asia (http://www.eetasia.com/) EE Times China (http://www.eet-china.com/) EE Times India (http://www.eetindia.co.in/) EE Times Japan (http://eetimes.jp/) EE Times Taiwan (http://www.eettaiwan.com/) EDN Asia (http://www.ednasia.com/) EDN China (http://www.ednchina.com/) EDN Taiwan (http://www.edntaiwan.com/) EDN Japan (http://ednjapan.com/) ESM China (http://www.esmchina.com/)
