Pre-silicon Software Development of Linux MTD Drivers

October 30, 2009
Ideally, device drivers are already available when customers get their hands on first samples. A simulator based approach can help work around hardware issues that might exist during this early phase (alpha phase).
Flash Basics

- Electrical charge is moved onto an isolated storage element in order to **program** data (done on a bit basis for NOR flash or on a page basis for NAND flash)
- Electrical charge is removed from the isolated area in order to **erase** data (done on a sector/block basis, e.g. 128 kB)
- To **read** data, a reference voltage is applied between source and drain and the current is measured (if current flows, the cell has been programmed)
- A special command set is being used to issue and to control the program and erase operations
# S29WS-R Command Set

<table>
<thead>
<tr>
<th>Command Sequence</th>
<th>Cycles</th>
<th>First Addr/Byte/Word</th>
<th>First Data</th>
<th>Second Addr/Byte/Word</th>
<th>Second Data</th>
<th>Third Addr/Byte/Word</th>
<th>Third Data</th>
<th>Fourth Addr/Byte/Word</th>
<th>Fourth Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>1</td>
<td>RA</td>
<td>RD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>XX</td>
<td>F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write Buffer Load</td>
<td>4-35</td>
<td>(SA) AAA (SA) 555</td>
<td>25</td>
<td>(SA) 555 (SA) AAA</td>
<td>WC</td>
<td>(SA) PA (12)</td>
<td>PD</td>
<td>(SA) PA (13)</td>
<td>PD</td>
</tr>
<tr>
<td>Buffer to Flash</td>
<td>1</td>
<td>(SA) AAA (SA) 555</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chip Erase</td>
<td>2</td>
<td>(SA) AAA (SA) 555</td>
<td>80</td>
<td>(SA) 555 (SA) AAA</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sector Erase</td>
<td>2</td>
<td>(SA) AAA (SA) 555</td>
<td>80</td>
<td>(SA) 555 (SA) AAA</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Register Read</td>
<td>2</td>
<td>(SA) AAA (SA) 555</td>
<td>70</td>
<td>(SA)</td>
<td>RR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Register Clear</td>
<td>1</td>
<td>(SA) AAA (SA) 555</td>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## ID/CFI Command Definitions

<table>
<thead>
<tr>
<th>ID/CFI Command</th>
<th>ID/CFI Entry (8) (11)</th>
<th>(SA) XAA (SA) X55</th>
<th>90 or 98</th>
<th>ID/CFI Read</th>
<th>1</th>
<th>(SA) RA</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID/CFI Exit</td>
<td>1</td>
<td>XXX</td>
<td>F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Simulator Based Approach for MTD Drivers

MTD Stack

- User Apps
- FFS
- Chip Driver
- Board Driver

Kernel Space

User Space

Device Model

Communication Channel
e.g. TCP/IP Socket

Simulation Server
DevSim is a cycle accurate SystemC model of Spansion Flash. It supports the following formats:

- DevSim object for Cadence Design Systems’ NC-Verilog on Linux
- DevSim object for Synopsys’ VCS on Linux
- DevSim object for Microsoft Visual Studio
- DevSim executable/server and command line client

Benefits:
- Developed by Spansion for internal verification
- Cycle accurate
- Detects and reports timing violations
- Creates a waveform of the session
- Resource sensitive - Simulates as much of the array as you use
- Truly non-volatile from session to session
- View the array outside of a session
- Array pre-loading supported
- Flash architecture controlled by CFI input file
- Works great for software development
  - Program and erase times can be altered to reduce real-time execution
Communication Channel Protocol

Read word

1 byte 4 bytes

0x10 Addr

1 word

Data

Request

Write word

1 byte 4 bytes 1 word

0x20 Addr Data

Request

Read block (optional)

1 byte 4 bytes 4 bytes

0x11 Addr Length

Length bytes

Data ......
MTD Board or Map Driver

The pivotal element connecting a chip driver (NOR flash) to the hardware is a `map_info`, see include/linux/mtd/map.h

```c
struct map_info {
    char *name;
    unsigned long size;
    resource_size_t phys;
    void __iomem *virt;
    void *cached;
    int bankwidth;

    map_word (*read)(struct map_info *, unsigned long);
    void (*write)(struct map_info *, const map_word, unsigned long);
    void (*copy_from)(struct map_info *, void *, unsigned long, ssize_t);
    void (*copy_to)(struct map_info *, unsigned long, const void *, ssize_t);
    void (*inval_cache)(struct map_info *, unsigned long, ssize_t);
    void (*set_vpp)(struct map_info *, int);
    ...
};
```

Function pointers to issue read & write cycles to the flash device

All the driver has to implement are functions that route the r/w requests over the communication channel to the model.
Proper Synchronization

- Replacing the standard I/O calls used for a physical device (`__raw_readw()`, `__raw_writew()` ) by socket calls changes the characteristics of the I/O from non-blocking to blocking.

- This can cause issues if the kernel holds spin-locks during the I/O. Another thread trying to acquire the spin-lock at the same time can stall the entire system (deadlock).

- Some MTD chip drivers hold a spin-lock while issuing commands. This spin-lock needs to be released before the socket operations take place and reacquired afterwards.

- To prevent other threads from accessing the MTD while the socket operations are ongoing, the device has to be put into a special mode so that other threads have to wait, i.e. sleep and try again later.
Live Demo

```
[gernot@MUC-N-0037:/home/gernot/FlashModel_Projects/Sys] ~
File   Edit  View   Terminal  Help
Read, address: 0x000000aa, data: 0x0008
Read, address: 0x000000ac, data: 0x0008
Read, address: 0x000000ae, data: 0x0010
Write, address: 0x00000000, data: 0x00f0 *
Write, address: 0x00000000, data: 0x00ff *
Write, address: 0x00000aaa, data: 0x0071 *
```

```
[gernot@MUC-N-0037:/home/gernot/devsim_mtd_nor]
[root@MUC-N-0037 devsim_mtd_nor]# insmod devsim.ko
[root@MUC-N-0037 devsim_mtd_nor]# Oct 22 13:37:18 MUC-N-0037 kernel: DevSim initialization...
Oct 22 13:37:18 MUC-N-0037 kernel: DevSim NOR flash 16-bit: Found 1 x16 devices at 0x0 in 16-bit bank
Oct 22 13:37:18 MUC-N-0037 kernel: Spansion Extended Query Table at 0x0040
Oct 22 13:37:18 MUC-N-0037 kernel: number of CFI chips: 1
Oct 22 13:37:18 MUC-N-0037 kernel: Creating 1 MTD partitions on "DevSim NOR flash 16-bit":
Oct 22 13:37:18 MUC-N-0037 kernel: 0x000000000000-0x000000010000 : "DevSim Partition 1"

[root@MUC-N-0037 devsim_mtd_nor]# flash_erase /dev/mtd0
Erase Total 1 Units
Performing Flash Erase of length 131072 at offset 0x0 done
[root@MUC-N-0037 devsim_mtd_nor]#
```
Further Advantages

Next to the early availability, a simulator based approach has further advantages:

✓ Simplified debugging
✓ Software trace support (virtual logic analyzer)
✓ Errors can easily be inserted
✓ Different device models can be loaded quickly
✓ Non-standard bus widths and setups can easily be tested

✓ Spec violation checks can be added
Summary

- A new approach for pre-silicon software development of MTD chip drivers has been proposed.
- In the context of the development of a new NOR flash driver it proved to be stable and beneficial.
- It can be used not only at an early stage where no Si is available yet, it also greatly simplifies testing and debugging.
- A similar approach should be conceivable for other memory technologies such as NAND or SPI devices.
- All it needs is to route the flash specific interface over a communication channel to a device model.
Thank You!
Trademark Attribution

Spansion®, the Spansion logo, MirrorBit®, MirrorBit® Eclipse™, ORNAND™, ORNAND2™, HD-SIM™ and combinations thereof, are trademarks of Spansion LLC in the U.S. and other countries. Other names used are for informational purposes only and may be trademarks of their respective owners.

This document is for informational purposes only and subject to change without notice. Spansion does not represent that it is complete, accurate or up-to-date; it is provided “AS IS.” To the maximum extent permitted by law, Spansion disclaims any liability for loss or damages arising from use of or reliance on this document.