Theory and practice of flash memory mobile forensics

Salvatore Fiorillo
Information Security Consultant
Theosecurity.com

Abstract

This paper is an introduction to flash memory forensics with a special focus on completeness of evidence acquired from mobile phones. Moving through academic papers and industrial documents will be introduced the particular nature of non-volatile memories present in nowadays mobile phones; how they really work and which challenges they pose to forensic investigators. Then will be presented an advanced test in which some brand new flash memories have been used to hide data in man-made bad blocks: the aim is to verify if forensic software tools are able to acquire data from such blocks, and to evaluate the possibility to hide data at analysts' eyes.

Keywords

Mobile forensic, OneNAND, NAND, NOR, bad blocks, wear levelling, ECC, FTL

PART ONE: FLASH MEMORY BASICS

The mobile environment

A Mobile Equipment (ME) is here understood as the radio handset portion of a more generic mobile phone (Jansen and Ayers, 2007), made by various components, most important of which are presented in the representation below (fig. 1).

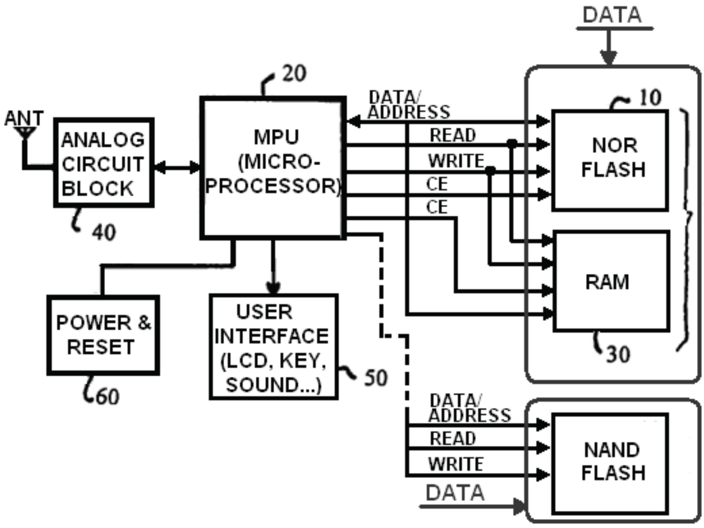


Fig. 1: Old mobile equipment layout with optional NAND module (Kwon, 2009)

During its evolution mobile phone passed from the PDA phase up to nowadays smart phones that lessen differences with personal computers (ibid). Storage capability also increased dramatically ranging from few Kilobits at very beginning up to several Gigabits of current mobile phones, increasing the space where data can be stored or hided, and adding complexity to work of law enforcement officers (Al-Zarouni, 2006): this paper aims to contribute in the shifting of the flash forensic field from the *knowable* to *known* Cynefin domain (Kurtz and Snowden, 2003).

On nowadays mobile equipment there are generally two memories: one for the operating system (the NOR flash) and the other (the NAND flash) for user data (Chang and Kuo, 2004). The extent of this paper is limited to data stored in NAND flashes: volatile RAM and SIM card analysis will be kept aside.

NOR and NAND

Flash memory is a non-volatile memory that can be electrically erased and rewritten with a specific process: likely hard disk (even very different for the lack of physical mechanisms), flash memory does not need power to maintain data stored in the chip for future access (O'Kelly, 2007). Coming from evolution of EPROM, the two main kind of flash memories are NAND and NOR. NOR flash have long erase and write times, but it is nearly immune to corruption and bad blocks, allows random access to any memory location and almost all controllers on mobile phones have a NOR interface (Pon et al., 2007). NAND flash offers higher density capabilities, is cheaper than NOR, is less stable, need a supporting separate RAM to work (ibid) and only allow sequential access mode (Gal and Toledo, 2005). In mobile equipments usually the NOR stores executable software (i.e. BIOS) and the NAND data storage such as image or mp3 files (Peng, 2006, Raghavan et al., 2005). In Appendices is reported a table comparing the two flash memories.

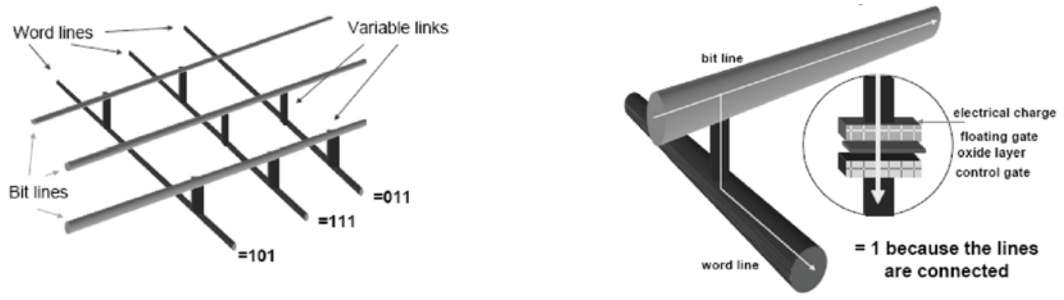


Fig. 2 Basic design of memory chip (left) and flash memory links (right) (O'Kelly, 2007)

Code model

There are two techniques to execute *program code* on flash devices (Numonyx, 2008a): Store and Download (SnD), requiring external RAM, and eXecute in Place (XiP) - faster than SnD but requiring random access. NOR uses XiP while NAND uses SnD.

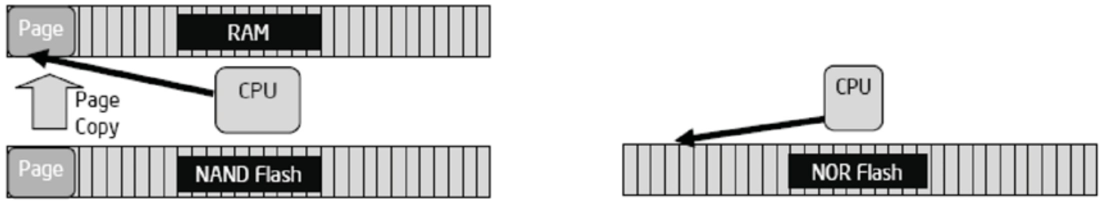


Fig. 3 Store and Download Code Model (left) and XiP Code Model (right) (Numonyx, 2008a)

One-way programming

Flash devices are only able to program a value from 1 to 0 but not from 0 to 1, so when data is updated, it is written to a new location and the old location is marked as *invalid* (Numonyx, 2008a). The invalid location is then erased - usually during a background process - before being reused.

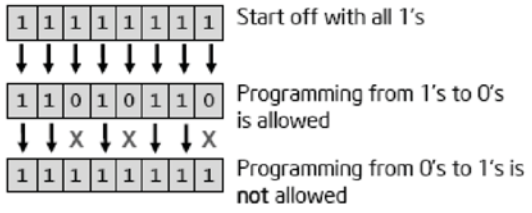

1 1 1 1 1 1 1 1
Start off with all 1's
1 1 0 1 0 1 1 0
Programming from 1's to 0's is allowed
X X X
1 1 1 1 1 1 1 1
Programming from 0's to 1's is not allowed

Fig. 4 Flash Programming Limitations (Numonyx, 2008a)

Wearing erase-write cycle

Unlike hard disks, the erase-write cycle in flash memories is a physically exhausting activity, so the lifetime on a flash memory is inversely proportional to its use. A location can be programmed and erased reliably up to 100,000 and 10,000 times respectively and, as general rule, the following formula could be used to calculate the expected lifetime of NAND flash with FAT filesystem (Numonyx, 2008a). Techniques to circumvent the problem of flash wearing will be discussed in next pages.

$$\text{Expected lifetime} = \frac{\text{Size of NAND flash x number of erase cycles x FAT overhead}}{\text{Bytes written per day}}$$

Fat overhead include all management activities the filesystem needs to perform files administration (Hendrikx, 1998)

Flash Filesystem Architecture

The Flash Filesystem Architecture is based on logical unit (LUN) , blocks, pages and sectors (Intel, 2006, Numonyx, 2008a, Samsung, 1999). A LUN is a logical division of the whole memory land.

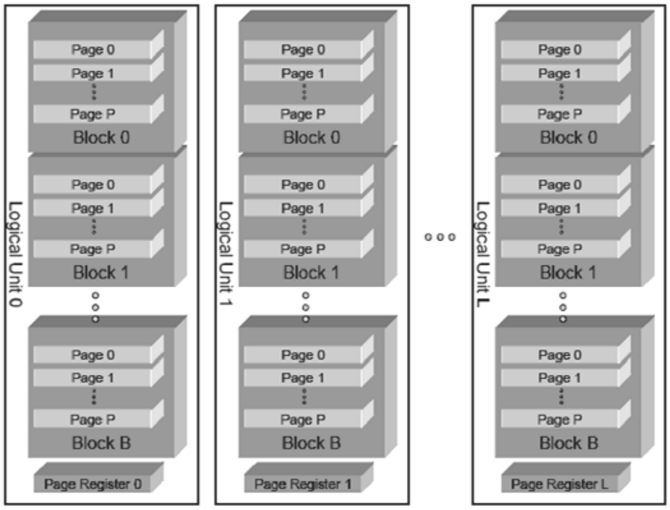


Fig. 5 Logical Units in NAND flash memories (Huffman, 2006)

LUNs are then split in blocks. Each block can vary in size, where the most common is 128KB. In the majority of NAND flash devices each block is made of 64 pages of 2KB each. A page is divided in two regions: the data area, and the spare area used for memory management purposes (more later). Pages are divided in sector units (or chunks) of 512 byte to emulate the popular sector size (ibid). The block is the smallest erasable unit while the page is the smallest programmable unit.

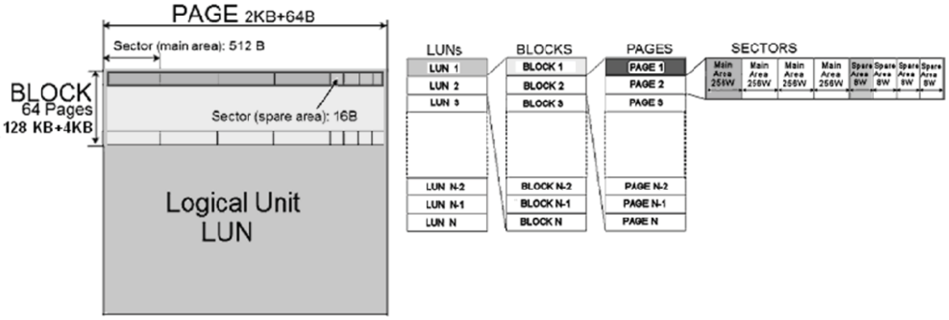


Fig. 6 Architecture of a flash memory

At first, a page was 528 bytes long as the original intent of the NAND Flash was to replace magnetic hard disk drives, so it was required a page to be big enough to store one sector (512 bytes) of data with extra 16 Bytes for management purpose (Inoue and Wong, 2004). Then, as capacity storage of flash increased, so did the default page size to comply with FAT file system. On 1Gb flash memory, there are 128 MB of addressable space: for hard drives sized up to 128 MB, the default cluster size in FAT system is 2KB with 4 sectors each, as in the flash memory except for the extra bytes (64B) (Microsoft, 2009)

The spare area

A spare area, referred also as out-of-band data, is a region generally made of 16 Bytes and there is one for each sector or chunks (Gal and Toledo, 2005, Raghavan et al., 2005); its size is not included in device capacity and it cannot be directly addressed (Elnec, 2009). One use of spare area is to store results of data verification: after a page has been erased, programmed or read, its status is verified with a particular algorithm (aka ECC – more next) and the relative output is later used to detect errors whenever the data is read back (BPMicrosystems, 2008). Spare area could store also information on the status of blocks and pages (Tsai et al., 2006), or other information similar to metadata seen in NFTS filesystem (Carrier, 2005, Casey, 2004). The following is a representation of spare area in Samsung OneNAND™ , for further information see (Samsung, 2005a).

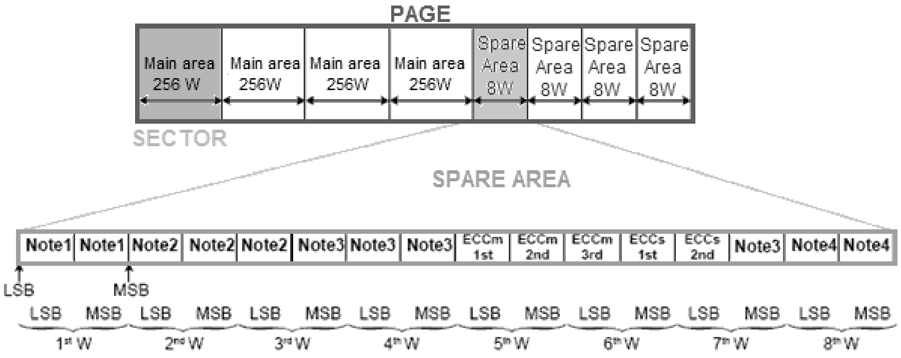
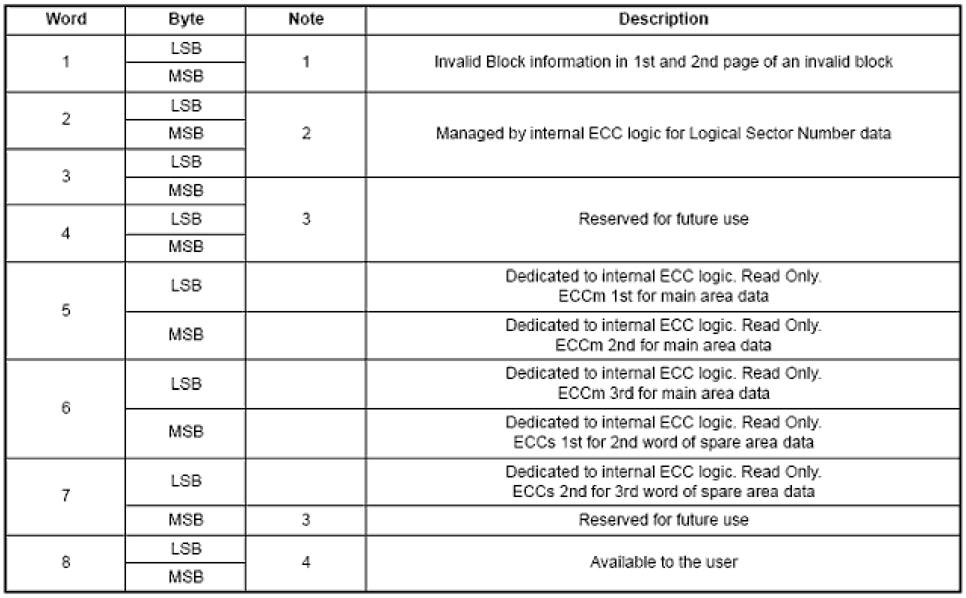


Fig. 7 Assignment of the spare area in the Internal Memory NAND on OneNAND™ (source: Samsung)



Word	Byte	Note	Description
1	LSB	1	Invalid Block information in 1st and 2nd page of an invalid block
	MSB		
2	LSB	2	Managed by internal ECC logic for Logical Sector Number data
	MSB		
3	LSB		
	MSB	3	Reserved for future use
4	LSB		
	MSB		
5	LSB		Dedicated to internal ECC logic. Read Only. ECCm 1st for main area data
	MSB		Dedicated to internal ECC logic. Read Only. ECCm 2nd for main area data
6	LSB		Dedicated to internal ECC logic. Read Only. ECCm 3rd for main area data
	MSB		Dedicated to internal ECC logic. Read Only. ECCs 1st for 2nd word of spare area data
7	LSB		Dedicated to internal ECC logic. Read Only. ECCs 2nd for 3rd word of spare area data
	MSB	3	Reserved for future use
8	LSB	4	Available to the user
	MSB		

Fig. 8 Spare Area Assignment in the Internal Memory NAND on OneNAND™ (source: Samsung)

The are two storage methods for spare areas: adjacent to data area or separate from it (Micron, 2006a). Looking at most of the Samsung datasheets it seems their mainly used model is the second one.

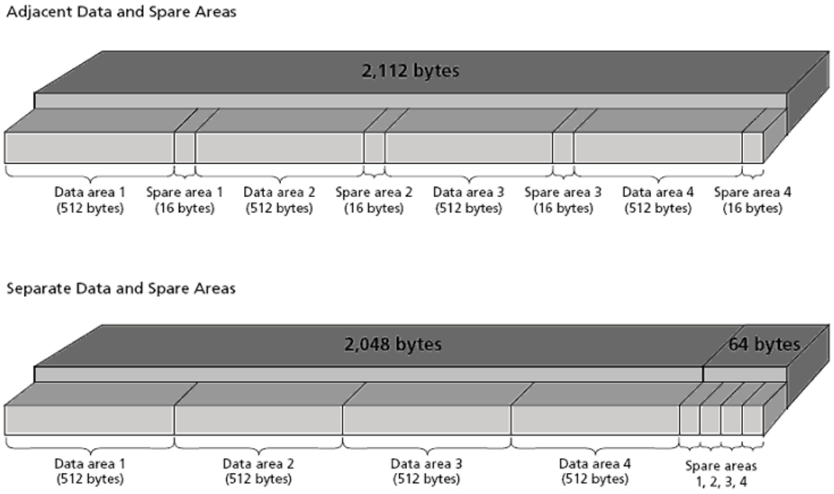


Fig. 9 Spare area storage methods (Micron, 2006a)

NAND vs. hard disk

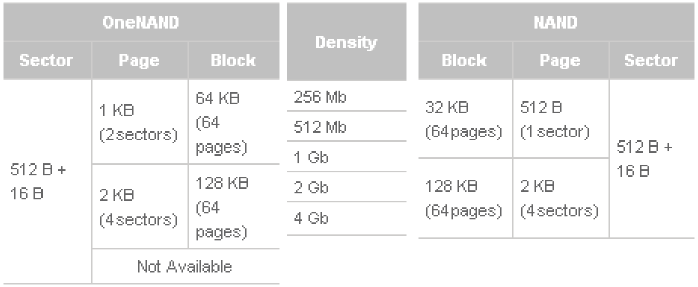
The main differences between flash devices and hard disks are (Raghavan et al., 2005):

standard size of sectors (see flash sector block size in the picture below);

unlike hard disks, the write and the erase operations in flash device can be an independent activity and related to the software using the flash apparatus;

flash chips have a limited lifetime due to erase wearing;

flash devices can be powered off without proper shutdown and still have consistent data: this is not possible in case of hard disk with normal file systems, so embedded systems need a specific file management flash oriented.



OneNAND			Density	NAND		
Sector	Page	Block		Block	Page	Sector
512 B + 16 B	1 KB (2sectors)	64 KB (64 pages)	256 Mb	32 KB (64pages)	512 B (1 sector)	512 B + 16 B
			512 Mb			
	2 KB (4sectors)	128 KB (64 pages)	1 Gb	128 KB (64pages)	2 KB (4sectors)	
			2 Gb			
	Not Available		4 Gb			

Fig. 10 Standard size of sector block of devices under 256 Mb and over 512 Mb density (source: Samsung)

Flash File-systems and Flash Translation Layer

"*A file system is a data structure that represents a collection of mutable random-access files in a hierarchical name space*" (Gal and Toledo, 2005). To operate with (legacy) host filesystem, NAND flash memories require either a specific filesystem or a specific driver. Actually we have both: indeed there are specific flash file systems (like YAFFS, JFFS, UBIFS, and LogFS) as well as specific driver better known as Flash Translation Layer.

"FTL is a driver that works in conjunction with an existing operating system (or, in some embedded applications, as the operating system) to make linear flash memory appear to the system like a disk drive" (Intel, 2006)

The main mission an FTL carries out is to support all tasks required for managing data transparently to host filesystem: i.e. a FAT filesystem will demand to the FTL all activities required to store and retrieve data properly to/from the NAND flash devices. (BPMicrosystems, 2008, Intel, 1998, Morris, 2007)

FTL main tasks are:

Mapping the storage area in virtual small sectors

Managing data on the flash so they appears to “write in place”

Housekeeping: as flash memories are subject to wear, it is required a software that will level the use of memory areas.

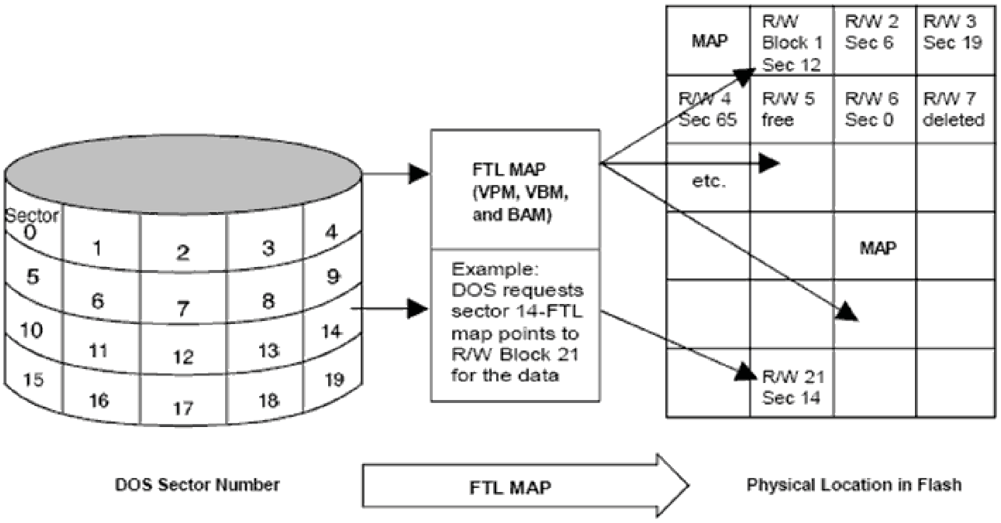


Fig. 11 FTL Sector Relocation (Intel, 1998)

FTL for NAND can come in several flavours: it can be the one made by the manufacturer and embedded in the device (i.e. Samsung), the one embedded in the operating system flash oriented (i.e. YAFFS) or can be made from a flash manufacturer as port for specific operating system like Unistore II made by Samsung for Symbian OS (Morris, 2007, Samsung, 2006b). For more info on algorithms and data structures see (Gal and Toledo, 2005).

Coming back to UBIFS, it is a new flash file system developed by Nokia engineers with help of the University of Szeged and may be considered as the next generation of the JFFS2 file-system (MTD_group, 2008).

Wear Levelling (WL) and Garbage Collection (GL)

When data in memory flash are updated, it is not possible to program the same page for the one-way programming peculiarity of flash devices, so the page containing the to-be-updated data is entirely rewritten to a new location (could be or not the same block). In the spare area, the page with new data is marked as valid (live), while the old one is marked as invalid (dead). When the number of dead pages in a block is more than a given clearance than all live pages are rewritten to new locations and the block erased to allow future programming: this is an underground process called Reclaim of Garbage Collection and it is activated without user involvement and at not fixed time (Tsai et al., 2006).

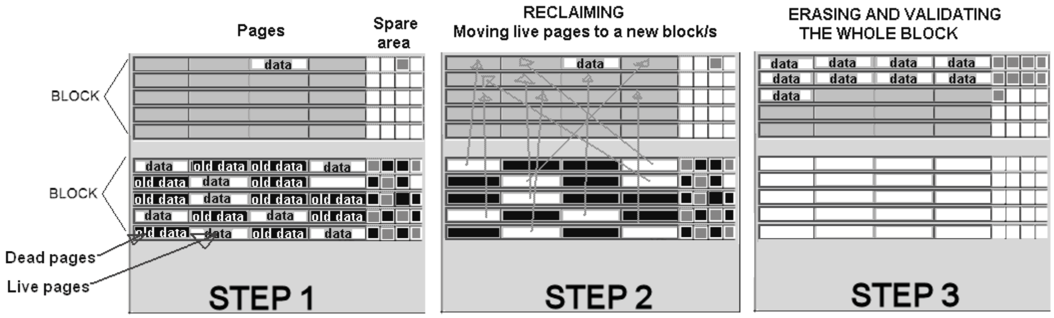


Fig. 12 The Reclaim process as part of the Wear Levelling policy

Note: in the example above, are used only two blocks but in the real world reclaiming could involve more blocks

To avoid excessive usury of same area despite others, a process called Wear Levelling manages blocks so that they are wisely used: there is a static wear levelling and a dynamic wear levelling, both attempts to extend lifetime of flash (Numonyx, 2008c, Jones, 2008). Wear levelling procedure can be embedded in the firmware of memory flash or left under care of host file system (Numonyx, 2008b, Numonyx, 2008c, Jones, 2008, JI et al., 2009).

Data in the invalid blocks or dead pages can store information of interest for the forensic analyst and should be acquired before Reclaim take place: analysts are asked not to alter the state of the evidence, but as Wear Levelling and Reclaim are underground processes this requirement can be hard to achieve and difficult to manage. In future works will be examined the effect of Reclaim in embedded devices: outcomes will be reported.

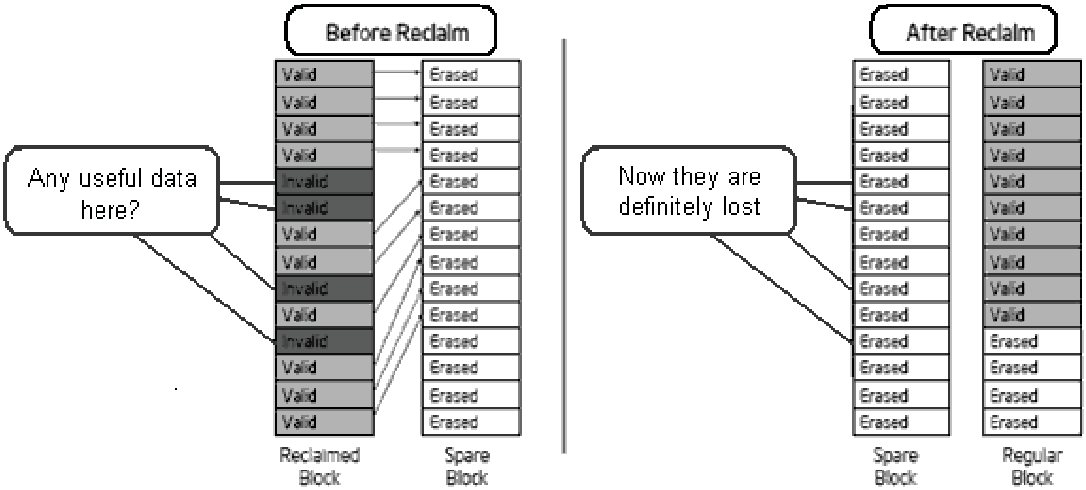


Fig. 13 The state of blocks before and after Reclaim (Intel, 2006)

Error Correction Code (ECC)

A page can be *programmed*, *erased* and *read*; after each operation is it necessary verify the status of the page. To perform this verification, flash devices use a verification algorithm that produces a sort of hash/CRC value for each accessed page: the value is then stored in the spare area (Numonyx, 2008d). This algorithm is generally referred as the Error Correction Code. If a bit error is detected after the read phase, it can be recovered by ECC; if the error is detected after programming or erasing cycle then a block replacement policy is activated (Micron, 2006a, Samsung, 1999). For further information on ECC, see also (Samsung, 2004). Unlike Wear Levelling, ECC logic is generally embedded in the firmware of all flash memories.

Even ECC algorithms are trade secrets, some hacking solutions are able to rewrite data in the flash device reconstructing the ECC (like the code present in Sony PlayStation 3 (NDT, 2008)). This is a new frontier of illegal activities, not covered here.

Bad Block Management (BBM)

If ECC reports a non recoverable error, it is required that area be marked as bad. Since the smallest erasable area unit is the block, for any unrecoverable error arising in any page, the whole block to which the page belongs will be invalidated requiring the replacement of such block, so it will not accessed again (Samsung, 2006b). Bad blocks identified during NAND lifecycle will be added to the list of bad blocks generated during factory production, and should not exceed 2% of the total number of blocks (Samsung, 2007, STMicroelectronics, 2004).

To manage invalid blocks, manufacturers do not share a unique rule, but refer to two replacement strategies: Skip Bad Block (SBB) and Reserve Block Area (RBA). In the SBB, when a bad block is detected the flash filesystem simply skips ahead to the next good block. In the RBA strategy, a predetermined area devoted as reservoir, is used to supply good blocks as replacement for the bad.

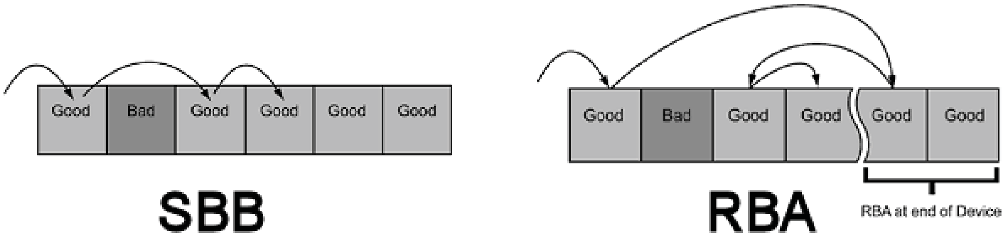


Fig. 14 Skip Bad Block (left) vs. Reserve Block Area replacement strategy (right) (BPMicrosystems, 2008)

Skip Bad Block strategy and related issues

The SBB can causes a shift between physical and logical arrangement of data in flash device with more than one LUN. SSB could also lead to a block encroachment where a block from a partition (B) is retrieved to be at service of a previous and contiguous partition (A). That is, it will be possible to have two blocks with the same number (BPMicrosystems, 2008, Breeuwsma et al., 2007).

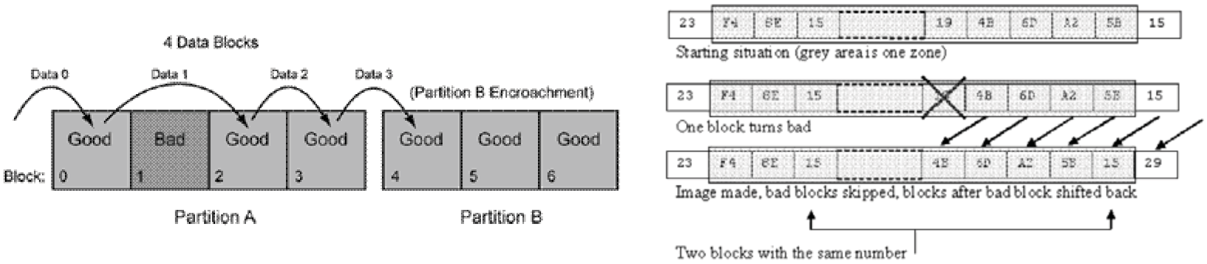


Fig. 15 Block encroachment (left) and Block number duplication (right) (ibid)

Reserve Block Area strategy and related issues

When utilizing RBA, partitioning of data is not done and the device is simply divided into user block area and reserve block area (BPMicrosystems, 2008, Samsung, 2006a). A proprietary table is used to map bad blocks to the RBA. In case the table gets lost, it should be possible to reconstruct a new one by reading flags in the spare area of all blocks – even if some authors think this is an extremely difficult task (Inoue and Wong, 2004).

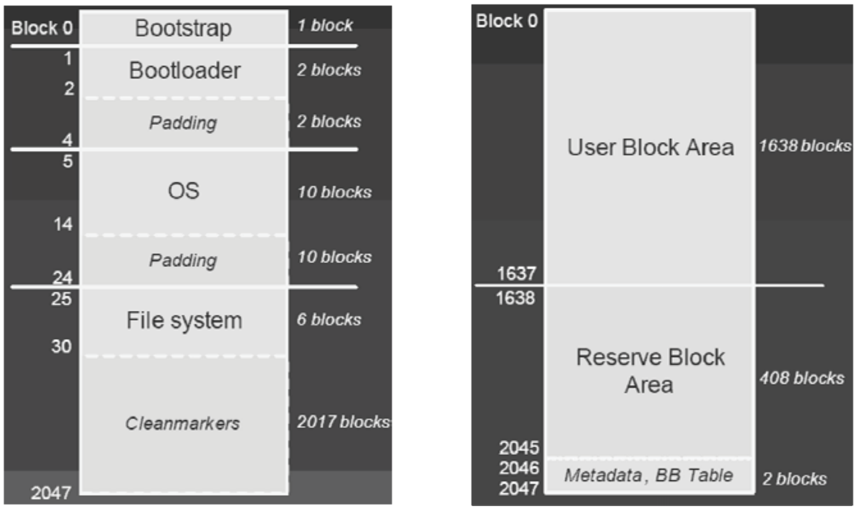


Fig. 16 Partitioning for Skip Bad Blocks (left) and Reserve Block Area (right) (White, 2008)

Raw NAND and Managed NAND[2].

When the FTL logic and relative functions are embedded in the NAND, then the flash is categorized as *managed NAND*, while when FTL is under care of host filesystem (the logic is external to the NAND) then the flash is said *raw NAND.* Raw NAND contains just the flash memory array and a Program/Erase/Read (P/E/R) controller *(Pon et al., 2007)*. For forensic analysis, it is fundamental considering difference between *raw* and *managed* NAND, with particular regard to effects of reclaim and bad block management.

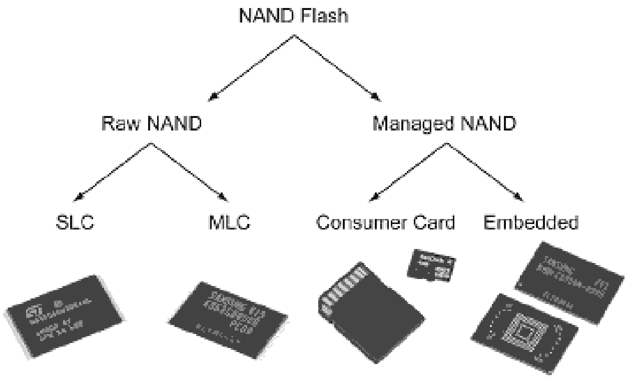


Fig. 17 NAND Flash type categories (BPM).

Evolution of flash memory: the Samsung OneNAND™

On 2003, Samsung developed a new unified flash memory device for code and data storage: the OneNAND™. This device has both high-speed data read function of NOR Flash and high speed write capability of NAND Flash. At date of writing the data storage capability of NAND area is up to 16 Gb. OneNAND has a NOR interface, so the chipset detects the OneNAND™ as NOR, while the data can be stored directly in the NAND area using multiplexed access lines. OneNAND™ is classified as a raw NAND with internal ECC capabilities (Samsung, 2005b).

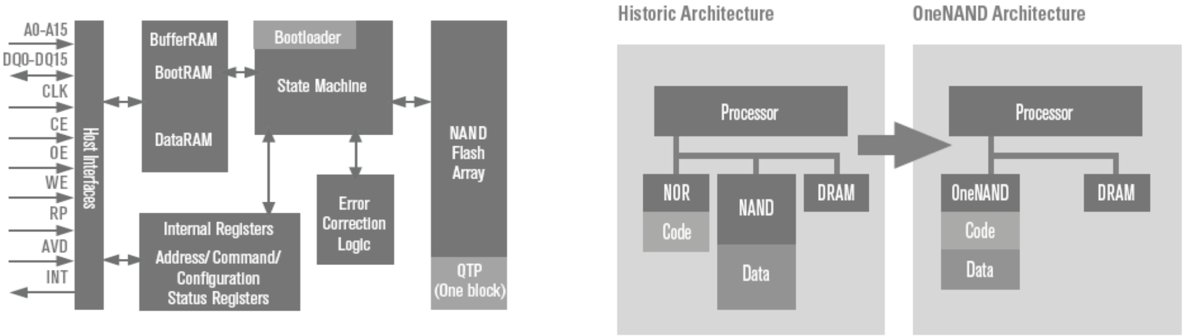


Fig. 18 OneNAND™ layout (left) and Historic vs OneNAND™ architecture(right) (Samsung)

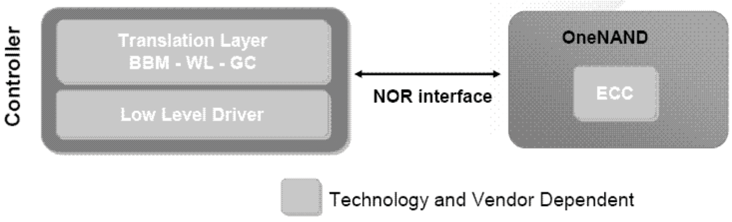


Fig. 19 Raw OneNAND™ (Numonyx, 2009)

[2] Usually this chapter is set at beginning of any flash document: the reason it has been set here is due to author's opinion that the point of view of engineers is not the same of evidences analysts

PART TWO: FLASH MEMORY FORENSIC

A brief digression on evidence metrics

Considering a digital device as body of evidence, it is possible to define some statements:

E as the full set of evidences **E**xisting on the device

A as the set of evidences **A**cquired by forensic tools (i.e. dd)

O as the set of evidences **O**bserved (found) by the analysts

so that:

Y is the ratio between **A**cquired evidences and **E**xisting evidences [A/E=**Y**] and represents the quality of forensic tools used (1=better, 0=worse);

K is the ratio between **O**bserved evidences and **A**cquired evidences [O/A=**K**] and represents the analyst's skill (1=better, 0=worse);

Z is the ratio between **O**bserved evidences and **E**xisting evidences [O/E=**Z**] and represents the overall quality of analysis (1=better, 0=worse).

Units of evidences			**Y** (A/E) (tool quality)	**K** (O/A) (analyst skill)	**Z** (O/E) (overall quality of analysis)
Existing (**E**)	Acquired (**A**)	Observed (**O**)			
100	100	100	**1**	**1**	**1**
100	80	80	0,8	**1**	0,8
100	80	60	0,8	0,75	0,6

Tab. 1 Quantitative relation between evidences, analyst's skill, and quality of tools

Thus, a good tool with a good analyst gives an overall good analysis (case 1), a mediocre tool (case 2) or a mediocre analyst (case 3) will limit the overall value of examination. Of course this is just a quantitative and not qualitative measurement: the importance of each evidence is set aside.

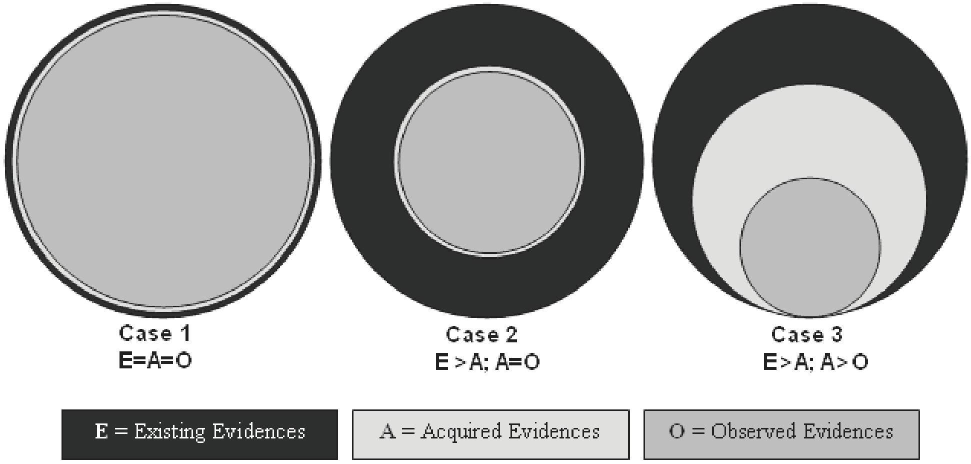


Fig. 20 Quantitative relation between evidences, analyst's skill, and quality of tools

Logical vs Physical acquisition

Logical and physical acquisitions are already well defined in the NIST Special Publication 800-101 Guidelines on Cell Phone Forensics (Jansen and Ayers, 2007):

> *"Forensic tools acquire data from a device in one of two ways: physical acquisition or logical acquisition. Physical acquisition implies a bit-by-bit copy of an entire physical store (e.g., a memory chip), while logical acquisition implies a bit-by-bit copy of logical storage objects (e.g., directories and files) that reside on a logical store (e.g., a file system partition). The difference lies in the distinction between memory as seen by a process through the operating system facilities (i.e., a logical view), versus memory as seen in raw form by the processor and other related hardware components (i.e., a physical view).*
>
> *Physical acquisition has advantages over logical acquisition, since it allows deleted files and any data remnants present (e.g., in unallocated memory or file system space) to be examined, which otherwise would go unaccounted."*

In the image below is given a representation of both methods, in case of memory not physically extracted from hosting device, that is, left on the phone and accessed with traditional means.

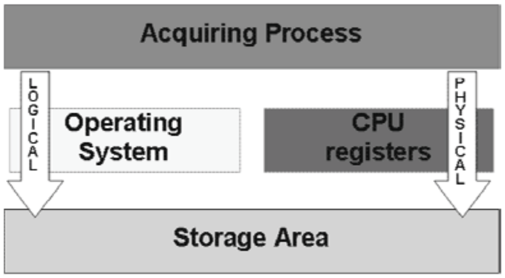


Fig. 21 Logical vs. Physical acquisition for flash memory on the hosting device (not extracted)

Proprietary cables with USB interface are used for both techniques, while JTAG or FBUS interfaces (where present) are mainly used for physical acquisition; it is also possible get data via infrared and Bluetooth interface using OBEX protocol, but this is a method that poses some limitation and is generally less used (McCarthy, 2005). Some Nokia phones are now explored: registry addresses are blurred for confidentiality.

Flash peculiarities in the acquisition process

During this research it comes out the high level of confidentiality surrounding the flash technologies and market, so that nobody seems to be able to set a definitive point on how others can use or implement flash technologies: a problem reported since the begin of mobile forensic (Willassen, 2003). In an attempt to understand better what really happen inside a flash memory, were organized some meetings with skilled people working in the flash manufacturing field and the focus was set on how to preserve integrity of evidence and grant completeness of acquisition. This is what came out:

Real effect of reclaim:

garbage collection is a known activity but not so well documented for seized devices

garbage collection is a background activity, this means that when a mobile phone is powered on, even in service mode, such activity *could be* autonomously triggered with the effect of destroying useful data in invalid blocks

Effective management of bad blocks:

if the FTL is embedded in the flash memory (like in case of managed flash) then it will be difficult to access and manage bad blocks because they will be hided to the host file system;

if the FTL is supplied from the host (like in case of raw flash) then there are chances to manage bad blocks properly and have direct access to them. Analogous experiences are reported with modern hard disks managed with GNU ddrescue3 (Carrier, 2005, Lyle and Wozar, 2007, Mukasey et al., 2008).

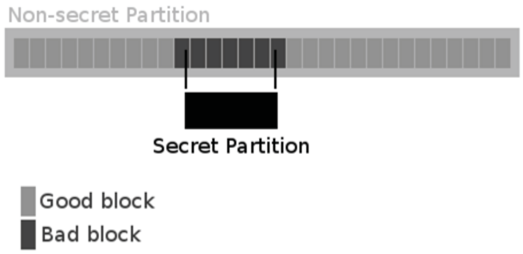
[3] There is still an open debate on hard disk bad block management. Some interesting links are: http://tech.groups.Yahoo. com/group/ForensicAnalysis/message/82,

Security through obscurity

Even knowing the memory specs, manufacturers can apply autonomous decisions on how manage the chip: it can happen that a managed flash will be used with disabled features, or that a flash raw memory be customized as for manufacturer needs. Furthermore, due to high competition and Intellectual Property protection, generally, there are not public information on the chip used. At begin of the research some manufacturers were contacted to get some info: it was even difficult to know the destination of some branded components.

Bad management of good blocks

A block is considered bad when there are multiple bit errors that are not recoverable (Numonyx, 2008a). Like hard disks, NAND flash generally ships with a list of existing bad blocks set in a location defined by the manufacturer. Additionally, to this list will be added all future blocks will fail to operate during device lifecycle. Forensic investigators are already aware of the possibility to manipulate Bad Block List to hide information (David, 2009) this aspect should not be underestimated in flash memories as they are able to store even larger quantity of data: a working OS could be as small as 50 MB (www.damnsmalllinux.org) or much less with Embedian distro (www.emdebian.org).



Hiding data in bad blocks (David, 2009)

Misuse of Hidden Protected Area

It could be possible for an hacker to store data even in the Hidden Protected Area also referred as One Time Programming (Samsung, 2007a). The size of this area is generally equal to one block but variants are allowed (Samsung, 2005c, Micron, 2006c); it can be blocked, but usually this task is left under hosting manufacturer care (ibid)..

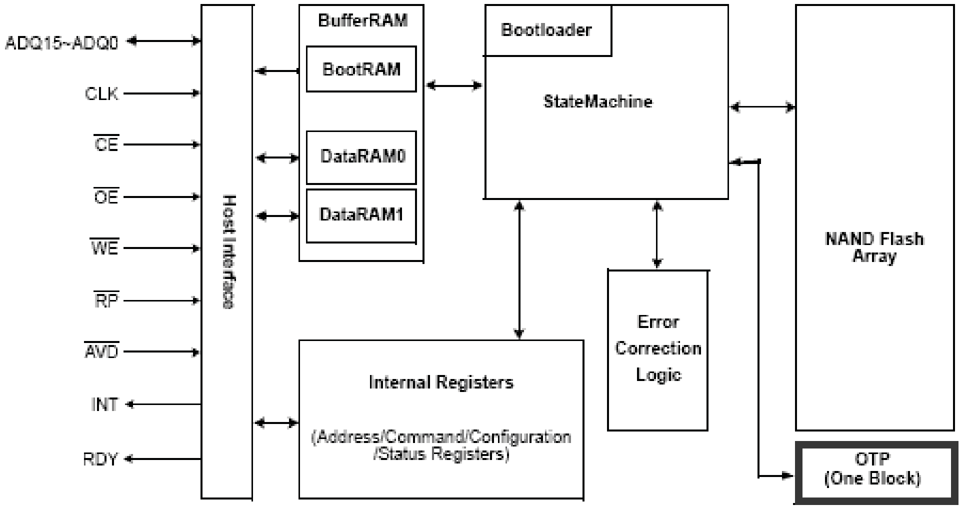


Fig. 22 Block Diagram on a multiplexed OneNAND™

Computer analysts already know the issue related to Host Protected Areas (HPA) and Device Configuration Overlays (DCO) in hard drives (Gupta et al., 2006, Carrier, 2005): with flash memories we have similar issues. In future works we plan to test the possibility to change (doubling) the dimension of such area and then to store and hide data in it.

HOW THE CHOICE OF THE FLASH MEMORY AND MOBILE PHONE WAS DRIVEN AND THE TEAM WAS SET

Simply, the choice of mobile phone and flash memory to use was made by statistics. Nokia is the best seller in the mobile phone market and Samsung is the leader in the NAND flash market

Market Share (%)

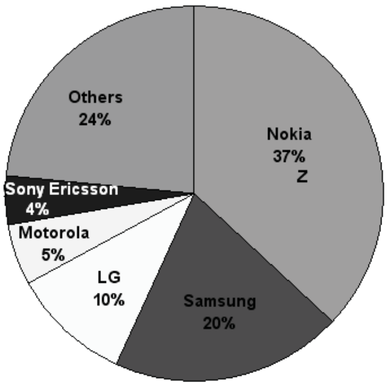


Fig. 23 Worldwide Mobile Terminal Sales to End Users in 2Q09 (Gartner, 2009)

2008 NAND flash brand sales breakdown

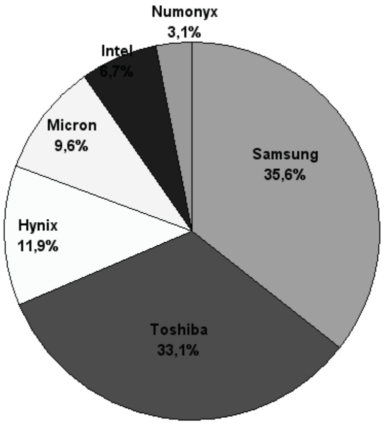
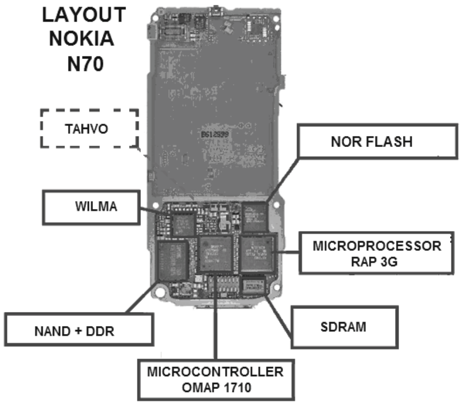


Fig. 24 4Q08 NAND Flash brand sales break down (DRAMeXchange, 2009)

Then the choice to use an OneNAND was made for its advanced characteristics and the Nokia model was chosen on the basis of a block of ten OneNAND available at moment. Numonyx has licensing agreement with Samsung to produce OneNAND™, so it was decided to call Numonyx for support and the folks there were happy to help. Then, was asked support to an advanced Nokia service repair centre that was willing to help, too: in few days a virtual team with high skilled people was s and ready to start. As this market is so hard-hitting, a low profile participation has been adopted.

How NOR and NAND are accessed on a Nokia N70

The implementation layout of NOR and NAND chips in a Nokia mobile phone (N70 model), is presented in the picture below (left). The combo memory (NAND+SDRAM) flash is managed by a TI microcontroller unit (mcu) OMAP 1710. OMAP stands for Open Multimedia Application Platform and it is the application processor running with Symbian operating system (EPOC). The NOR flash is managed by the microprocessor RAP3G (3G Radio Application Processor). Evidences on mobile phone are stored in NAND flash: whatever means are used, to access the NAND storage area it is required to move through the OMAP processor (right).



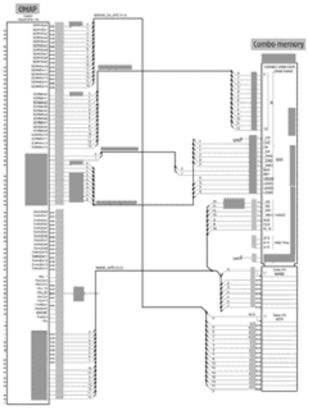
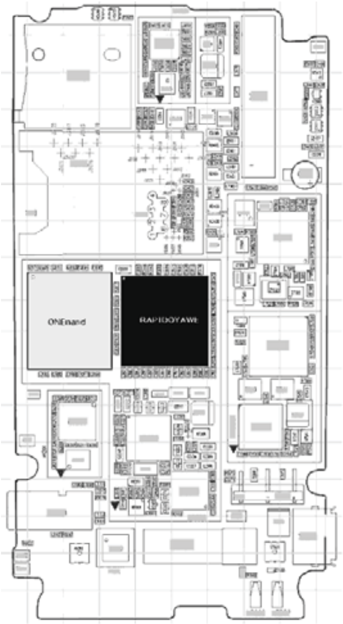
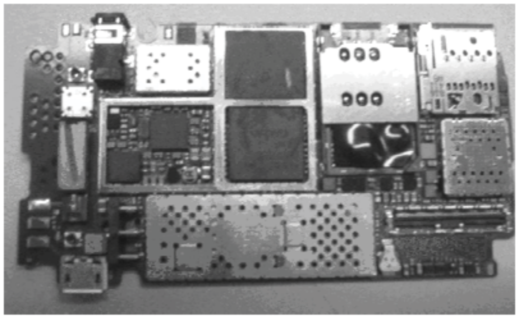


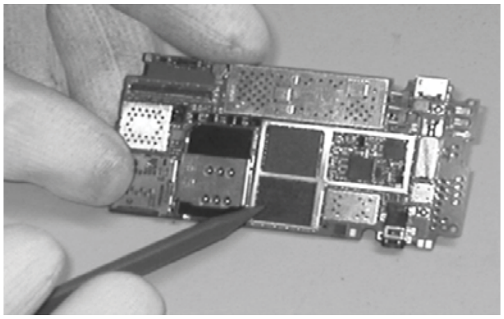
Fig. 25 Layout of a Nokia N70 (left), and OMAP and NAND flash relation on Nokia N70 (right)

How OneNAND™ is accessed on a Nokia 6650F

The Nokia 6650F phone has been introduced on the market on 2008. The application memory of the device consists of NAND/DDR combo memory. The stacked DDR/NAND application memory has 512 Mbit of DDR memory and 1024 Mbit of flash memory[4].   This is the phone we have chosen to be used for tests presented later: on the left the phone schematic, then two picture of the internal side (with indication of the OneNAND™), the relation between processor and flash memory and flash memory pins layout. Larger images are available in appendices.







[4] 1024 Mb are equal to 128 MB.

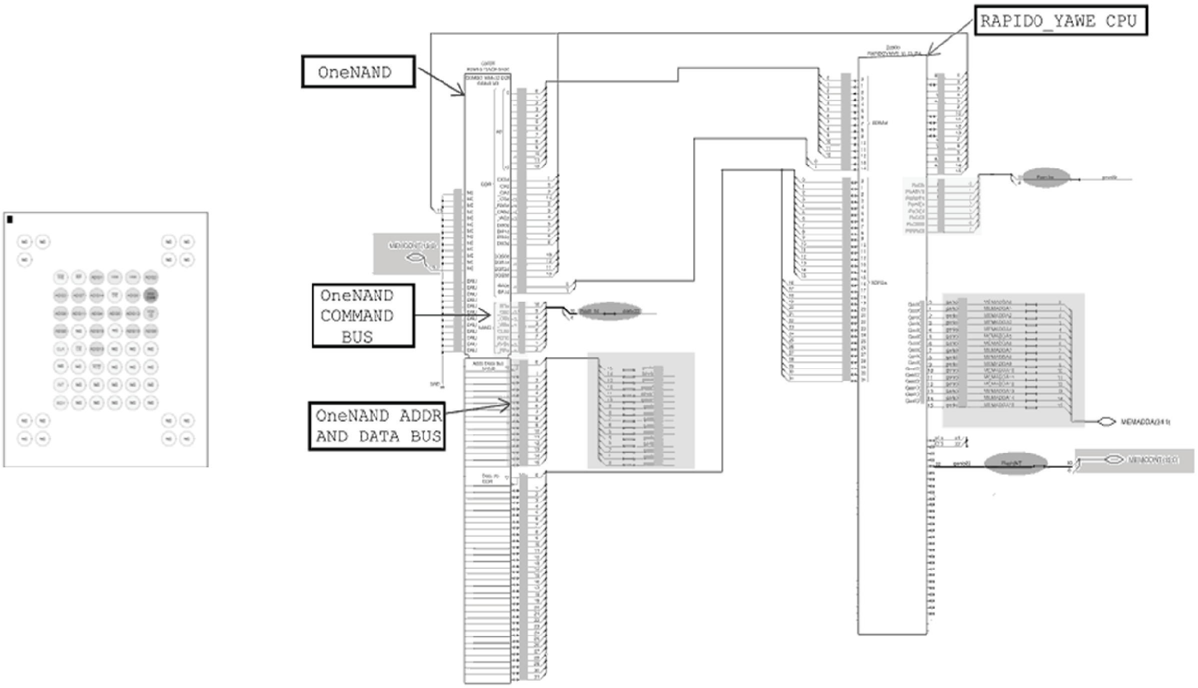


Fig. 26 From left to right (clockwise): Nokia 6650F layout; the internal hardware, stencil pointing at the OneNAND™ flash; schematic showing connections between CPU and OneNAND™, and generic OneNAND™ pins layout.

How data on NAND are accessed via USB or JTAG on a Nokia 6120c

To perform a memory dump of the flash memory via physical acquisition on a Nokia 6120c, either with a USB cable or a FBUS/JTAG interface, it is required processor involvement (in this case it is a RapidoYawe[5]). In the tables below are presented schematics of connections between two devices (memory and processor). This phone will replace the Nokia 6650F in our tests, as explained later: the layout is very similar. Larger images are available in appendices.

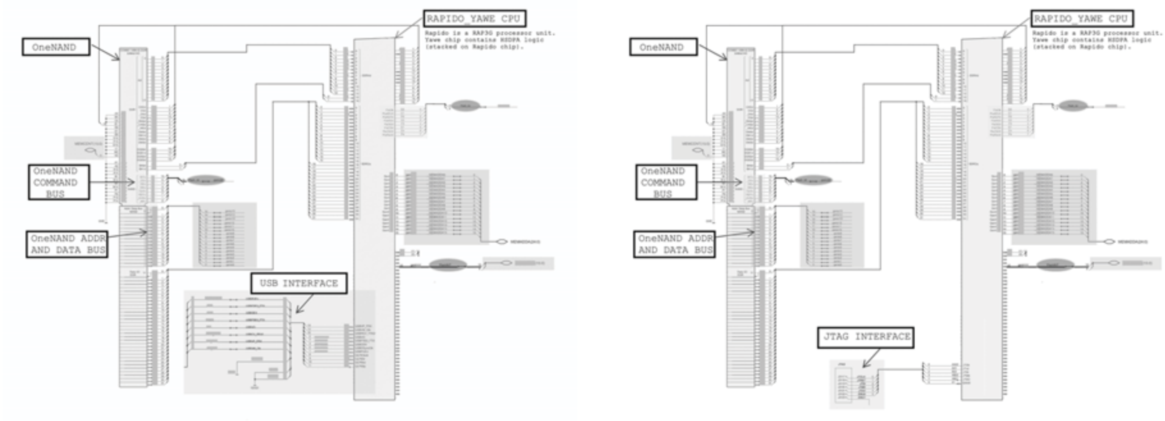

RAPIDO_YAWE CPU
Rapido is a RAP3G processor unit. Yawe chip contains HSDPA logic (stacked on Rapido chip).
OneNAND
OneNAND COMMAND BUS
OneNAND ADDR AND DATA BUS
USB INTERFACE
RAPIDO_YAWE CPU
Rapido is a RAP3G processor unit. Yawe chip contains HSDPA logic (stacked on Rapido chip).
OneNAND
OneNAND COMMAND BUS
OneNAND ADDR AND DATA BUS
JTAG INTERFACE

Fig. 27 Adapted layout of access to NAND memory via USB (left) or JTAG (right)

Test Phase 1: preparing the phone

[5] The chip with HSDPA logic (YAWE) stacked on the RAP3G processor unit (RAPIDO) forms the RapidoYawe CPU

On a new flash memory (identical to the one on the testing Nokia mobile phone) were stored some data in four good blocks; such blocks were then marked as bad, by opportunely manipulating the relative spare area. Next, the original flash device embedded in the phone was replaced with the one with four bad blocks and the phone refurbished with original software: now, there is a working phone with data hided in bad blocks. The detailed procedure is in the appendices.

Test Phase 2. Feeding forensic tools with our phones: results and feedbacks.

At beginning, when decision on which type of phone to use was made, it was considered an advantage to use a Nokia phone, due to its popularity. Not too much attention was paid on the specific model we were using: all in all there was an OneNAND™ inside and this was considered an advantage for the research. As the testing memory was a raw NAND, we were optimist forensic software would be able to acquire bad blocks because there were not embedded FTL layer could interfere with the imaging process.

Then, we used some of the best forensic software to test the acquisition of bad blocks from our phones, and this is what we got (in alphabetical order).

CelleBrite UFED.

This solution was not able to perform the physical acquisition.

Logicube CellDEK

We were not able to perform any acquisition with CellDEK because the required module, even already ordered, was not available at time of examination.

Micro Systemation XACT

This solution was not able to perform the physical acquisition.

Paraben Device Seizure 3.1

This solution was not able to perform the physical acquisition.

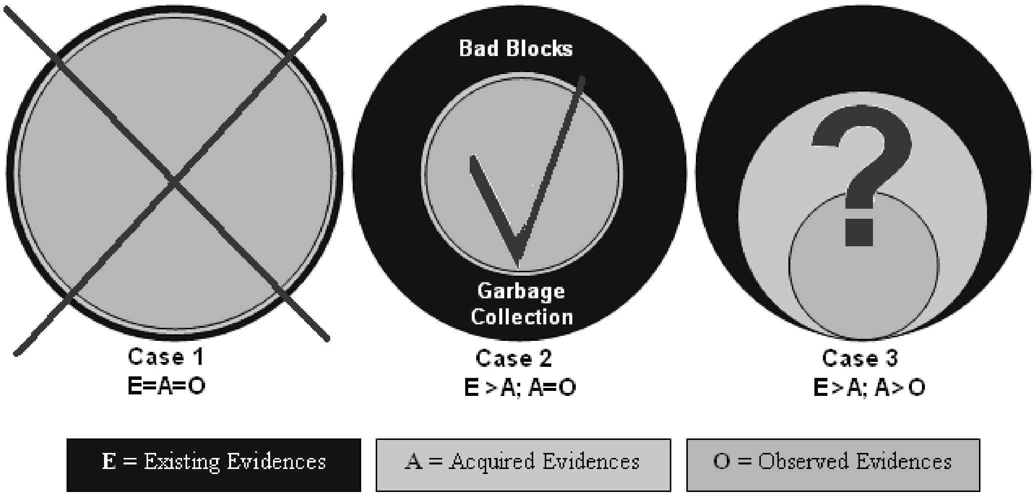
At this stage, was decided to speak directly with technical support of these companies and tell them the problem we faced. An email was sent either to companies aforementioned and to others that have been tested their products with NIST (as reported in the CFTT web page http://www.cftt.nist.gov/mobile_devices.htm). The test of the emails is reported in appendices. So far, these are the replies we got:

CelleBrite, Micro Systemation and Paraben confirmed the inability of their solution to get physical acquisition of our phone (even they can do with others); Guidance Software, Logicube, and Susteen did not reply.

For what we tested and understood, with these solutions and the phone we used, if sensitive data are hided in bad blocks they will go undetected. Furthermore, with this software, good blocks with wrong ECC (i.e. due to power failure) could hide valid data to forensic analyst.

Reporting to forensic metrics

Our test take a lot of time to be set and only few minutes to be waived: we were a little disappointed. Going back to evidences metrics seen before, we should say that any forensic tool not able to deal with bad blocks (completeness of evidence) should fall at least in the case number two. This without considering underground Reclaim activities, yet (the effect of Reclaim on integrity of evidence need further analysis).



Quantitative relation between existing evidences, quality of tools, and analyst's skill

Physical acquisition as option: what says the NIST.

Many companies are proud to say their products have been successfully tested with NIST, but what exactly say a NIST report on mobile physical acquisition and completeness of evidences acquired?

A first answer can be found either in the version 1.1 (NIST, 2008) or 1.2 (NIST, 2009) of *GSM Mobile Device and Associated Media Tool Specification and Test Plan,* where is reported in the section CFT-IMO-05/06 and CFT-IMO-04, respectively, that physical acquisition is an optional feature. For analyst with hard disk forensic background, it could seem a little strange considering physical acquisition an *option*.

Furthermore, the word *completeness* is reported in the *2004 Digital Data Acquisition Tool Specification*, in the *2005 Digital Data Acquisition Tool Test Assertions and Test Plan Draft 1 for public comment Version 1.0*, in the *2008 GSM Mobile Device and Associated Media Tool Specification and Test Plan (ver 1.1)* but not in the *GSM Mobile Device and Associated Media Tool 3 Specification and Test Plan (ver 1.2)*: the question is why completeness of evidences is then shifted to be an optional feature. The NIST were contacted either at institutional and authors' addresses (email in appendices). This is the synthesis of answers got - the source asked not to be cited, but to refer to CFTF site

> Optional test cases are treated as Core test cases IF the tool provides the capability defined by the test case. Unfortunately, all mobile forensic tools do not have the ability to perform a physical acquisition at this time. The CFTT formal testing methodology validates that tools perform as they are designed not as one might wish them to.
>
> Physical Acquisition is not an unreachable limit, but some tools are designed only for logical acquisitions. The specification and test plan state that if the tool provides the functionality optional cases and assertions are tested as if they are core. By following the CFTT formal testing methodology it allows all tools that have the ability to acquire data from mobile devices to receive a fair validation.

The aim of this paper is not to argue with NIST, but for what is written in the second sentence above, test on tools designed either for logical and physical acquisition, like Cellebrite UFED 1.1.05, should report physical acquisition in the core features: but by reading *Test Results for Mobile Device Acquisition Tool: Cellebrite UFED 1.1.05* it is possible to see that physical acquisitions is reported in the CFT–IMO–05 section, as an optional feature.
In the email sent to NIST, author suggests to shift this feature from optional to core section, because a document released from so regarded source, should not allow a workaround of an important point like this.

A confidential answer

We asked to forensic software houses cited above, why it is so difficult to perform a physical acquisition of non-volatile memory[6] embedded in phones made by different manufacturers but using the same raw flash memory and the same I/O interface. This is the answer got from a source asked not to be disclosed:

- IP protection: many phone manufacturers need to protect their know-how, so they encrypt some area of the memory and use proprietary bootloading solutions. This means that a forensic software house should be able to decrypt, without altering, the content of the evidence and also it need do this for any mobile phone on the market: a very onerous task that in the lack of a collaboration between chip manufacturer and software developers is too uneconomical. When a flasher is used to change IMEI or unlock a phone it exactly circumvents this protection (for this, the source states further that in future mobile phones, JTAG interface will be disabled to prevent illegal activities).
- Market alliance: for reasons seen above, forensic solution providers could not have interest to release something harmful for phone manufacturers because otherwise the latter will not be anymore cooperative with them.

The ONFI project

The resolve the problem of disorder in the flash market, some manufacturers decided to setup a consortium to define some standards: it is the Open NAND Flash Interface (ONFI) consortium. "*The ONFI is an industry Workgroup made up of more than 80 companies that build, design-in, or enable NAND Flash memory, dedicated to simplifying NAND Flash integration into consumer electronic products, computing platforms, and any other application that requires solid state mass storage. We define standardized component-level interface specifications as well as connector and module form factor specifications for NAND Flash"(http://onfi.org).*

FUTURE WORKS

We plan to do some feature works especially to test the effect of reclaim in a controlled environment (like a mobile phone left in standby), and capture (by sniffing) and analysis of data travelling on the bus to/from mcu and NAND. Results will be reported to community.

CONCLUSION

In this paper has been attempted to offer a wide overview of forensic analysis of non-volatile flash memory.

Starting from academic and industrial literature, we ended with a practical and documented test in which some data were hided in memory blocks (then marked as bad) to verify if it was possible to foul the acquisition process of nowadays forensic solutions. It was demonstrated that hiding data in such blocks is achievable: none of the software tested was able to get a physical acquisition of the flash memory.

Furthermore a suggestion to considerer physical acquisition a core feature was sent to the NIST to make them more aware of the problem of data hiding in flash memories and the need to grant the completeness of evidence.

COPYRIGHT

[6] We should not forget that on OneNAND we have both volatile memory (DDR) and non-volatile memory (NAND).

EMAIL SENT TO MOBILE FORENSIC SOLUTIONS

TO: [contact-name]@paraben.com; [contact-name]@susteen.com; [contact-name]@msab.com; [contact-name]@guidancesoftware.com; [contact-name]@cellebrite.com; [contact-name]@logicube.com

Object: Conference paper on possible limits of your forensic tools

To whom it may concern,
my name is Salvatore Fiorillo and I am writing a forensic paper to be presented at 7th Australian Digital Forensics Conference in Perth WA on December 1-3, 2009.

The paper focuses on flash memories, especially NAND and OneNAND: in a Nokia 6120c mobile phones I have replaced the orginal 1Gb OneNAND flash memory with an equivalent memory on which I had previosuly marked four blocks as bad and stored some data in them. The aim is to verify the capability of forensic software to acquire data from such fake bad blocks.

For what I understood it is not possible to acquire data from such blocks because the software you produce in not able to perform a physical acquisition of the OneNAND either on the mobile phone I am using and on many others: where available to public, I have verified this information with the list of supported models.

I will report these results: if you think I am wrong, please send me, with any possible urgency, your opinion and possibly some facts I could present.

This email and your answer, if any, will be documented in the paper.

Best regards

Salvatore Fiorillo

AN EXTRACT OF ANSWERS SO FAR RECEIVED:

From: [contact-name]@cellebrite.com

Dear Salvatore,
I have attached the list of the UFED Physical supported model list (see the Physical dump column).

As you can see we support about 395 models with Physical dump and this includes manu models with the OneNAND flash memory that we extract completely including the spare areas.

So your conclusion that the UFED Physical is not able to perform a physical acquisition is not correct.

In specific, in the supported phone list, you can see a list of the Samsung models that are supported and most use a OneNAND flash memory that we acquire and decode. We are doing this by using our own boot loaders (unlike other tools that use unknown boot loaders that limit their capabilities) that were custom made for forensic acquisition.

UFED Physical does not support a physical dump for the Nokia 6120c at this stage.

Thanks,

Ron

From: [contact-name]@msab.com

Hi Salvatore,

I have checked regarding your question and the particular phone, Nokia 6120c is supported in our product .XRY (logical acquisition) but not yet in XACT (physical acquisition). We do not have a problem with physical extraction of the OneNAND flash memory, but that this phone is not supported in XACT.

Hope this information helps!

Kind Regards,

Maria

From: [contact-name]@paraben.com

Salvatore,

Some of what you are asking below is not exactly clear so I will do my best to get you some base information. With the scenario you are describing I don't believe any software will be able to read that memory. We can get a physical image of memory off of a Nokia phone, but the issue is that with what you are wanting to read the only way to do that low level of access to OneNAND memory for a Nokia device is to physically remove the chip and read the chip itself.

--Amber

EMAIL SENT TO CFTT

TO: cftt@nist.gov; richard.ayers@nist.gov; wjansen@nist.gov; Karen.scarfone@nist.gov

Object: flash memory mobile forensic and tool specification.

Dear all,

I am writing you to suggest a revision of *"GSM Mobile Device and Associated Media Tool Specification and Test Plan",* section CFT-IMO-04.

Please, let me explain how I come to such request.

I am preparing a forensic paper on non-volatile memory (flash) embedded in mobile phone: the paper should be presented at 7th forensic conference in Perth. My aims were to (a) make a basic point in flash forensic field and (b) verify the possibility to retrieve data from fake bad blocks.

I used a Nokia mobile phone and OneNAND flash memory. In the latter were stored data in four good blocks and subsequently such blocks were marked as bad (I did it by disassembling, programming and then reassembling the chip from the PCB).

To retrieve hided data, were used some major forensic tools like CelleBrite UFED, Logicube CellDEK, Micro Systemation XACT and Paraben Device Seizure 3.1. I was not able to get data from bad blocks because none of software used were able to acquire a physical image of the testing memory.

A request of verification was send to software houses: CelleBrite and Micro Systemation confirmed the inability of their solution to get physical acquisition of that phone (even they can do with many others); Paraben will reply after the 13th, Guidance Software, Logicube, and Susteen did not reply.

In the paper is stated that with these solutions and the phone used, if sensitive data were hided in bad blocks they will go undetected. Now with phones equipped with even more increasing storage area, IMHO, this could led to a problem: how many data we can store in , say, 30% of 1 Gb of space? Should a forensic officer care of them?

So even I am conscious that it is hard to achieve and (at least in short term) not a feasible solution for all mobile phones, I suggest to move Physical Acquisition section (9.2.3) from Optional features to Core features. To say more, I have had further conversations with some lawyers, asking them if my outcomes could be used to invalidate (more or less) the value of evidence acquired. The answer was positive: not because evidence found on the mobile phone could be waived but because there is not assurance of completeness of acquisition. It is possible don't get an evidence that could discharge a person (so we have an innocent condemned) or don't get an evidence could charge a person (so we have a guilty set free). I suppose the second case have much more chances to happen.

With Regards

Salvatore Fiorillo

NOTE: The answers got from NIST/CFTT, as required, are kept confidential but a synthesis of their content is reported in the section dedicated to NIST (*Physical acquisition as option: what says the NIST*).

BAD BLOCKS MANAGEMENT AT MANUFACTURING TIME

Bad blocks can be present on new flash devices or arise during lifecycle. How bad blocks are managed in flash memory is something forensic investigators should be aware, mainly if they have to deal with physical acquisition (Breeuwsma et al., 2007).

Samsung

Samsung, with Unistore II handles bad blocks with the Block Management Layer (part of XSR) and a bad block mapping table : this scheme remaps a bad block to one of the reserved blocks so that the data contained in a bad block is not lost and new data writes on a bad block is avoided (Samsung, 2006b).

The eXtended Sector Remapper (XSR) adopts RBA and locates bad block info in the Reservoir area: that is, a zone located in the highest address region of the NAND flash (Samsung, 2008). Any block not presenting the FFFFh value in the 1st sector of the 1st and/or 2nd page in the spare area is to be considered a bad block

Once bad blocks have been located, Samsung, like others (BPMicrosystems, 2008) , recommends that the bad blocks be no longer accessed.

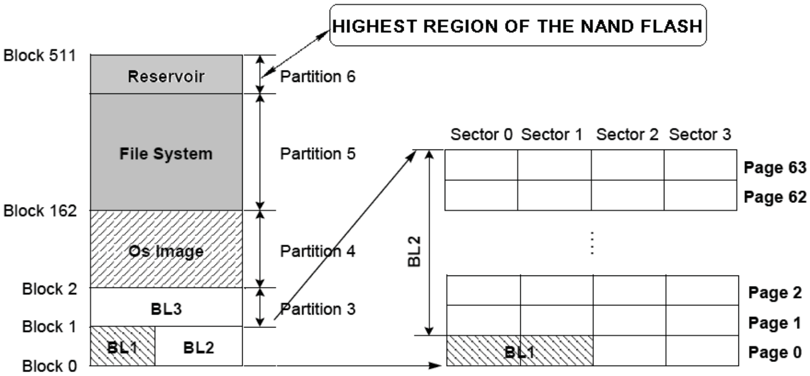


Fig. 28 The Reservoir area in Samsung flash memory (source: Samsung)

Toshiba

In Toshiba flash memory, the standard factory location for the information of the bad block byte is stored to byte 0x205 (byte 517 or the 518th byte) of a NAND page. Usually, if this byte is 00h, then the block is bad since the making of the device (factory default), if the byte is F0h then the block is gone bad during device lifecycle. Any block that is not all FFh (all 1s) in byte 517 (starting from byte 0) of the 1st page of a block is a bad block. Toshiba NAND TSOP devices can also use of a bad block table held in the first block of the NAND Flash Block 0. Toshiba recommends that the bad blocks no longer be accessed (Inoue and Wong, 2004).

Micron

Micron flash devices identify bad blocks by setting 1st byte of 1st and 2nd page to a value not 0xFF. Many different ECC implementations are available: to determine the level of ECC protection necessary, Micron policy is to refer to data sheet and requirements of the end system using the device (Micron, 2006b).

Numonyx

Numonyx flash memory uses the Hardware Adaptation Layer (HAL) software to manage Bad Blocks that develop during the lifetime of the NAND Flash device. Any block where the 6th Byte/ 1st Word in the spare area of the 1st or 2nd page (if the 1st page is Bad) does not contain FFh is a Bad Block. ST HAL can use either Skip Block Method or Reserve Block Method. For ST NAND Flash devices, the Reserved Area size is equal to 2% If a block becomes bad during the NAND Flash lifetime, the Bad Block Management software re-maps the Bad Block, and copies the data it contains to the block that will replace it (Numonyx, 2008e).

The Bad Block Table is created by reading all the spare areas in the NAND Flash memory and saving the table in a safe block (generally block 0) so that on rebooting the flash memory, the table is easy found in a known location. The blocks contained in the Bad Block Table are not addressable: if the Flash Translation Layer addresses one of the Bad Blocks, then the Bad Block Management software redirects it to a good block (ibid)

Test Phase 1: preparing the phone (full steps)

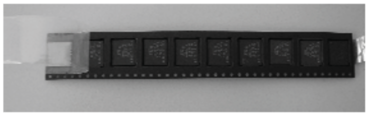
As wrote, the team was made involving folks from Germany, Italy and even Singapore. It was decided to adopt a strong documentation policy, NDA based, so in case something could get wrong, everybody could be able to help knowingly.

The following are the steps performed.

At first we defined which memory could be used: the choice was made for a 1Gb OneNAND™: it was considered one on the most used on mobile phones, and more interesting then simply NAND.

A stick of 10 OneNAND™ flash memories was send from Singapore.

The stick was send to lab to be programmed.



At lab, four OneNAND™ were programmed with a flash tester:

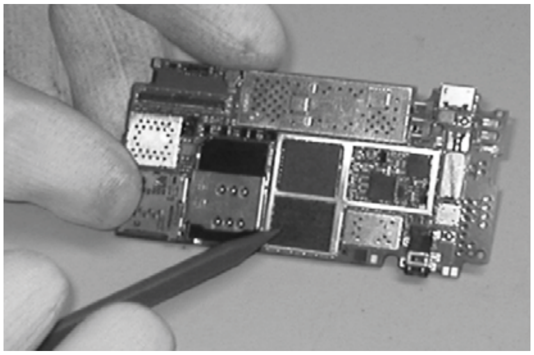
- four blocks were marked as bad in the respective spare area (blocks marked as bad were: 100, 125, 500 e 625)
- some text string was stored in that blocks

Then the memories were send to Nokia lab.

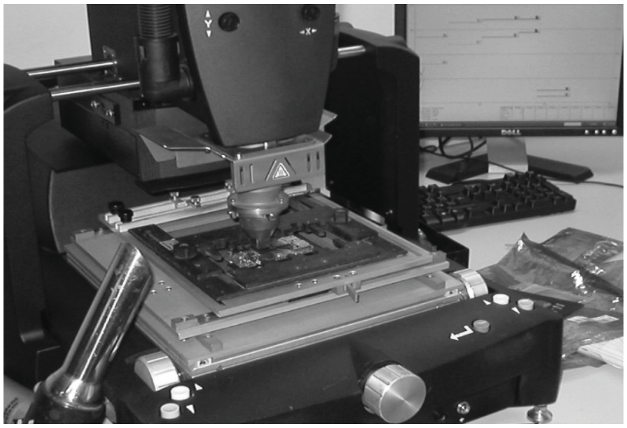
Four mobile phones Nokia 6150F were made available for tests.

After the failing of replacement of the first OneNAND flash on the 6150F, was decided to use four different phones, so we moved to two more manageable Nokia 6120C

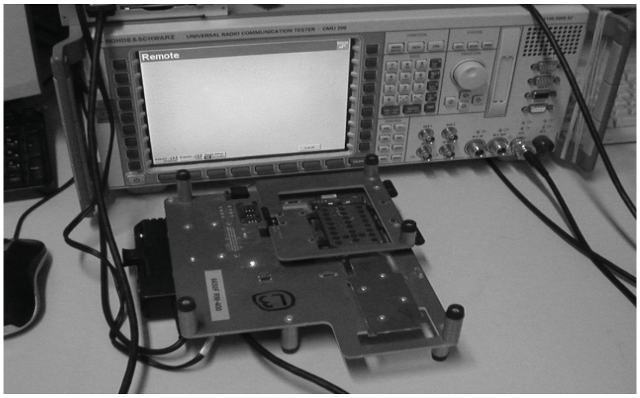
This two phones were disassembled, too and was made a control on the compliance OneNAND™ to be replaced



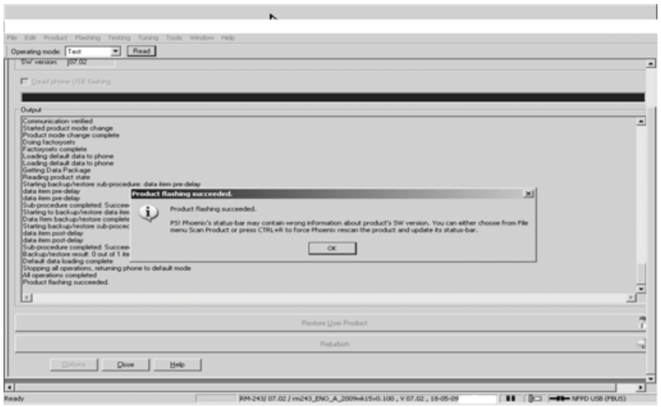
The original flash chips was then extracted using a reworking machine and with same procedure were implemented flash OneNAND™ with fake bad blocks.



The the state of the device was then verified using a jig interface



The phones were then refurbished and set at factory default



Operating mode: Test

SW version: 07.02

Output

```
Communication verified
Started product mode change
Product mode change complete
Doing factoysets
Factoysets complete
Loading default data to phone
Loading default data to phone
Getting Data Package
Reading product state
Starting backup/restore sub-procedure: data item pre-delay
data item pre-delay
data item pre-delay
Sub-procedure completed: Succee
Starting to backup/restore data item
Data item backup/restore complete
Starting backup/restore sub-procec
data item post-delay
data item post-delay
Sub-procedure completed: Succee
Backup/restore result: 0 out of 1 ite
Default data loading complete
Stopping all operations, returning phone to default mode
All operations completed
Product flashing succeeded.
```

Product flashing succeeded.

Product flashing succeeded.

PS! Phoenix's status-bar may contain wrong information about product's SW version. You can either choose from File menu Scan Product or press CTRL+R to force Phoenix rescan the product and update its status-bar.

Ready   RM-243/ 07.02 / rm243_ENO_A_2009wk15v0.100 , V 07.02 , 18-05-09   NFPD USB (FBUS)

Finally, logs were checked to verify correctness of jobs done

As expected, even after formatting the flash the four blocks were still there and correctly reported (full log in Appendices)

```
Target programming
BB Asic Index 00 (RAPx) programming succeeded
Device type is: 03
Device Index is: 00
Number of Bad Blocks: 4
Number of Initial Bad Blocks: 4
Number of run-time Bad Blocks: 0
Target programming completed
```

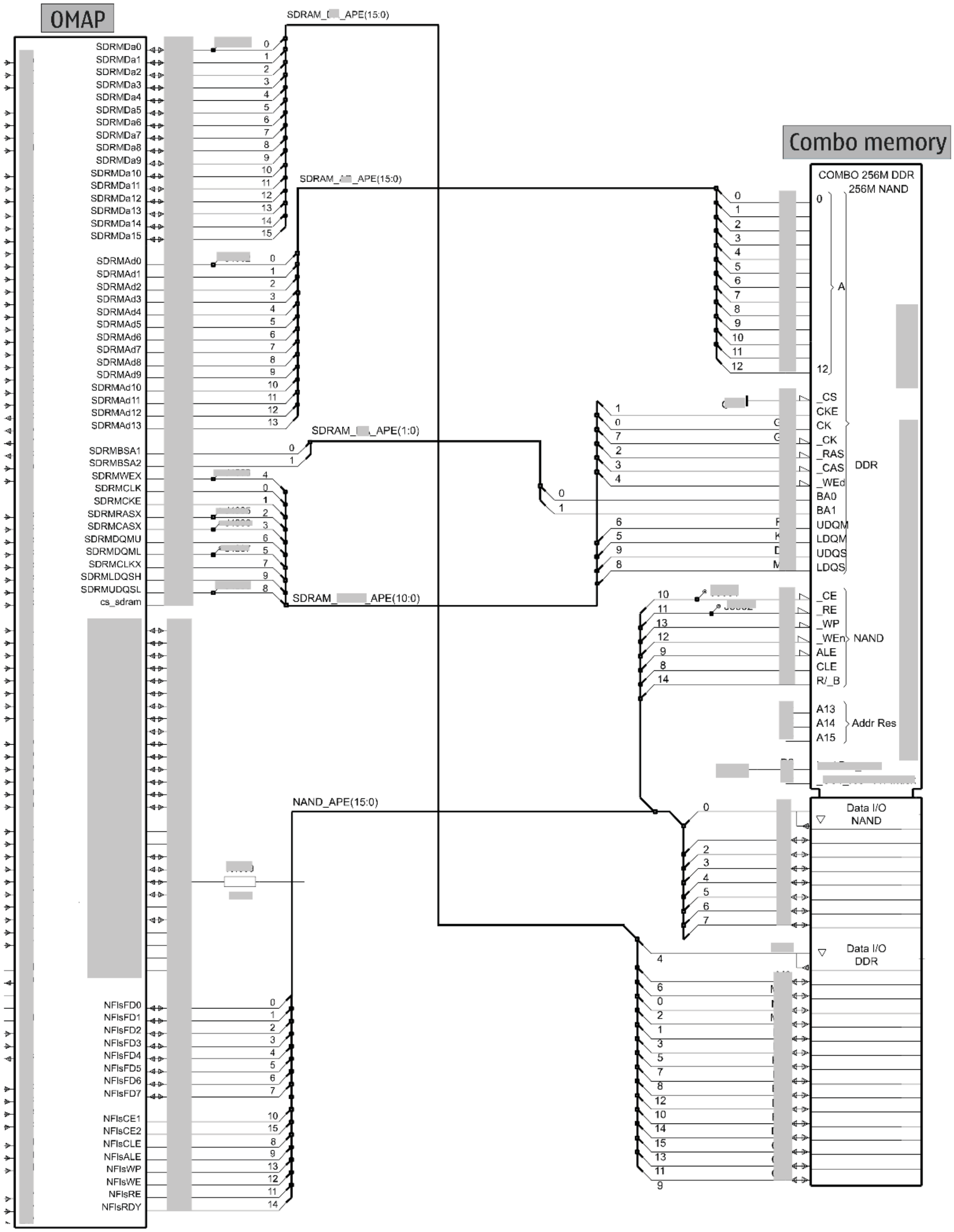
THE PARTIAL LOG OF ONENAND FLASHING

```
Flashing started
Creating product data items list
Product data items list created
Backup not required
Flashing phone
Current mode of Programming Device: Normal
Preparing for file upload...
File uploading... 0%
File uploading... 1%
…
File uploading... 98%
File uploading... 99%
File uploading... 100%
File uploaded
Finishing file upload... 0%
File upload finished 100%
CU-4 already in FBUS mode
Phone programming
Flashing preparation done
CMT secondary download
BB Asic Index is: 00
Asic ID is: 000000010000029600010006400C196101021603
BB Asic Index is: 00
EM Asic ID is: 00000265
BB Asic Index is: 00
EM Asic ID is: 00000B20
BB Asic Index is: 00
Public ID is: 1C7000128994005854F899958F2B0878B45C9946
BB Asic Index is: 00
Asic Mode ID: 00
BB Asic Index is: 00
Hash: OLIPUAV5D3C48E6DC22B49DC5063A2AD
BB Asic Index is: 00
ROM ID: AXC1X6713X691XX8
Start procedure done
CMT Algorithm code download
BB Asic Index is: 00
Device type is: 05
Device Index is: 00
Manufacturer code: 0000
Device ID: 0000
Extended device ID: 0000
Revision ID: 0000
BB Asic Index is: 00
Device type is: 04
Device Index is: 00
Manufacturer code: ffff
Device ID: 0000
Extended device ID: 0000
Revision ID: 0000
BB Asic Index is: 00
Device type is: 00
Device Index is: 00
Manufacturer code: 0020
Device ID: 0030
Extended device ID: 0000
Revision ID: 0000
BB Asic Index is: 00
Device type is: 00
Device Index is: 01
Manufacturer code: 0000
Device ID: 0001
Extended device ID: 0000
Revision ID: 0000
BB Asic Index is: 00
Device type is: 03
Device Index is: 00
Manufacturer code: 0020
Device ID: 0030
Extended device ID: 0000
Revision ID: 0021
CMT Algorithm code downloaded
Target programming
Maximum time for Target flash erasing : 300s and programming 2015s
Using timeout value: 2315s
CMT Algorithm code downloaded
Target erasing
Maximum time for Target flash erasing : 300s and programming 2015s
Using timeout value: 2315s
Target erasing completed
Target programming
BB Asic Index 00 (RAPx) programming succeeded
Device type is: 03
Device Index is: 00
Number of Bad Blocks: 4
Number of Initial Bad Blocks: 4
Number of run-time Bad Blocks: 0
Target programming completed
Target disconnect
File upload finished 100%
CU-4 already in FBUS mode
Phone programming
CMT Algorithm code downloaded
Target erasing
Maximum time for Target flash erasing : 300s and programming 5s
Using timeout value: 305s
Target erasing completed
Target programming
BB Asic Index 00 (RAPx) programming succeeded
Device type is: 03
Device Index is: 00
Number of Bad Blocks: 4
Number of Initial Bad Blocks: 4
Number of run-time Bad Blocks: 0
Target programming completed
Target disconnect
File programmed successfully!
Time taken to erase 0s and program 1s.
Total time 1 seconds.
Flashing time: 0 min and 1 sec
Phone flashing completed. Waiting for phone to boot up
Bootup successful
Verifying communication to product (before flash finalizing)
Communication verified
Product code changed
Started product mode change
Product mode change complete
Doing factorysets
Factorysets complete
Loading default data to phone
Loading default data to phone
Getting Data Package
Reading product state
Starting backup/restore sub-procedure:
data item pre-delay
data item pre-delay
data item pre-delay
Sub-procedure completed: Succeeded., result code: 0
Starting to backup/restore data item: ProductProfile, version: 1.0
Data Item backup/restore completed: Succeeded., result code: 0
Starting backup/restore sub-procedure:
data item post-delay
data item post-delay
data item post-delay
Sub-procedure completed: Succeeded., result code: 0
Backup/restore result: 0 out of 1 items were not backed up
Default data loading complete
Stopping all operations, returning phone to default mode
All operations completed
Product flashing succeeded
.
```

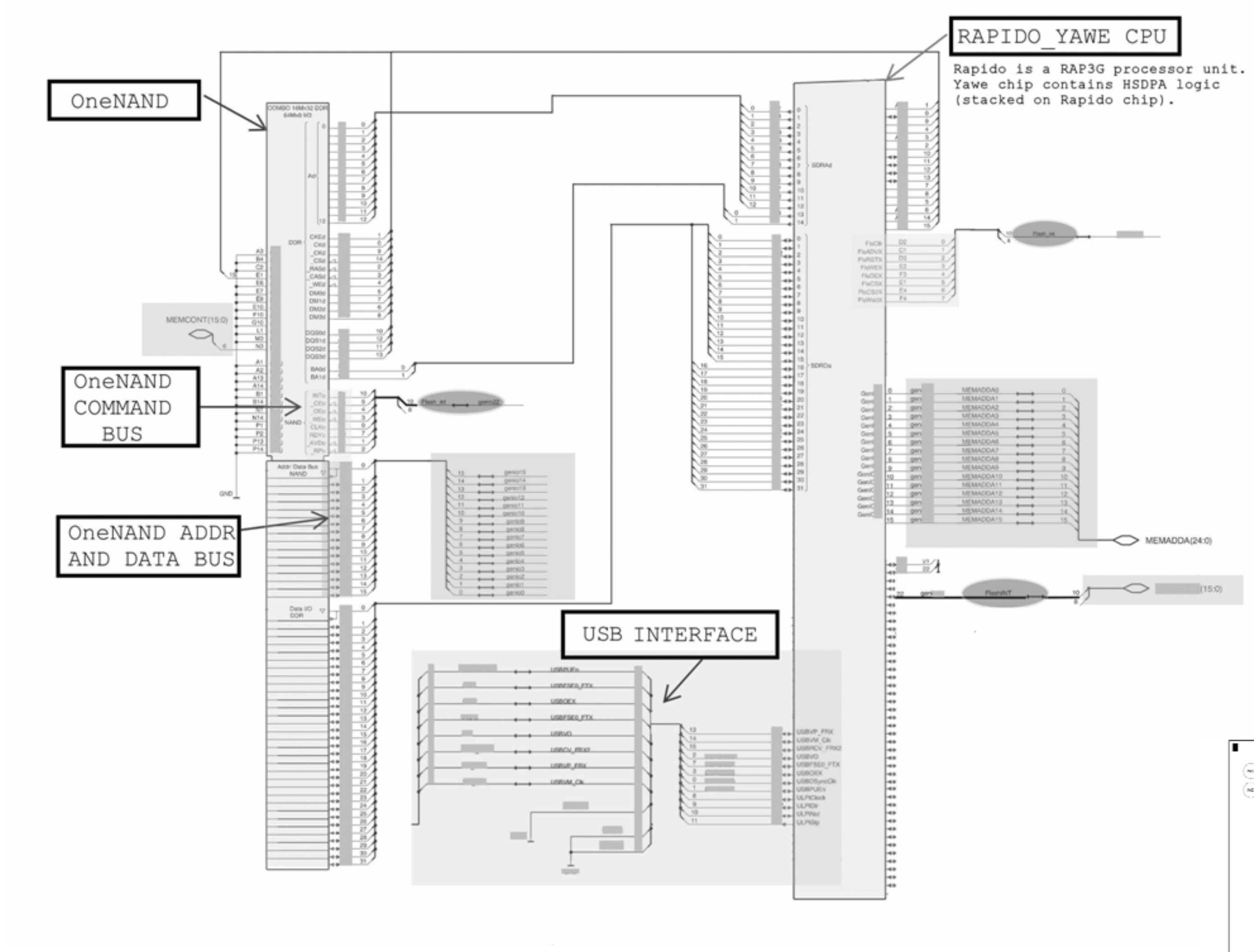
	NOR	***NAND***
Access to data	The data can be accessed at random like SRAM. The operations on the flash can be: *Read routine:* Read the contents of the flash. *Erase routine:* Erase is the process of making all the bits on a flash 1. Erase on the NOR chips happens in terms of blocks (referred to as erase regions). *Write routine:* Write is the process of converting a 1 to 0 on the flash. Once a bit is made 0, it cannot be written into until the block is erased, which sets all the bits in a block to 1	The NAND chips divide the storage into blocks, which are divided into pages again. Each page is divided into regular data and out-of-band data. The out-of-band data is used for storing metadata such as ECC (Error-Correction Code) data and bad block information. The NAND flash like the NOR flash has three basic operations: read, erase, and write. However, unlike NOR which can access data randomly, the NAND reads and writes are done in terms of pages whereas erases happen in terms of blocks
Interface to the board	These are connected like the normal SRAM device to the processor address and data bus.	There are multiple ways of connecting the NAND flash to the CPU varying across vendors. Raw NAND access is done by connecting the data and command lines to the usually 8 IO lines on the flash chip.
Execution of code	Code can be executed directly from NOR because it is directly connected to the address/data bus.	If code is in NAND flash it needs to be copied to memory for execution.
Performance	NOR flash is characterized by slow erase, slow write, and fast read	NAND flash is characterized by fast erase, fast write, and fast read.
Bad blocks	NOR flash chips are not expected to have bad blocks because they have been designed to hold system data	These flashes have been designed as basically media storage devices at lower prices, so expect that they have bad blocks. Normally these flash chips come with the bad sectors marked in them. Also NAND flash sectors suffer more the problem of bit flipping where a bit gets flipped when being written to; this is detected by error correcting algorithms called ECC/ EDC, which are done either in hardware or in software
Usage	These are basically used for code execution. Boot loaders can exist on the NOR flashes because the code from these flashes can be directly executed. These flashes are pretty expensive and they provide lesser memory densities and have a relatively shorter life span (around 100,000 erase cycles).	These are used mainly as storage devices for embedded systems such as set-top boxes and MP3 players. If you plan to use a board with only NAND, you may have to put in an additional boot ROM. They offer high densities at lower prices and have a longer life span (around 10 to the power of 6 erase cycles)

Tab. 2 *NOR versus NAND Flash (Raghavan et al., 2005)*

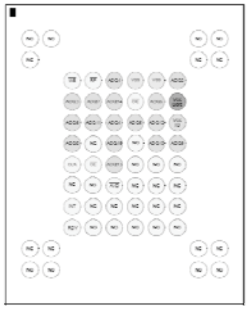
Adapted layout of OMAP and NAND on Nokia N70



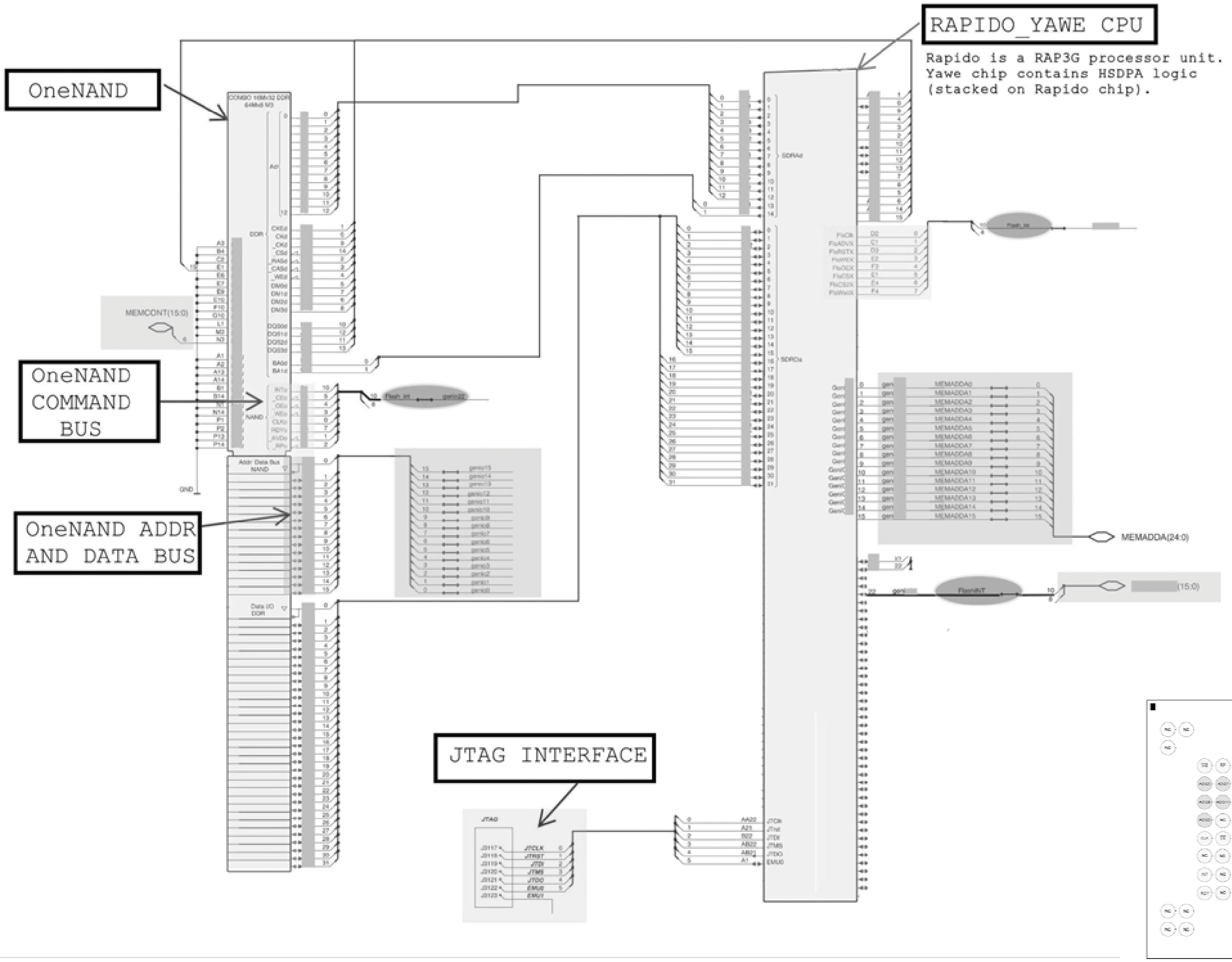
ADAPTED LAYOUT OF USB ACCESS TO ONENAND™ MEMORY ON NOKIA 6120C

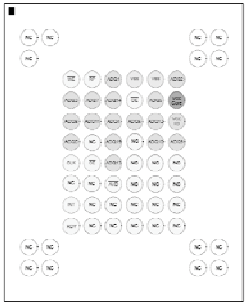


Rapido is a RAP3G processor unit. Yawe chip contains HSDPA logic (stacked on Rapido chip).



ADAPTED LAYOUT OF JTAG ACCESS TO ONENAND™ MEMORY ON NOKIA 6120C


RAPIDO_YAWE CPU
Rapido is a RAP3G processor unit.
Yawe chip contains HSDPA logic
(stacked on Rapido chip).
OneNAND
OneNAND COMMAND BUS
OneNAND ADDR AND DATA BUS
JTAG INTERFACE
MEMCONT(15:0)
MEMADDA(24:0)
SDRAd
SDRDa
JTAG
GND



REFERENCES

AL-ZAROUNI, M. 2006. *Mobile Handset Forensic Evidence - a challenge for Law Enforcement.* [Online]. Available: http://viewer.zoho.com/docs/zwcUl [Accessed].

BPMICROSYSTEMS. 2008. *Understanding NAND Flash Factory Programming* [Online]. Available: http://www.bpmicro.com/whitepapers/BPM_NAND_White_Paper_1008_docmetrics.pdf [Accessed].

BREEUWSMA, M., JONGH, M. D., KLAVER, C., KNIJFF, R. V. D. & ROELOFFS, M. 2007. Forensic Data Recovery from Flash Memory. *Small Scale Digital Device Forensics Journal,* 1.

CARRIER, B. 2005. *File System Forensic Analysis*, Addison-Wesley.

CASEY, E. 2004. *Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet*, Elsevier Academic Press.

CHANG, L.-P. & KUO, T.-W. 2004. *A Real-Time Garbage Collection Mechanism for Flash-Memory Storage Systems in Embedded Systems* [Online]. Available: http://www.cis.nctu.edu.tw/~lpchang/papers/RTCSA02-rdy.pdf [Accessed].

DAVID. 2009. *Hide Data in Bad Blocks* [Online]. Available: http://blog.crowdway.com/2009/04/22/hide-data-in-bad-blocks/ [Accessed].

DRAMEXCHANGE. 2009. *The 2009 Outlook of Taiwanese DRAM vendors* [Online]. Available: http://www.dramexchange.com/weeklyresearch/Post/2/1945.html [Accessed].

ELNEC. 2009. *Understanding NAND Flash Factory Pre-Programming Schemes* [Online]. Available: http://www.elnec.com/sw/an_elnec_nand_schemes.pdf [Accessed].

GAL, E. & TOLEDO, S. 2005. Algorithms and data structures for flash memories. *ACM Comput. Surv.,* 37**,** 138-163.

GARTNER. 2009. *Gartner Says Worldwide Mobile Phone Sales Declined 6 Per Cent and Smartphones Grew 27 Per Cent in Second Quarter of 2009* [Online]. Available: http://www.gartner.com/it/page.jsp?id=1126812 [Accessed].

GUPTA, M. R., HOESCHELE, M. D. & ROGERS, M. K. 2006. Hidden Disk Areas: HPA and DCO. *International Journal of Digital Evidence* 5.

HENDRIKX, J. 1998. *Space efficiency* [Online]. Available: http://www.xs4all.nl/~hjohn/SFS/spaceeff.htm [Accessed].

HUFFMAN, A. 2006. *A Standard Interface for NAND Flash* [Online]. Available: http://onfi.org/wp-content/uploads/2009/02/idf_s06_mems005_revd_taiwan.pdf [Accessed].

INOUE, A. & WONG, D. 2004. *NAND Flash Applications Design Guide* [Online]. Available: http://www.toshiba.com/taec/components/Generic/nand_design/nand_design_guide.pdf [Accessed].

INTEL. 1998. *Understanding the Flash Translation Layer (FTL) Specification* [Online]. Available: http://www.scribd.com/doc/7168824/Understanding-Flash-Translation-Layer [Accessed].

INTEL. 2006. *Flash File Systems Overview* [Online]. Available: http://www.eet-china.com/ARTICLES/2006NOV/PDF/Intel_WP_Flash_File_Systems_Overview.pdf [Accessed].

JANSEN, W. & AYERS, R. 2007. *NIST Special Publication 800-101. Guidelines on Cell Phone Forensics* [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-101/SP800-101.pdf [Accessed].

JI, Y. L., CHANG, C. C., YEN, C. N. & SHONE, F. 2009. *Method of wear leveling for non-volatile memory and apparatus using the same*. United States patent application.

JONES, M. T. 2008. *Anatomy of Linux flash file systems* [Online]. Available: http://www.ibm.com/developerworks/linux/library/l-flash-filesystems/ [Accessed].

KURTZ, C. & SNOWDEN, D. 2003. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM System Journal,* 42**,** 22.

KWON, T. 2009. *Memory device in mobile phone* [Online]. United States: Samsung Electronics Co., Ltd (KR). Available: http://www.freepatentsonline.com/7496377.html [Accessed].

LYLE, J. R. & WOZAR, M. 2007. Issues with imaging drives containing faulty sectors. *Digital investigation journal***,** 1 3 – 1 5.

MCCARTHY, P. 2005. *Forensic Analysis of Mobile Phones* [Online]. Available: http://www.8051projects.net/e107_files/public/1236046309_9698_FT19075_forensic_analysis_of_mobile_phones.pdf [Accessed].

MICRON. 2006a. *TN-29-19. NAND Flash 101: An Introduction to NAND Flash and How to Design It In to Your Next Product* [Online]. Available: http://download.micron.com/pdf/technotes/nand/tn2919.pdf [Accessed].

MICRON. 2006b. *TN-29-17: NAND Flash Design and Use Considerations Introduction* [Online]. Available: http://download.micron.com/pdf/technotes/nand/tn2917.pdf [Accessed].

MICRON. 2006c. *TN-29-11: NAND Flash Security Overview* [Online]. Available: http://download.micron.com/pdf/technotes/nand/tn2911.pdf [Accessed].

MICROSOFT. 2009. *Default cluster size for NTFS, FAT, and exFAT* [Online]. Available: http://support.microsoft.com/kb/140365 [Accessed].

MORRIS, B. 2007. *The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS*, John Wiley & Sons.

MTD_GROUP. 2008. UBIFS - UBI File-System. Available: http://www.linux-mtd.infradead.org/doc/ubifs.html.

MUKASEY, M. B., SEDGWICK, J. L. & HAGY, D. W. 2008. Test Results for Digital Data Acquisition Tool: DCCIdd (Version 2.0).

NDT. 2008. *Re: Unscramble your PS3 dump with Flow Rebuilder!* [Online]. Available: http://www.infectus.biz/forum/index.php?topic=2279.msg16333#msg16333 [Accessed].

NIST. 2008. *GSM Mobile Device and Associated Media Tool Specification and Test Plan (ver. 1.1)* [Online]. Available: http://www.cftt.nist.gov/documents/GSM-Mobile-Device-and-Associated-Media-Tool-Specification-and-Test-Plan-050508a.pdf [Accessed].

NIST. 2009. *GSM Mobile Device and Associated Media Tool Specification and Test Plan (ver. 1.2)* [Online]. Available: http://www.cftt.nist.gov/GSM%20Mobile%20Device%20and%20Associated%20Media%20Tool%20Specification.pdf [Accessed].

NUMONYX. 2008a. *Flash File Systems Overview* [Online]. Available: http://www.numonyx.com/Documents/WhitePapers/Flash_file_systems_WP.pdf [Accessed].

NUMONYX. 2008b. *Garbage collection in NAND flash memories* [Online]. Available: http://www.numonyx.com/Documents/Application%20Notes/AN1821.pdf [Accessed].

NUMONYX. 2008c. *Wear Leveling in NAND flash memories* [Online]. Available: http://www.numonyx.com/Documents/Application%20Notes/AN1822.pdf [Accessed].

NUMONYX. 2008d. *AN1823 Application note: Error correction code in single level cell NAND flash memories* [Online]. Available: http://www.numonyx.com/Documents/Application%20Notes/AN1823.pdf [Accessed].

NUMONYX. 2008e. *AN1819 - Application note. Bad block management in NAND flash memories* [Online]. Available: http://www.numonyx.com/Documents/Datasheets/AN1819.pdf [Accessed].

NUMONYX. 2009. *RE: Raw NANDs (personal comunication).*

O'KELLY, T. 2007. *Reference guide for flash cards and drives* [Online]. Available: http://memorex.com/downloads/whitepapers/ReferenceGuideforFlashCardsandDrives7-10-07.pdf [Accessed].

PENG, W.-C. 2006. *Multi Level and 2-Bit/Cell Operation for SONOS Memory with Wrapped-Select-Gate Structure Using Source-Side Injection Programming* [Online]. Available: http://ethesys.lib.fcu.edu.tw/ETD-search/getfile?URN=etd-0704106-122939&filename=etd-0704106-122939.pdf [Accessed].

PON, H., LEE, C. & ANYIMI, C. 2007. *Sizing up flash options for mobile devices* [Online]. Available: http://www.smallformfactors.com/pdfs/Intel.RG07.pdf [Accessed].

RAGHAVAN, P., LAD, A. & NEELAKANDAN, S. 2005. *Embedded Linux System Design and Development* Auerbach.

SAMSUNG. 1999. *Application note for NAND Flash Memory* [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/applicationnote/app_nand.pdf [Accessed].

SAMSUNG. 2004. NAND Flash ECC Algorithm. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/applicationnote/eccalgo_040624.pdf.

SAMSUNG. 2005a. NAND Flash Spare Area Assignment Standard. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/applicationnote/spare_assignment_standard.pdf.

SAMSUNG. 2005b. *OneNAND™ Features & Performance* [Online]. Available: http://210.118.57.82/global/business/semiconductor/products/fusionmemory/downloads/applicationnote/onenand_features_performance_051104.pdf [Accessed].

SAMSUNG. 2005c. *MuxOneNAND Specification. MuxOneNAND1G(KFM1G16Q2M-DEB6)* [Online]. Available: http://www.samsung.com/global/system/business/semiconductor/product/2007/6/11/OneNAND/1Gbit/KFM1G16Q2M/ds_mux_kfxxg16q2m_66mhz_rev12.pdf [Accessed].

SAMSUNG. 2005d. Application Note for OTP Program. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/applicationnote/2005_0628_OTP_AppNote.pdf.

SAMSUNG 2006a. RFS v1.2.1- Pre-programming Guide.

SAMSUNG. 2006b. *UniStore II v1.5.1. Installation Guide* [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/unistoreII_v151_installation_guide_web.pdf [Accessed].

SAMSUNG. 2007. *XSR 1.5. Bad Block Managment* [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/xsr_v15_badblockmgmt_application_note.pdf [Accessed].

SAMSUNG. 2007a. *TFS4-Hidden Protected Area. Application Note.* [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/tfs4_v16_hpa_rev10.pdf [Accessed].

SAMSUNG. 2008. *XSR v1.6.1 Porting Guide* [Online]. Available: http://www.samsung.com/global/business/semiconductor/products/flash/downloads/xsr_v161_porting_guide.pdf [Accessed].

STMICROELECTRONICS. 2004. How to Boot from a Single Level Cell NAND Flash Memory. Available: www.eetasia.com/ARTICLES/2004DEC/A/2004DEC08_MEM_AN03.PDF [Accessed 21/11/2009].

TSAI, Y.-L., HSIEH, J.-W. & KUO, T.-W. 2006. Configurable NAND Flash Translation Layer. *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*

WHITE, B. 2008. *Factory Preprogramming Solutions for NAND Flash Devices* [Online]. Available: http://www.bpmmicro.com/pdf/FMS2008-WhiteT1BSlides.pdf?utm_source=flashstream-site&utm_medium=footer-link&utm_campaign=nand-slides [Accessed].

WILLASSEN, S. Y. 2003. Forensics and the GSM mobile telephone system. *International Journal of Digital Evidence,* 2

COPYRIGHT