Understand your NAND and drive it within Linux

Miquèl Raynal
Bootlin
miquel@bootlin.com

bootlin

New name, same team

Miquèl Raynal

- Embedded Linux engineer at ~~Free Electrons~~ → Bootlin
  - Embedded Linux **development**: kernel and driver development, system integration, boot time and power consumption optimization, consulting, etc.
  - Embedded Linux, Linux driver development, Yocto Project / OpenEmbedded and Buildroot **training courses**, with materials freely available under a Creative Commons license.
  - `https://bootlin.com`
- Contributions
  - **Active contributor to the NAND subsystem**
  - **Kernel support for various ARM SoCs**
- Living in **Toulouse**, south west of France

What is this talk about?

- Introduction to the basics of NAND flash memory
- How they are driven by the NAND controller
- Overview of the Linux memory stack, especially the new interface to drive NAND controllers: `->exec_op()`

Disclaimer

- I am not a NAND expert, more the NAND maintainer slave
- I will probably oversimplify some aspects
- This presentation is not about history nor NOR technology
- Focus on SLC NAND (Single Level Cell)

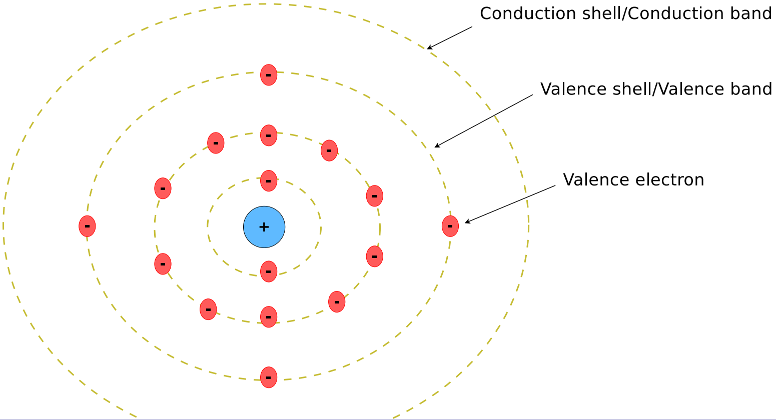
The commercial minute

- Main purpose: replace hard disks drives
- Main goal: lowest cost per bit
- Widely used in many consumer devices, embedded systems...
- Flavors:
  - **Raw NAND / parallel NAND** ⇐
  - Serial NAND (mostly over SPI)
  - Managed NAND with FTL (Flash Translation Layer)
    - SD cards
    - USB sticks
    - SSD
    - etc

Understanding the NAND memory cell

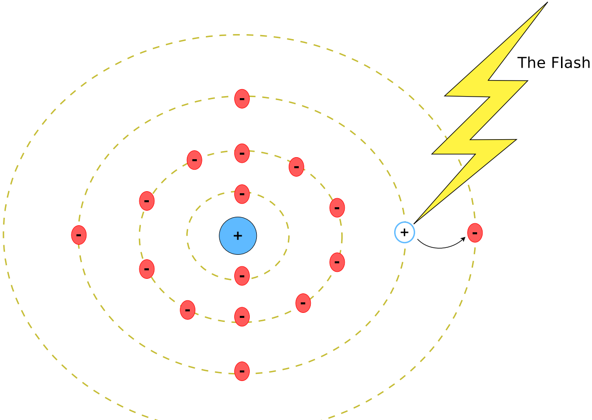
Back to school: Silicon

- Silicon, Si
  - Electrically balanced (neutral)
  - 14 electrons spread in 3 orbits
  - 4 electrons in the valence shell → easy bonding with other Silicon atoms (crystal)



Back to school: electricity

- Electricity $\implies$ free electrons
  - Silicon is almost an insulator
  - Valence electron stroke by light $\rightarrow$ absorbs energy $\rightarrow$ jumps to the conduction band
  - Free electrons drift randomly unless a voltage is applied $\rightarrow$ attracted to the $+$ side

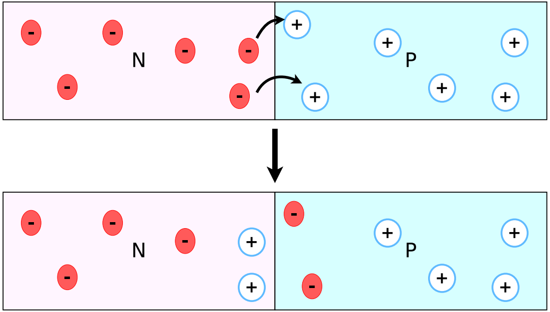


Back to school: doping

- Nothing to do with cycling
- Purpose of doping: enhance conductivity
  - Add impurities (atoms with more or less valence electrons than Si)
  - Once bound with 4 Si atoms:
    - 1 free electron $\leftarrow$ N-doping
    - 1 hole $\leftarrow$ P-doping
  - Still electrically neutral

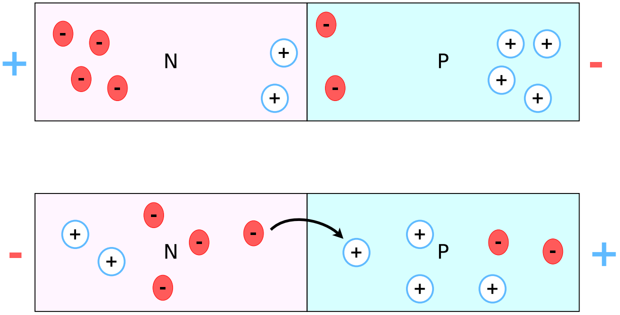
P-N junction: the diode

- Electrons close to the junction will jump to recombine with the closest hole
- Creation of a barrier of potential: a non-crossable electric field
- Depletion region thickness is modular

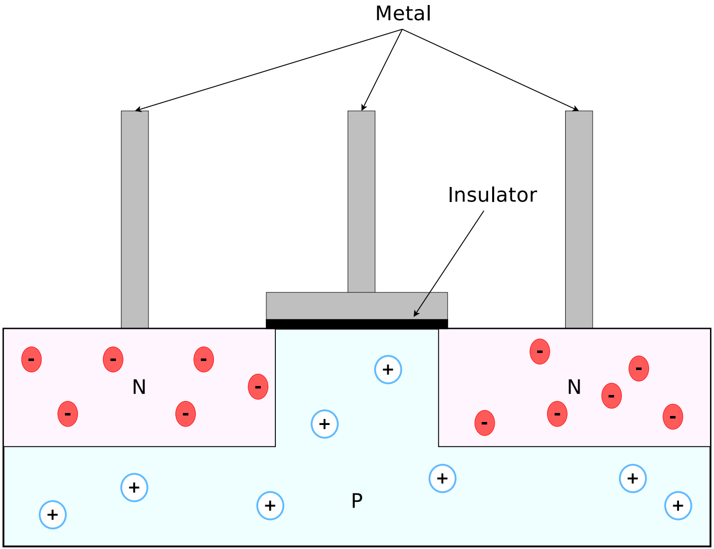


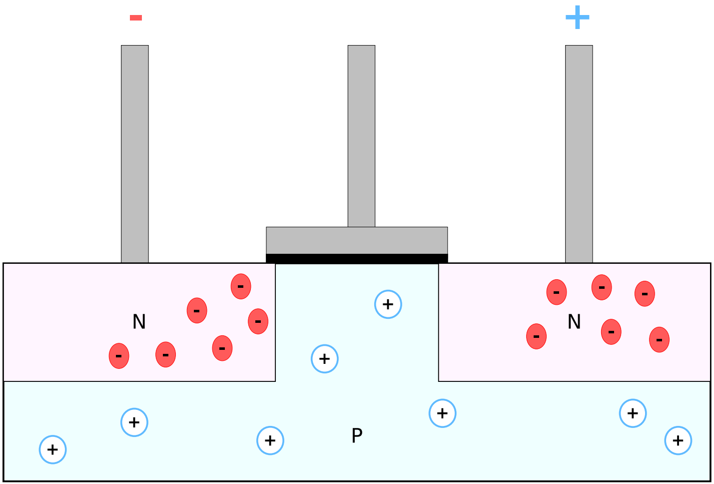
P-N junction: the diode

- Electrons close to the junction will jump to recombine with the closest hole
- Creation of a barrier of potential: a non-crossable electric field
- Depletion region thickness is modular

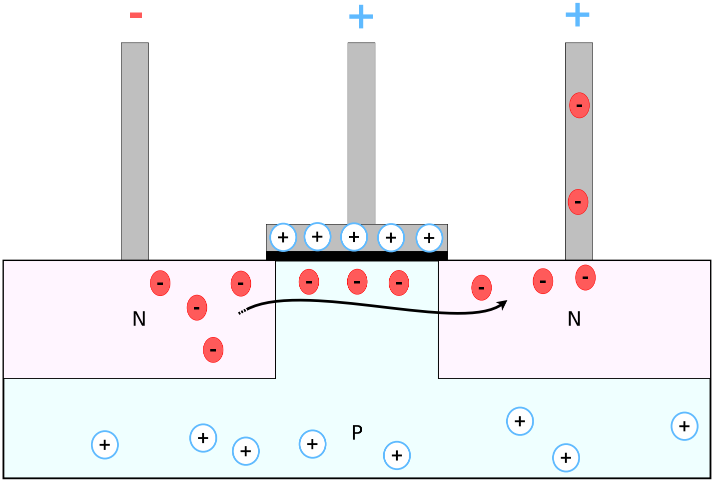


Metal-Oxide-Semiconductor Field-Effect Transistor

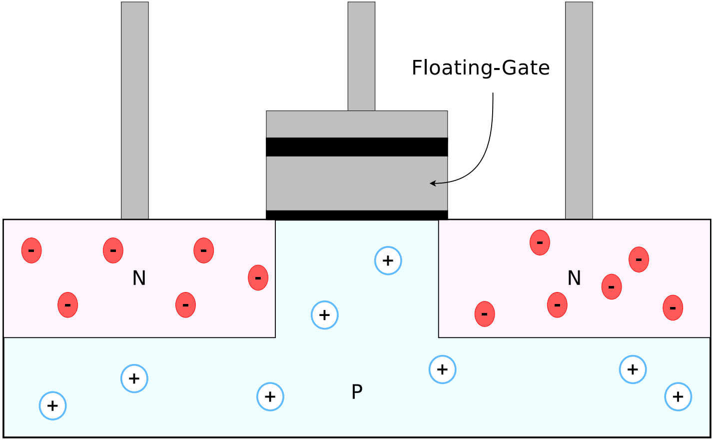



-
+
N
N
P

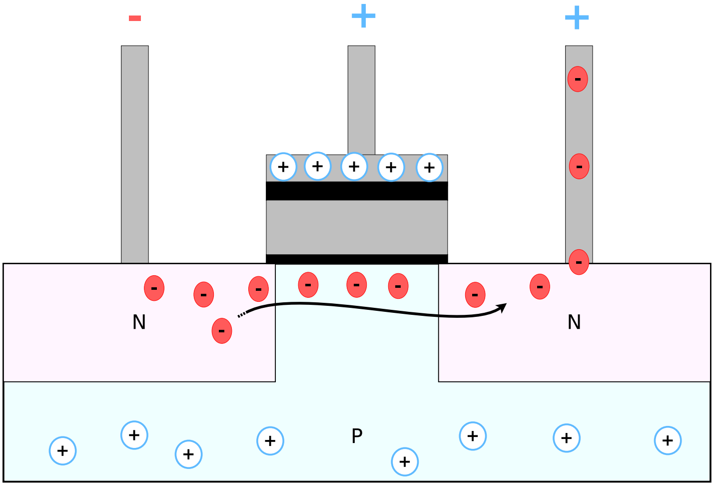
Metal-Oxide-Semiconductor Field-Effect Transistor



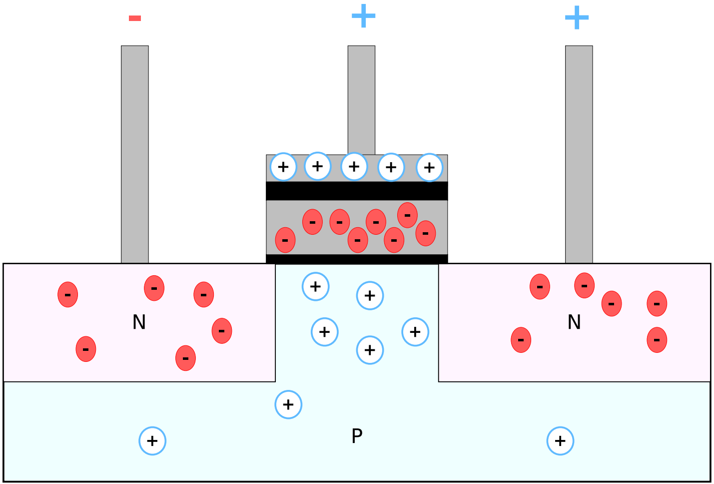
Floating-gate transistor



Floating-gate transistor: reading a one

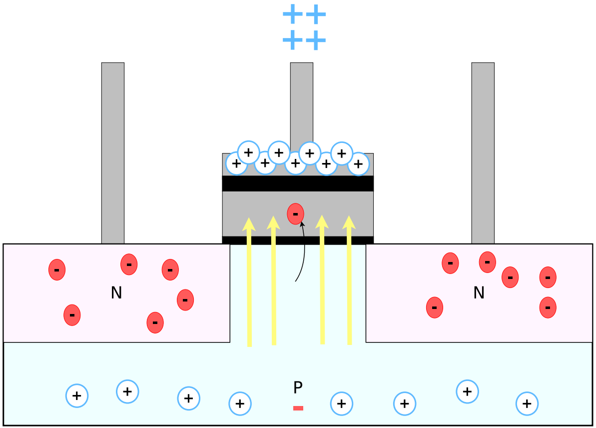


Floating-gate transistor: reading a zero



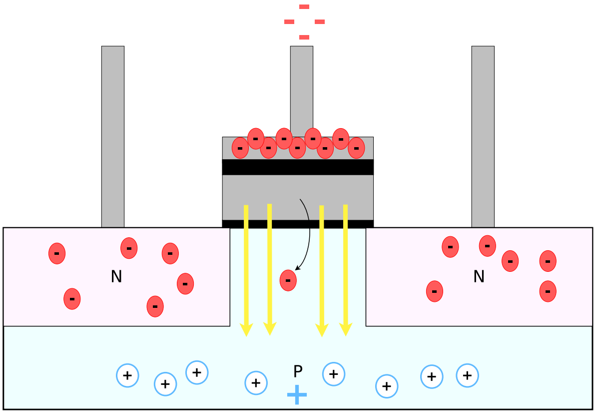
Programming a cell to a 0 state

- Change the charge of the floating-gate
- No electrical contact → Fowler-Nordheim tunneling

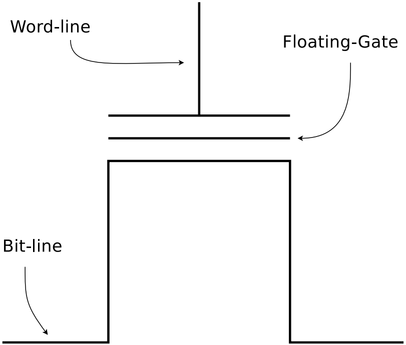


Erasing a cell to a 1 state

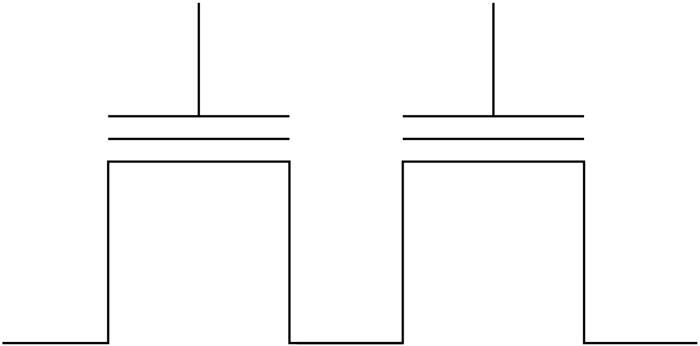
- Reverse the electric field
- Done by applying a high negative voltage on the control gate



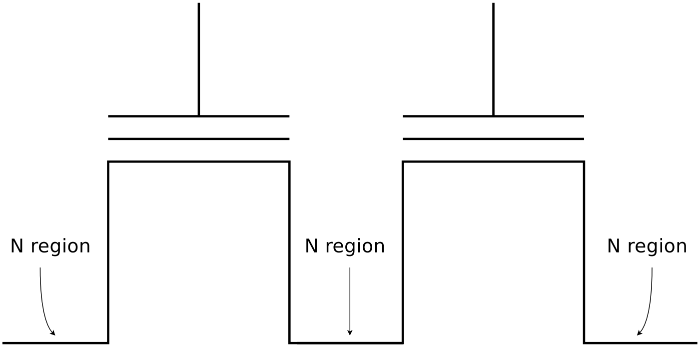
Memory cell



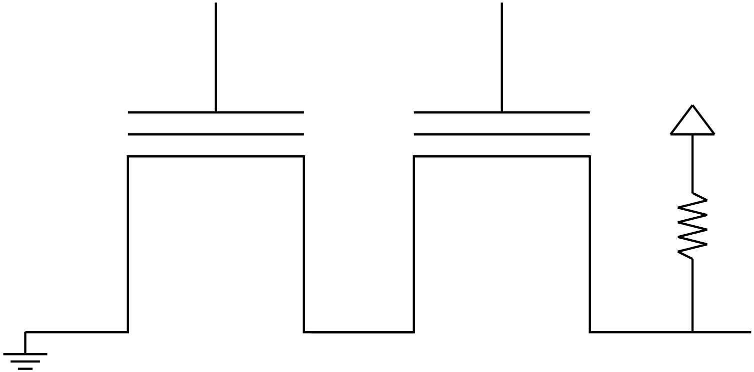
NAND gate



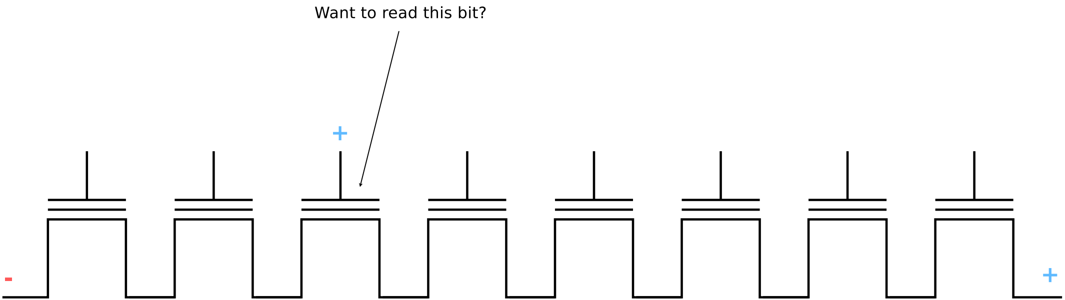
NAND gate



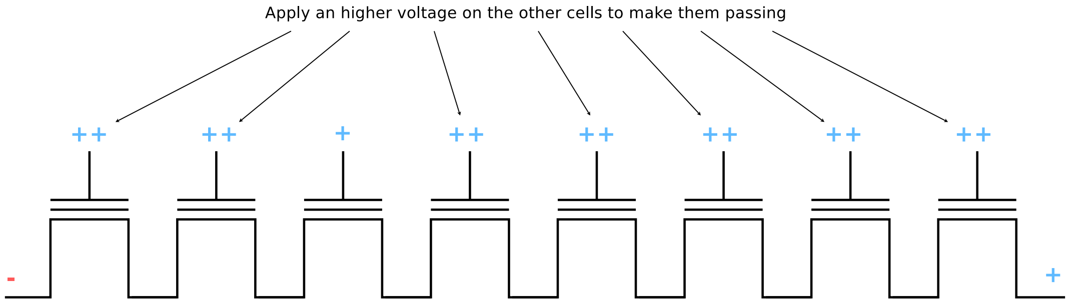
NAND gate



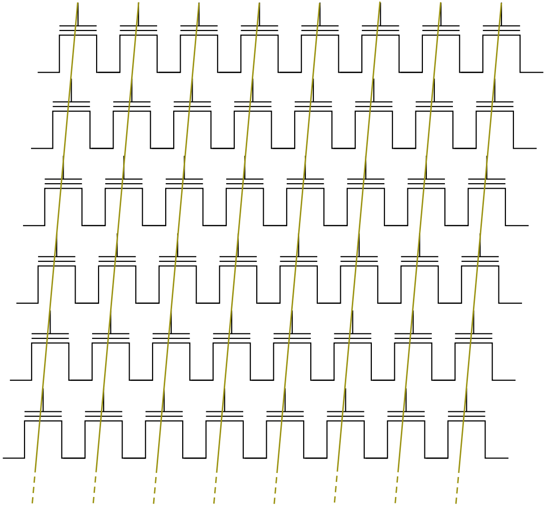
Memory string



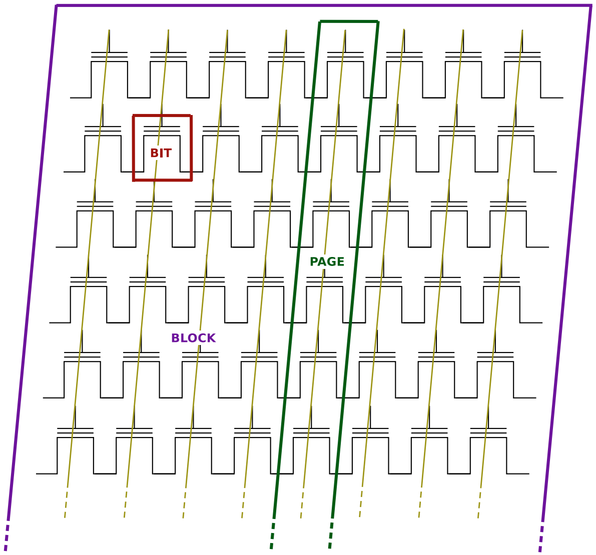
Memory string



Memory array

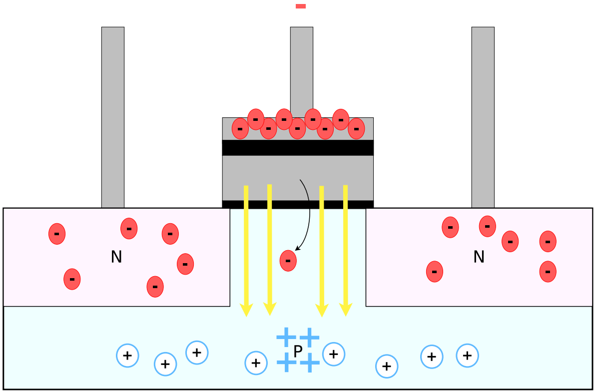


Memory array



Erasing

- High negative voltage → not that easy to produce
- Bulk is the same for all cells → "eraseblock"

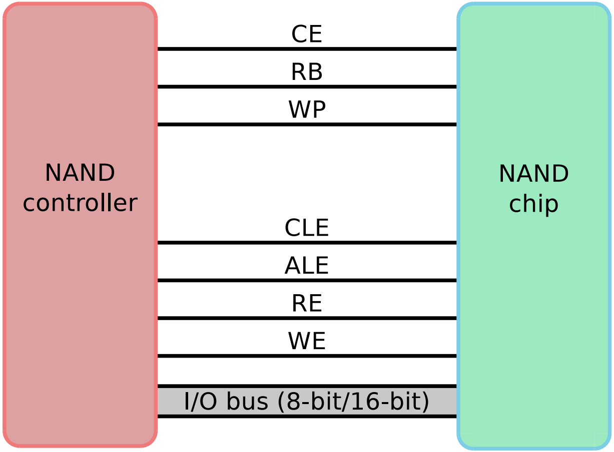


Main design flaw: bitflips

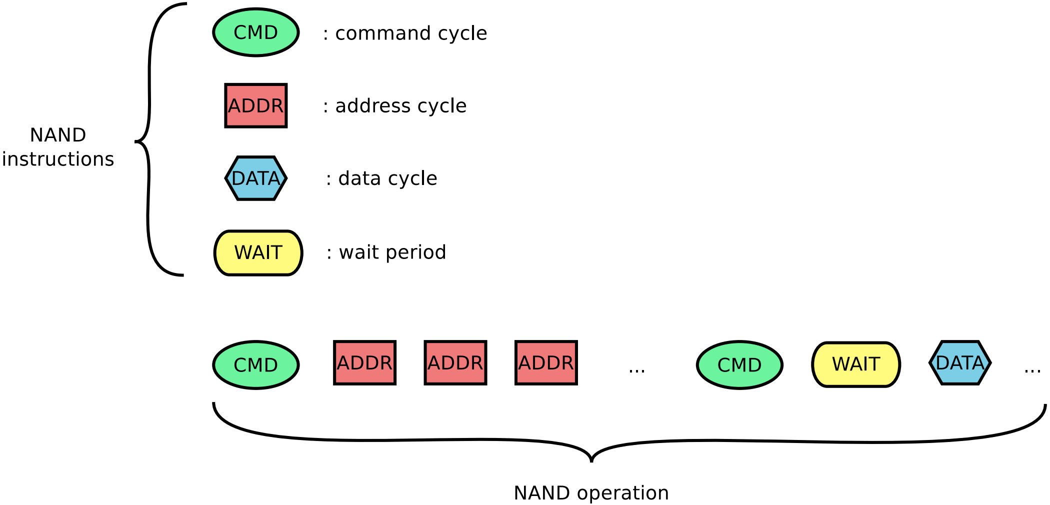
- Tunneling $\rightarrow$ stochastically distributed
- Cells may not be fully erased/programmed
  - Electrons without enough energy might get trapped, creating a depletion region
  - Oxide becomes negative, preventing tunneling of the electrons if the barrier gets too high
- Data retention issue
  - Writing/erasing moves electrons through the oxide layer
  - Electrons will dissipate their energy colliding with the material, damaging it $\rightarrow$ possible charge loss
- Read/write disturbances
- $\sim$100k program/erase cycles with SLC NAND

Driving a NAND chip: the NAND controller

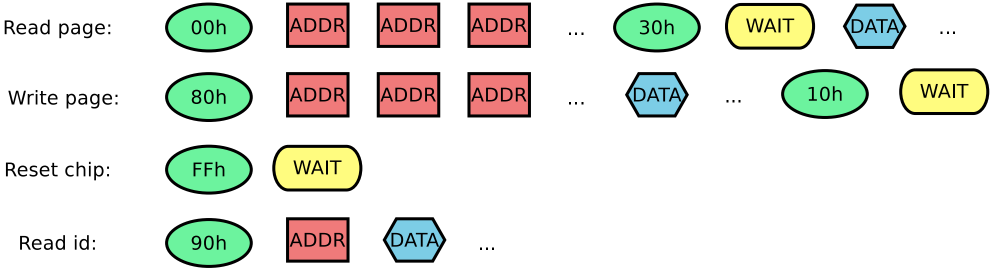
NAND bus


NAND controller
CE
RB
WP
CLE
ALE
RE
WE
I/O bus (8-bit/16-bit)
NAND chip

NAND protocol



NAND protocol (examples)

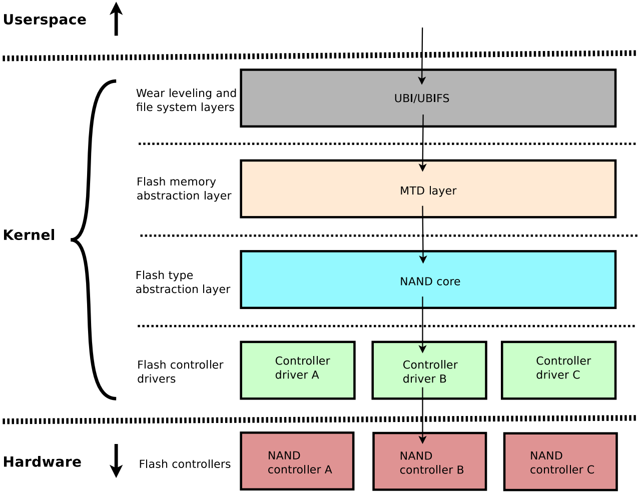


NAND controller

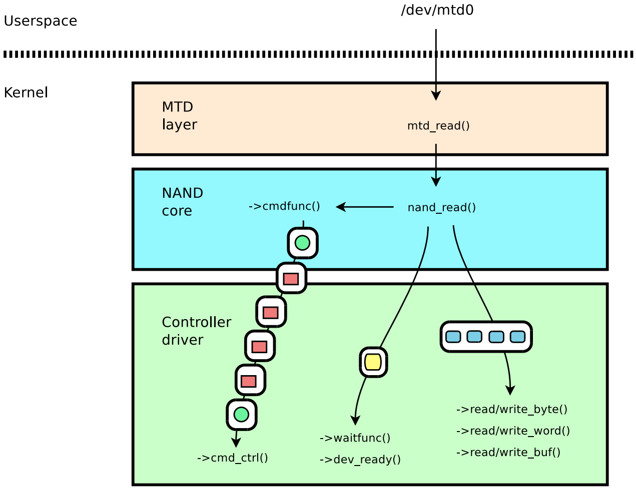
- Controllers are often embedded in a SoC
- Diverse implementations, from the most simplest to highly sophisticated ones
- Controller job: communicate with the NAND chip
  - Can embed an ECC engine to handle bitflips
  - Can embed advanced logic to optimize throughput
    - Sequential accesses
    - Parallel die accesses

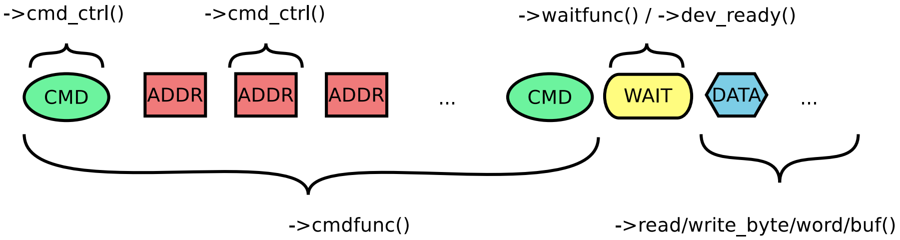
Dealing with NAND from Linux

Linux MTD stack



NAND legacy stack




->cmd_ctrl()
->cmd_ctrl()
->waitfunc() / ->dev_ready()
CMD
ADDR
ADDR
ADDR
...
CMD
WAIT
DATA
...
->cmdfunc()
->read/write_byte/word/buf()

Limitations of the old methods

- NAND controllers cannot handle such fine grain instructions
- NAND controller drivers started to overload `->cmdfunc()`, which introduced new issues:
  - Need for the IO length (not provided by `->cmdfunc()`) → drivers started predicting what the core "next move" would be
  - NAND operations evolve over the time → need to add support for vendor specific operations → hard to maintain as support across the NAND controllers is not uniform at all → patch all the drivers for each operation addition in the core
  - According to the NAND maintainer, vendors are creative

    "Why are they so mean to us?!" – Boris Brezillon, 04/01/2018

  - NAND controller drivers have to re-implement everything → encourages people to implement a minimal set of commands

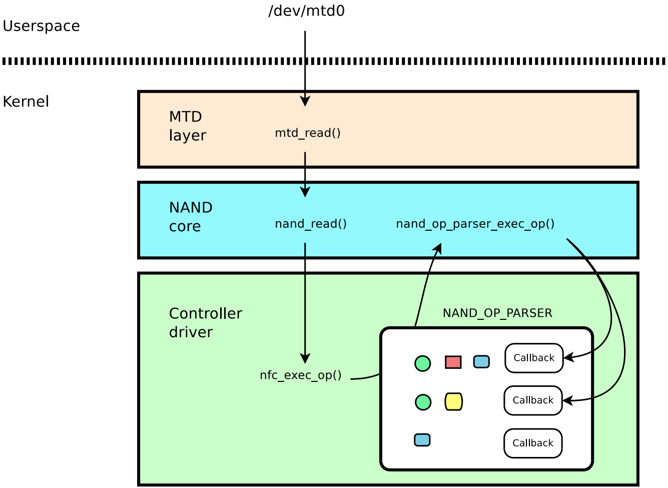
Addressing these limitations: ->exec_op()

- Create a new interface that asks to execute the whole operation
- Just a translation in NAND operations of the MTD layer orders
  - Don't try to be smart, logic should be in the NAND framework
- Calls the controller `->exec_op()` hook and pass it an array of instructions to execute
- Should fit most NAND controllers we already know about
- Introduction in Linux v4.16 expected
- Marvell's NAND controller driver migrated
- More to come: FSMC, Sunxi, VF610, Arasan, MXC, Atmel...

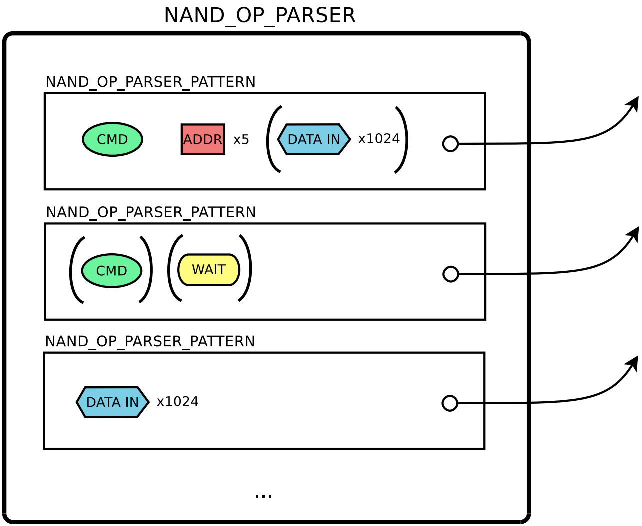
`->exec_op()` controller's implementation

- When receiving an array of sequential instructions:
  - Parse the sequence
    - Split in as much sub-operations as needed to perform the task
  - Declare if the overall operation can be handled
    - Otherwise return -ENOTSUPP
- Simple controllers → trivial logic
- More complex controllers → use the core's parser

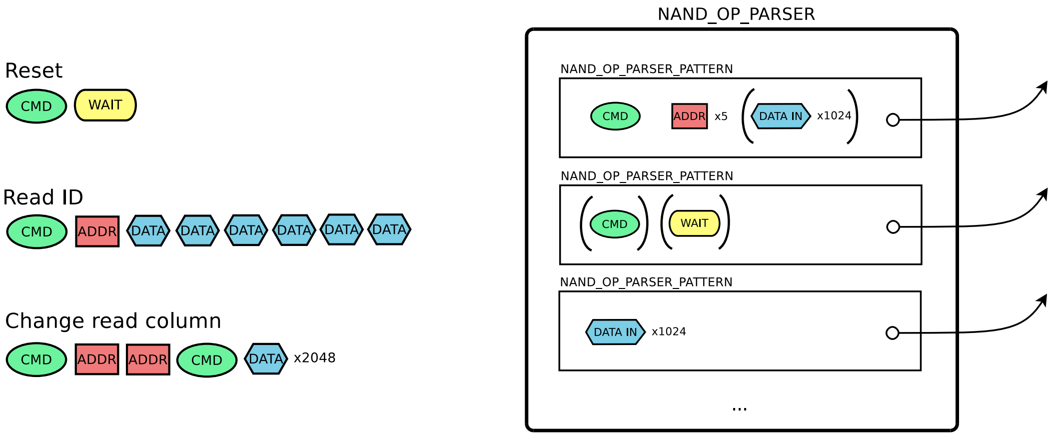
Parser



/dev/mtd0
Userspace
Kernel
MTD
layer
mtd_read()
NAND
core
nand_read()
nand_op_parser_exec_op()
Controller
driver
NAND_OP_PARSER
nfc_exec_op()
Callback
Callback
Callback


NAND_OP_PARSER
NAND_OP_PARSER_PATTERN
CMD
ADDR
x5
DATA IN
x1024
NAND_OP_PARSER_PATTERN
CMD
WAIT
NAND_OP_PARSER_PATTERN
DATA IN
x1024
...

Swipe right to match



NAND hooks

- Various hooks should be implemented by the controller driver
  - `->exec_op()` is one tool to do "low-level" operations
  - `->setup_data_interface()` to manage controller timings
  - `->select_chip()` to select a NAND chip die

Good habits when you hack a NAND controller driver

- Test with the userspace tools through the /dev/mtd* devices
  mtd-utils: nandbiterrs, nandreadpage, flash_speed, flash_erase, nanddump, nandwrite, etc

Good habits when you hack a NAND controller driver

- Test with the userspace tools through the `/dev/mtd*` devices
  mtd-utils: `nandbiterrs`, `nandreadpage`, `flash_speed`, `flash_erase`, `nanddump`, `nandwrite`, etc
- Get the NAND documentation
  `dd if=/dev/zero of=nand.txt`

Good habits when you hack a NAND controller driver

- Test with the userspace tools through the `/dev/mtd*` devices
  mtd-utils: `nandbiterrs`, `nandreadpage`, `flash_speed`, `flash_erase`, `nanddump`, `nandwrite`, etc
- Get the NAND documentation
  `dd if=/dev/zero of=nand.txt`
- Ping the MTD community early on the public mailing-list

Good habits when you hack a NAND controller driver

- Test with the userspace tools through the /dev/mtd* devices
  mtd-utils: nandbiterrs, nandreadpage, flash_speed, flash_erase, nanddump, nandwrite, etc
- Get the NAND documentation
  dd if=/dev/zero of=nand.txt
- Ping the MTD community early on the public mailing-list
- Do not forget to add the maintainer(s) in copy, it puts them in a bad mood

Sources/Links

- Presentation by Boris Brezillon (Free Electrons/Bootlin) at ELCE 2016 in Berlin: "Modernizing the NAND framework, the big picture"
  https://www.youtube.com/watch?v=vhEb0fgk71M
  https://events.linuxfoundation.org/sites/events/files/slides/brezillon-nand-framework_0.pdf
- Presentation by Arnout Vandecappelle (Essensium/Mind) at ELCE 2016 in Berlin: "Why NAND flash breaks down"
  https://www.youtube.com/watch?v=VajB8vCsZ3s
  https://schd.ws/hosted_files/openiotelceurope2016/36/Flash-technology-ELCE16.pdf
- YouTube channel "Learn engineering" that democratizes physical concepts
  https://www.youtube.com/watch?v=7ukDKVHnac4
- SlideShare by Nur Baya Binti Mohd Hashim (UNIMAP) about semiconductors
  http://slideplayer.com/slide/10946788

Questions? Suggestions? Comments?

Miquèl Raynal

miquel@bootlin.com

https://bootlin.com/pub/conferences/2018/fosdem/raynal-exec-op/

Backup

- For throughput or compatibility purpose, a controller driver may overload the following functions defined by the core to bypass `->exec_op()` and talk directly to the NAND controller
  - `->read/write_page()`
  - `->read/write_oob()`
    - Bitflips should be corrected and reported by the controller driver
    - Let the NAND core handle the rest and report to upper layers
- It is also mandatory to fill their “raw” counterpart in order to be able to test and debug all the functionalities of the driver
  - `->read/write_page_raw()`
  - `->read/write_oob_raw()`