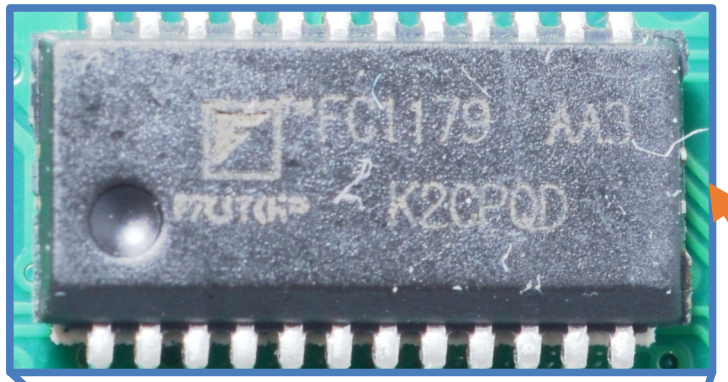
Data recovery from an FirstChip (FC) based device with Bad Bit columns demo

Alexey Taran

ruSolut

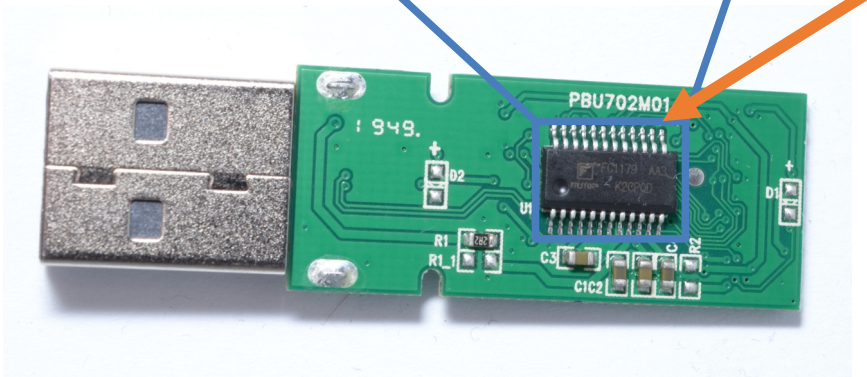
USB Mass Storage device with FC 1179 Controller

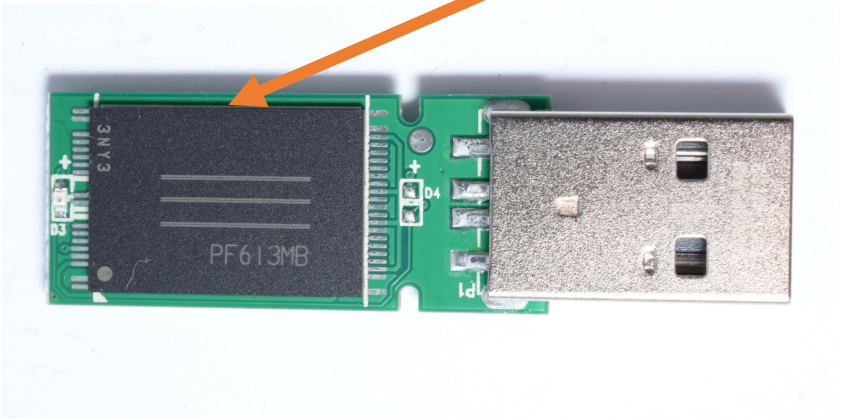


Used in cheap devices
Designed for low quality and refurbished NAND chips

FC 1179 controller

NAND Memory





Bad bits on physical image

Bad bits look like an vertical columns through dump

Does not cross the Block/Plane borders

May appear as single bit columns or grouped

Bad bits problems

Bad bits

Bad bits are:

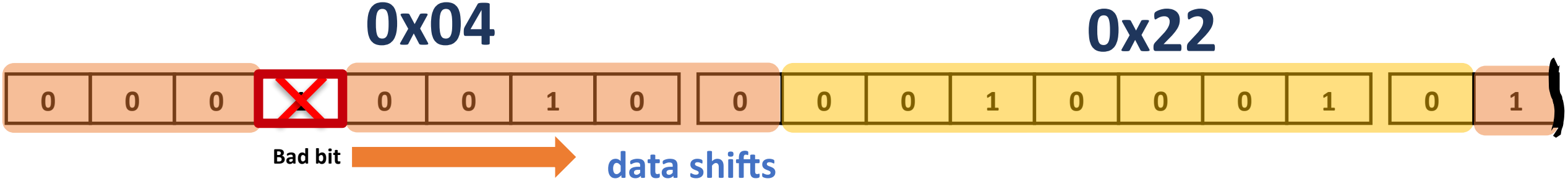
Unique for each FC case, as depends on physical NAND chip – Solution for one device will not work for exact same another device

Change Page Structure – XOR and ECC could not be found automatically with Bad bits

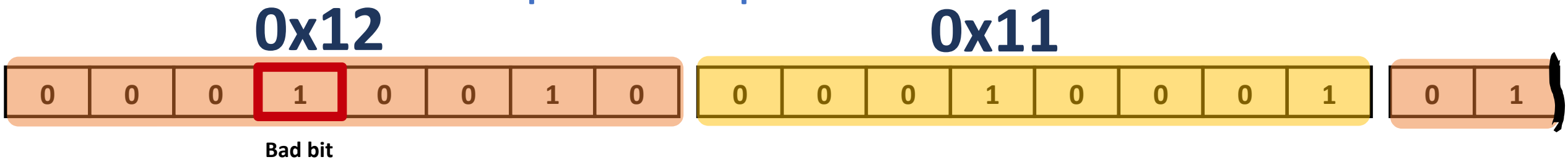
Different between Blocks/Plains in one device – Requires to make custom Page structures for each Plane

Difference in Write and Read

What was written by controller



What was red from NAND chip to the dump



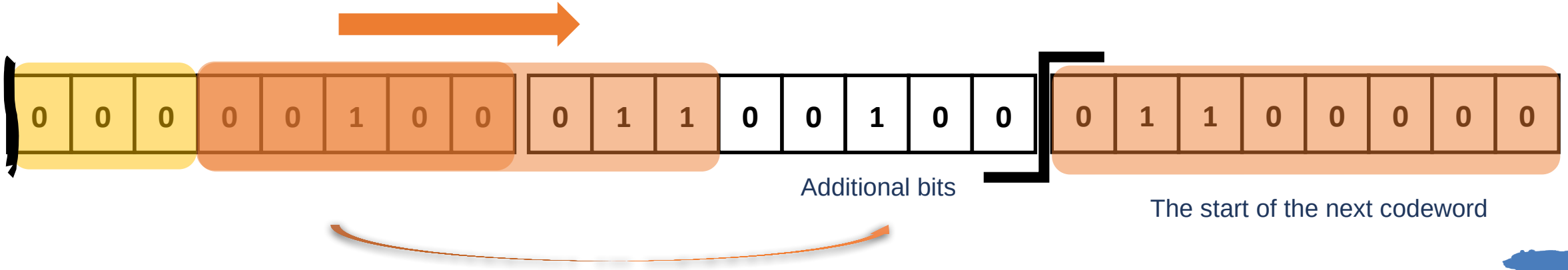
Even one bad bit column is enough to shift whole page and make all dump/data completely unreadable

ruSolut

Additional bits

As a result, the end of codeword is shifted and the controller writes additional bits to make the next codeword starts from the proper byte border (data is red/written by INTEGER BYTES!)

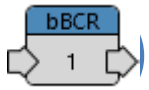
For instance, in case of 3 bad bits, 5 additional bits have to be added to make the next codeword starts from the proper integer byte
The values for the additional bits are taken from the previous byte, so in other words, the controller just repeats the last byte until the next physical byte border.



Removing Bad bits

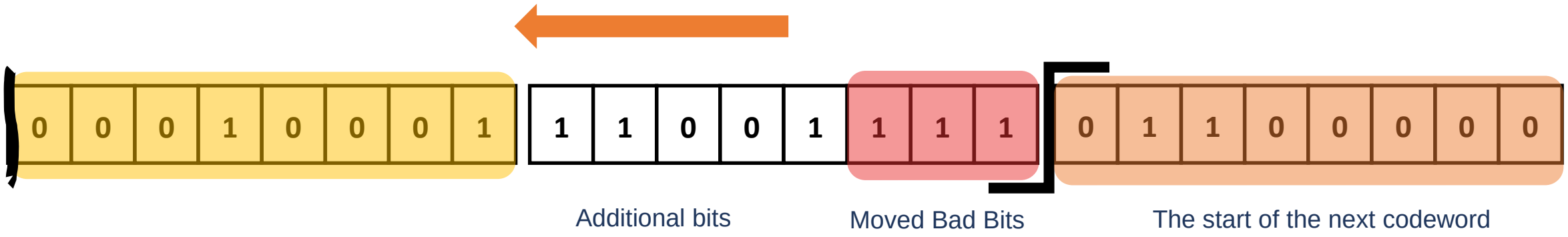
Controller works with Bad bits only on codeword level
That means each codeword always starts from integer byte

To remove Bad bits at first required to set page structure (define Data areas)

Using Badbit Column remover element (

) mark Bad Bit positions on the dump to remove them

As a result marked Bad bits will move to the end of Codeword, revealing actual data written by controller



Removing Bad bits

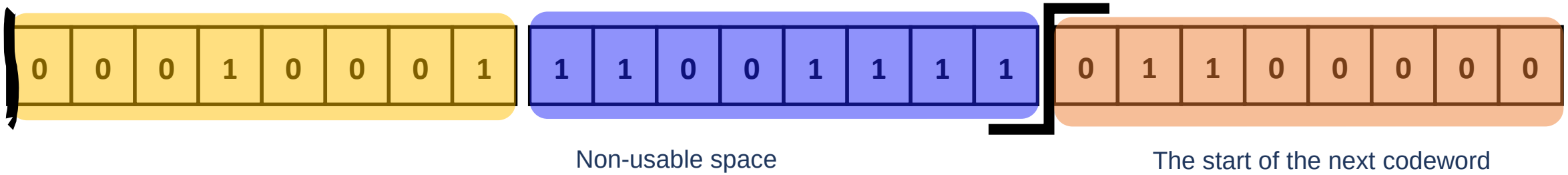
As a result of Bad bit removal, at the end of codewords the additional full bytes of non-usable space appears
That causes next codeword shift and prevents automatic XOR and ECC determinition

To remove Bad bits at first required to set page structure (define Data areas)

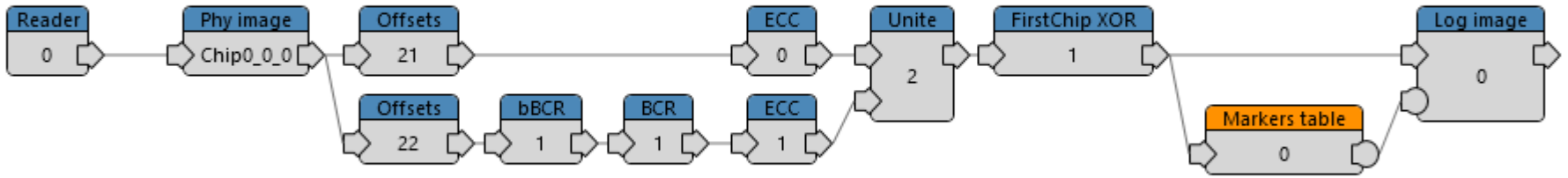
Using BadColumn remover element (

) these bytes colud be also removed

As a result after Bad Bit Column removal and Bad Column removal further steps for case solution could be applied for successfull data recovery



Overal view of FC case solution with Bad Bits



Every case with FC controller the XOR key is unique and must be generated with unique VNR's AI-XOR element

After ECC correction and XOR key generating and applying Logical image could be build with Markers table to recover actual Data from storage device