Coreboot – Raminit

DDR3 memory initialization basics on Intel Sandy Bridge platforms

By Patrick Rudolph

Coreboot – Raminit

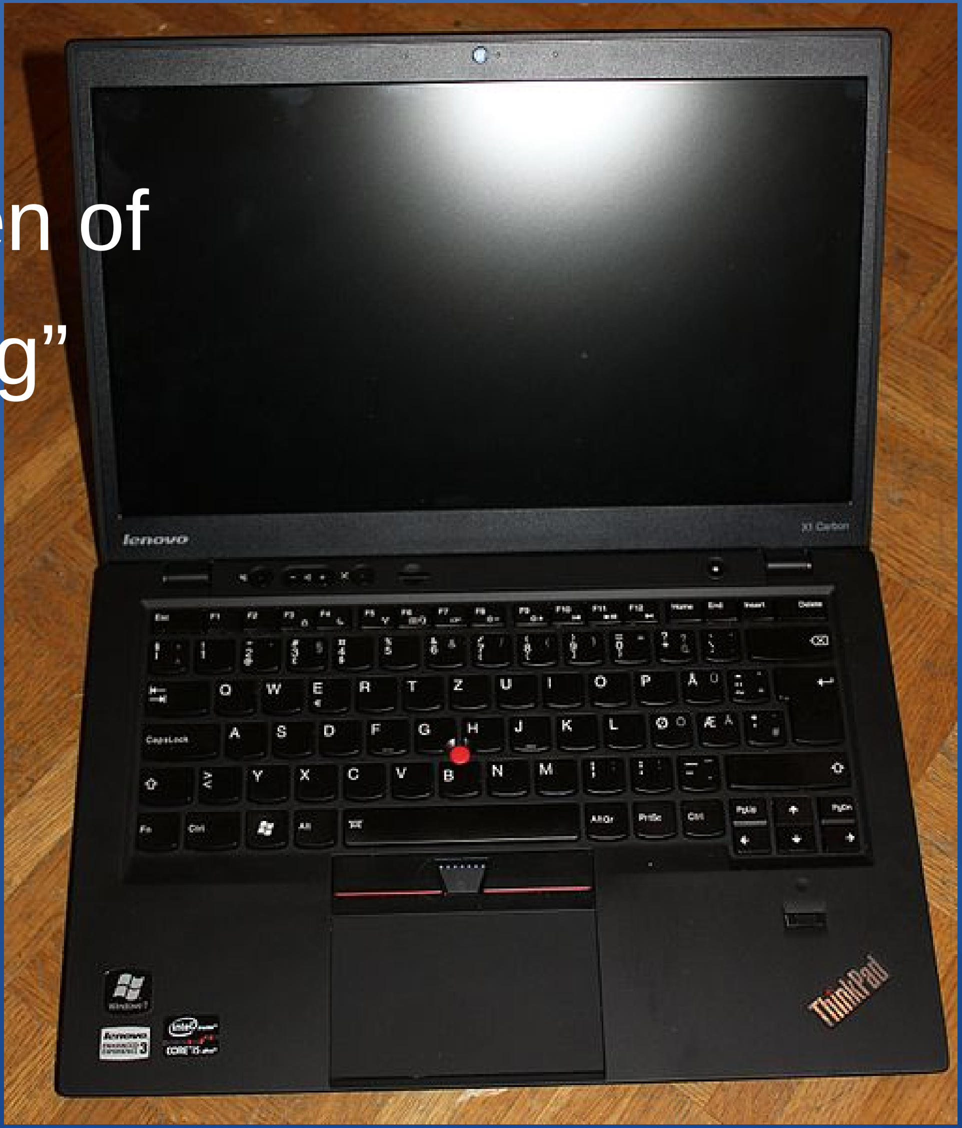
Who am I ?

- B.Sc. Engineer in Electronics and Information Technologies @ RUB
- Working as Linux Admin/systems integrator
- coreboot developer since 2015

Coreboot – Raminit

First contact

coreboot’s black screen of “something went wrong”



Picture Source: By Mariofan13 (Own work) CC BY-SA 3.0

Coreboot – Raminit

Capters

Coreboot – Raminit

History

Coreboot – Raminit

Initial native raminit was done by:

1) Damien Zammit
2) Vladimir Serbinenko

Coreboot – Raminit

MRC vs Native raminit

- Raminit is done by Memory Reference Code (MRC)
- Reverse Engineered using Serialice
- Register accesses decoded with MRS register documentation
- Algorithms can finally be documented

Coreboot – Raminit

MRC

- Blob (Closed source)
- NDA required
- Depends on metadata (CAR, SPD, processor operating mode, ...)
- Does chipset initialization
- Memory test
- Stack setup
- Firmware shadow

- Visual Studio C/C++
- Written in C
- 32bit protected mode
- Compile time support for mobile/desktop platforms

Coreboot – Raminit

Native Raminit

- Open Source
- Not very well documented (yet)
- No chipset initialization
- Memory test (very basic)
- Allows to gather details about hardware

- Written in C
- 32bit protected mode
- GCC / clang

Coreboot – Raminit

Native Raminit features

Coreboot – Raminit

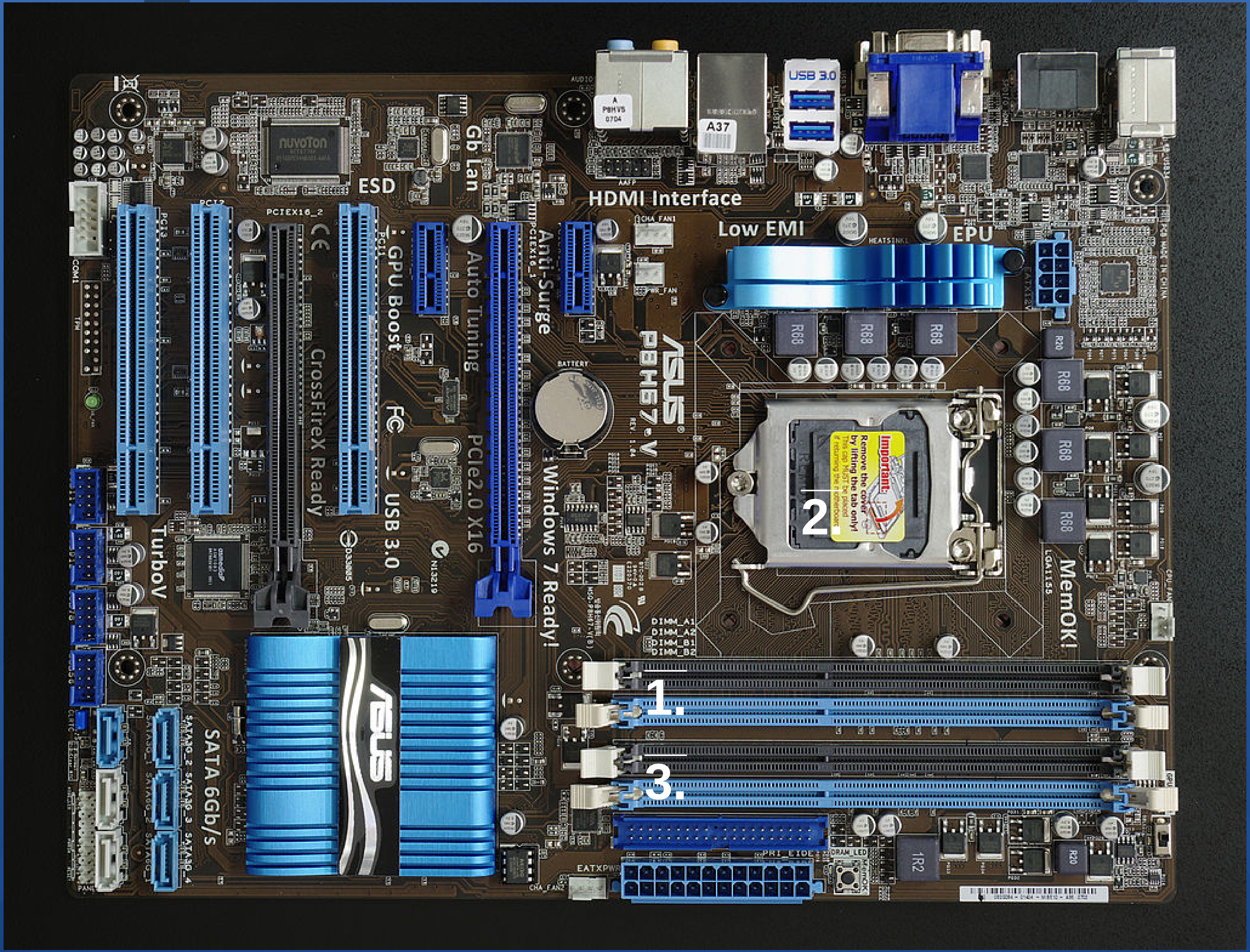
Native Raminit features

- Support for MRC cache
- Support for XMP profiles
- Failsafe by disabling one channel
- Beep on death (Lenovo only)

Coreboot – Raminit

Definitions

Coreboot – Raminit



Picture Source: By smial (talk) - Own work, FAL, https://commons.wikimedia.org/w/index.php?curid=14636011

Coreboot – Raminit

Definition:

1)Slot (Channel 0)

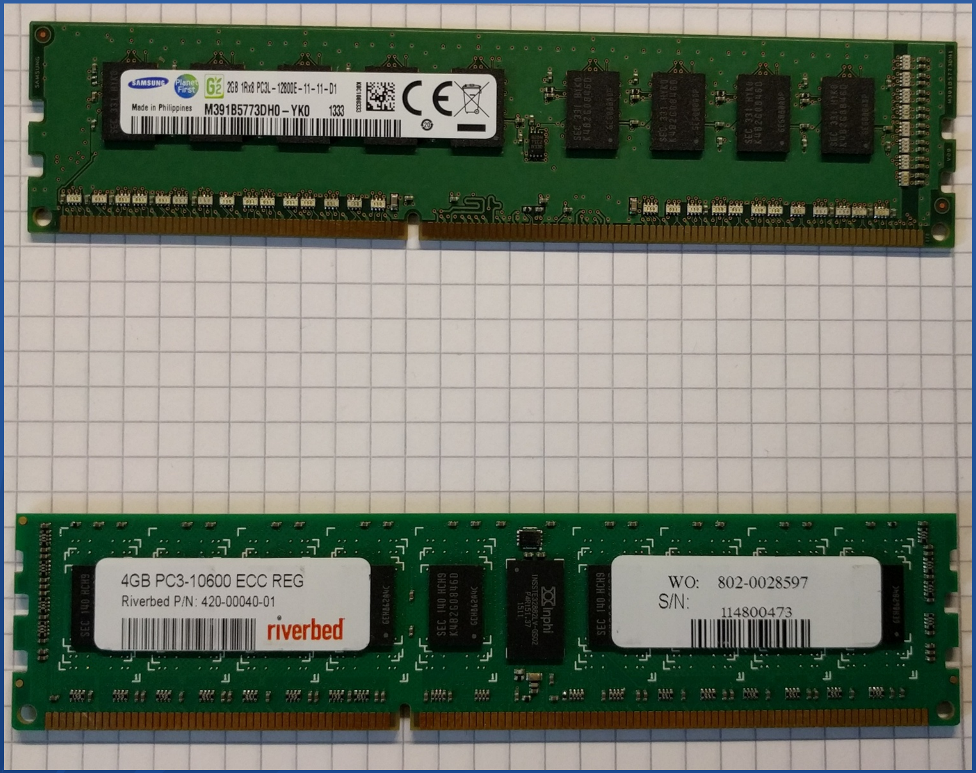
2)CPU / Memory controller

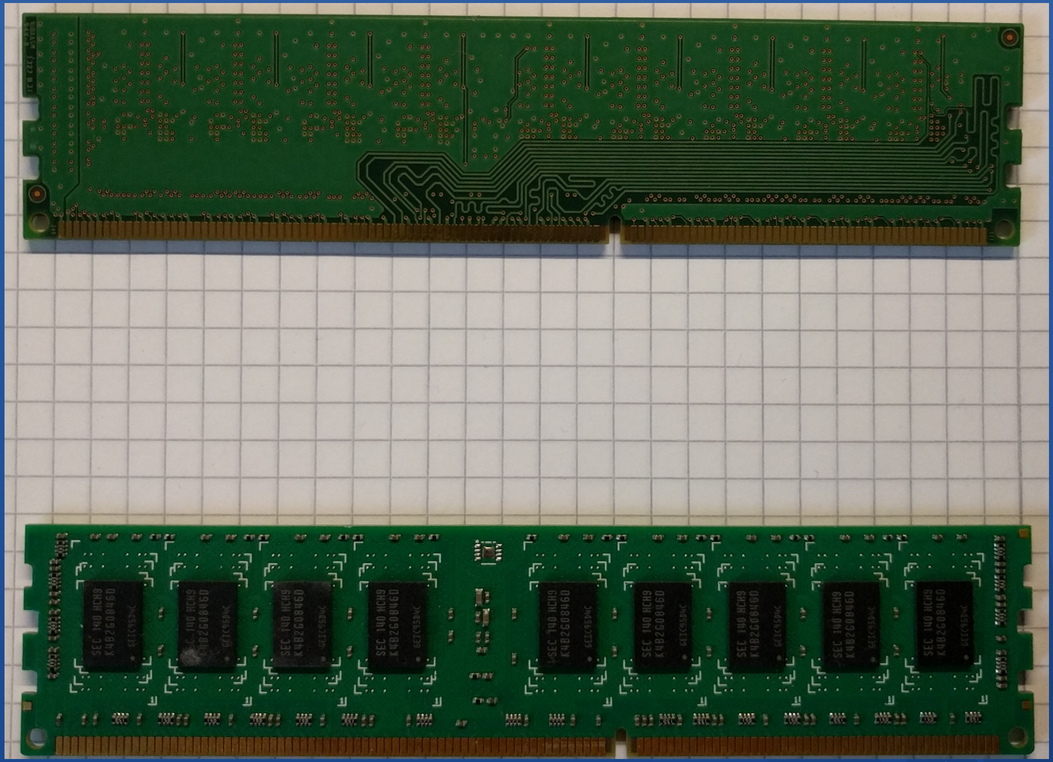
3)Slot (Channel 1)

- Each channel supports up to two slots
- Each slots supports up to two ranks

Coreboot – Raminit

Single vs Dual Rank DIMM





Picture Source: By Patrick Rudolph

Coreboot – Raminit

Memory Rank:

Group of DRAM chips that share

1) Chip select (CKE)
2) On Die Termination (ODT)

Ranks can’t be access simultaneously as:

3) Share DATA
4) Share CMD / ADDR

Coreboot – Raminit

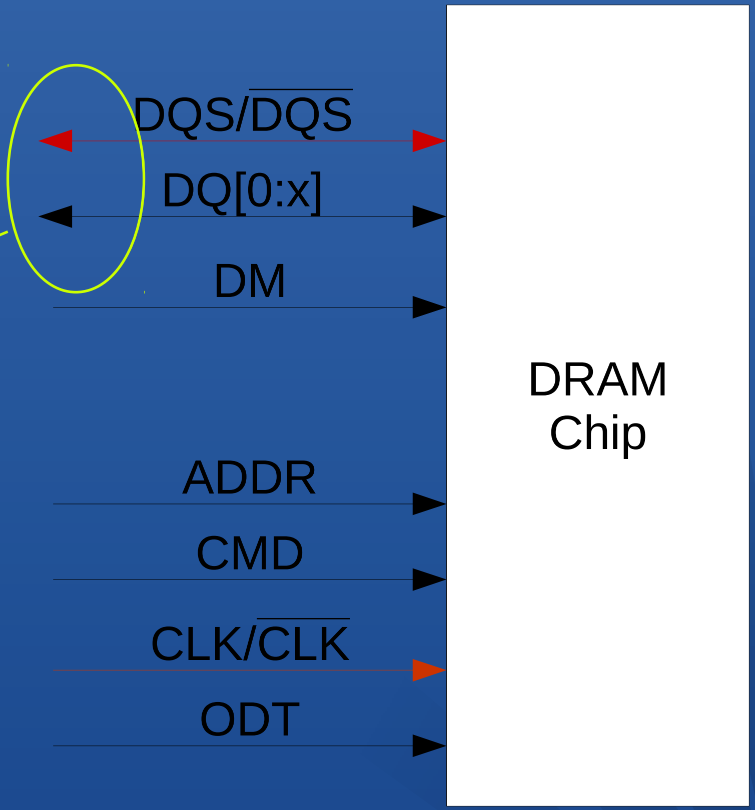
DRAM Chip:

DQS: Data Strobe, bidirectional

DQ: Data, bidirectional, Width x4, x8, x16

Definition Lane:

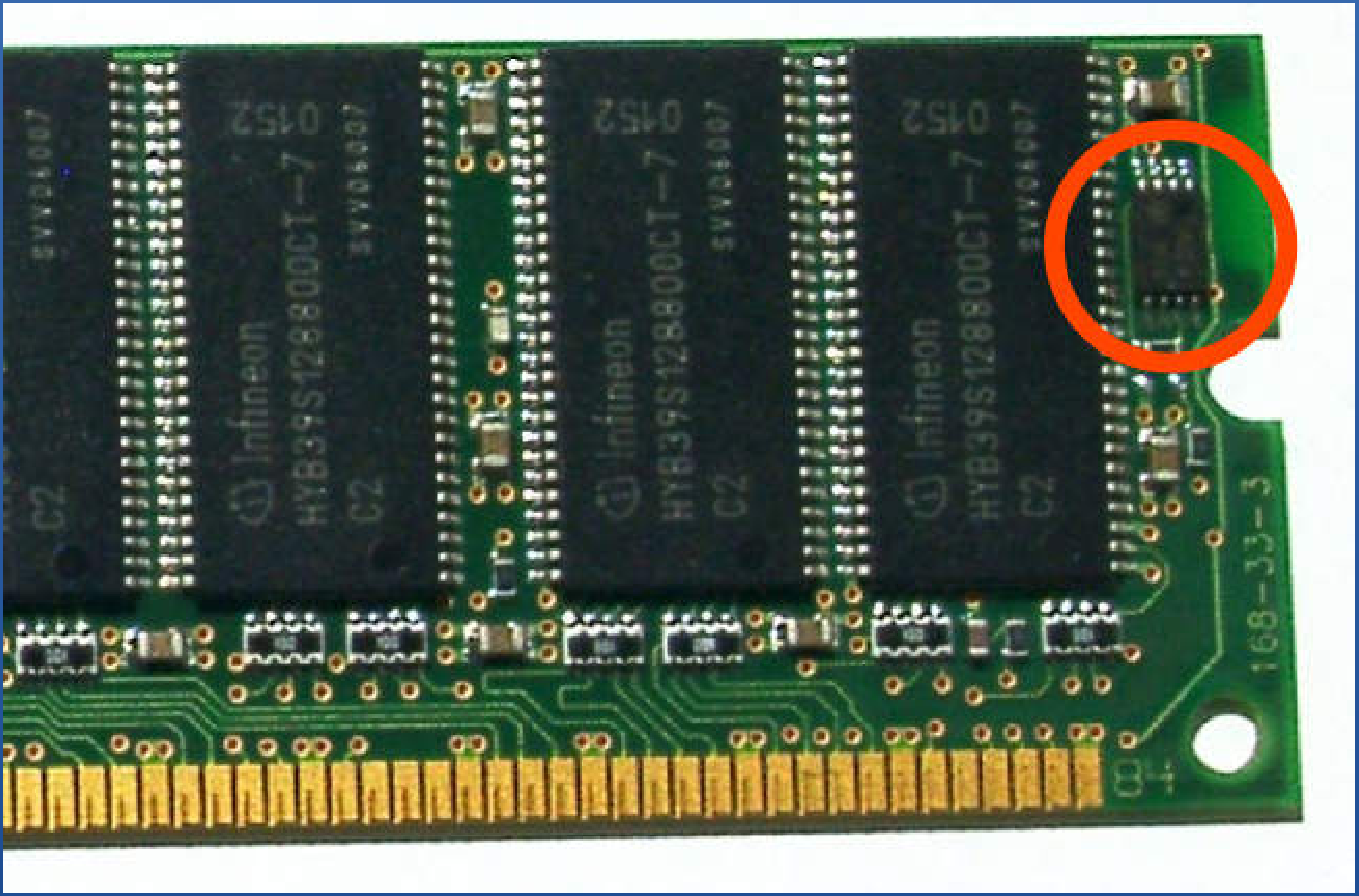
Group of DQS/$\overline{\text{DQS}}$ and DQ[0:x]



Coreboot – Raminit

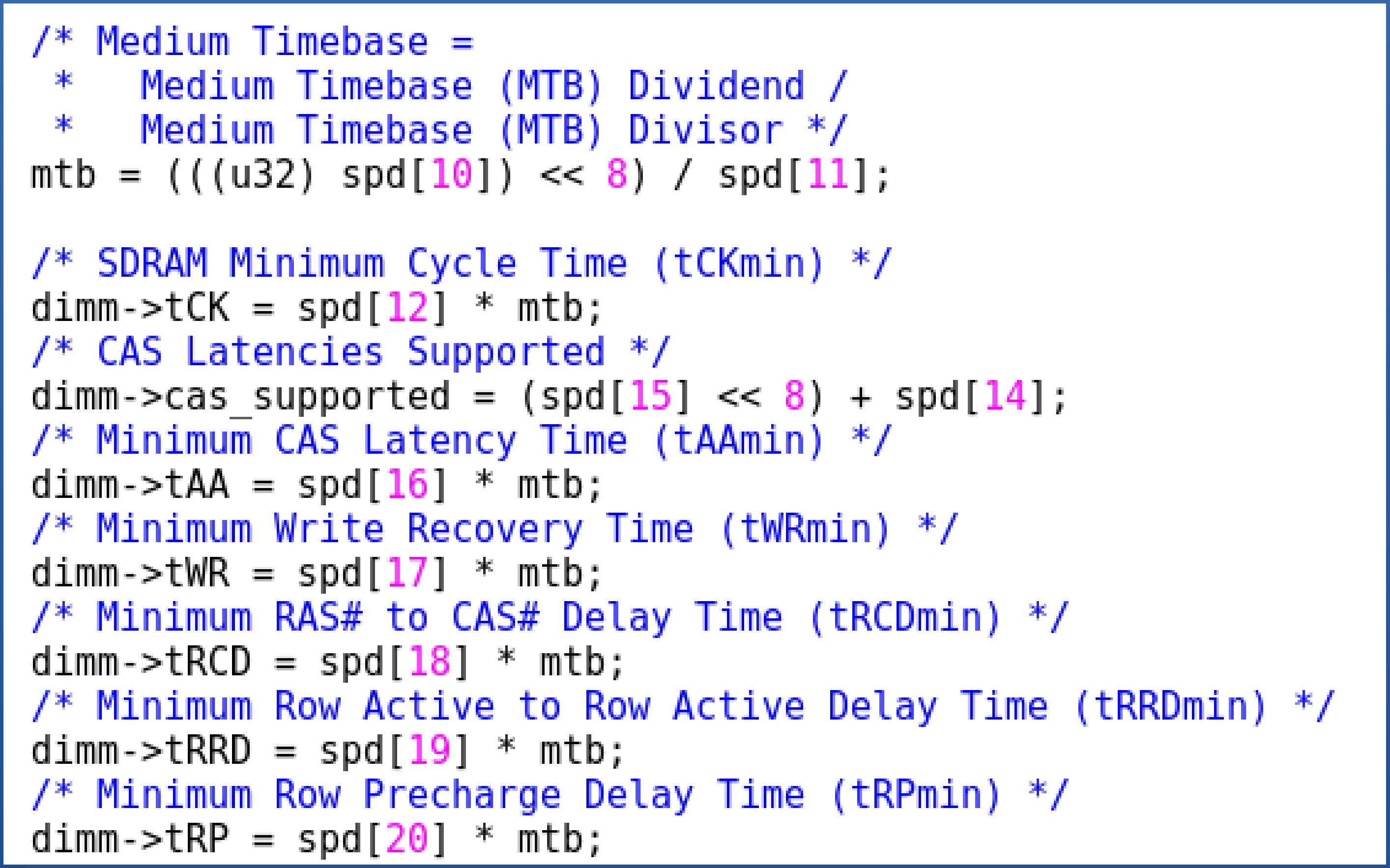
Read and decode SPD

Coreboot – Raminit



Picture Source: Pemu (SPD_SDRAM.svg) [CC-BY-SA-3.0 or GFDL]

Coreboot – Raminit



```
/* Medium Timebase =
 *   Medium Timebase (MTB) Dividend /
 *   Medium Timebase (MTB) Divisor */
mtb = (((u32) spd[10]) << 8) / spd[11];

/* SDRAM Minimum Cycle Time (tCKmin) */
dimm->tCK = spd[12] * mtb;
/* CAS Latencies Supported */
dimm->cas_supported = (spd[15] << 8) + spd[14];
/* Minimum CAS Latency Time (tAAmin) */
dimm->tAA = spd[16] * mtb;
/* Minimum Write Recovery Time (tWRmin) */
dimm->tWR = spd[17] * mtb;
/* Minimum RAS# to CAS# Delay Time (tRCDmin) */
dimm->tRCD = spd[18] * mtb;
/* Minimum Row Active to Row Active Delay Time (tRRDmin) */
dimm->tRRD = spd[19] * mtb;
/* Minimum Row Precharge Delay Time (tRPmin) */
dimm->tRP = spd[20] * mtb;
```

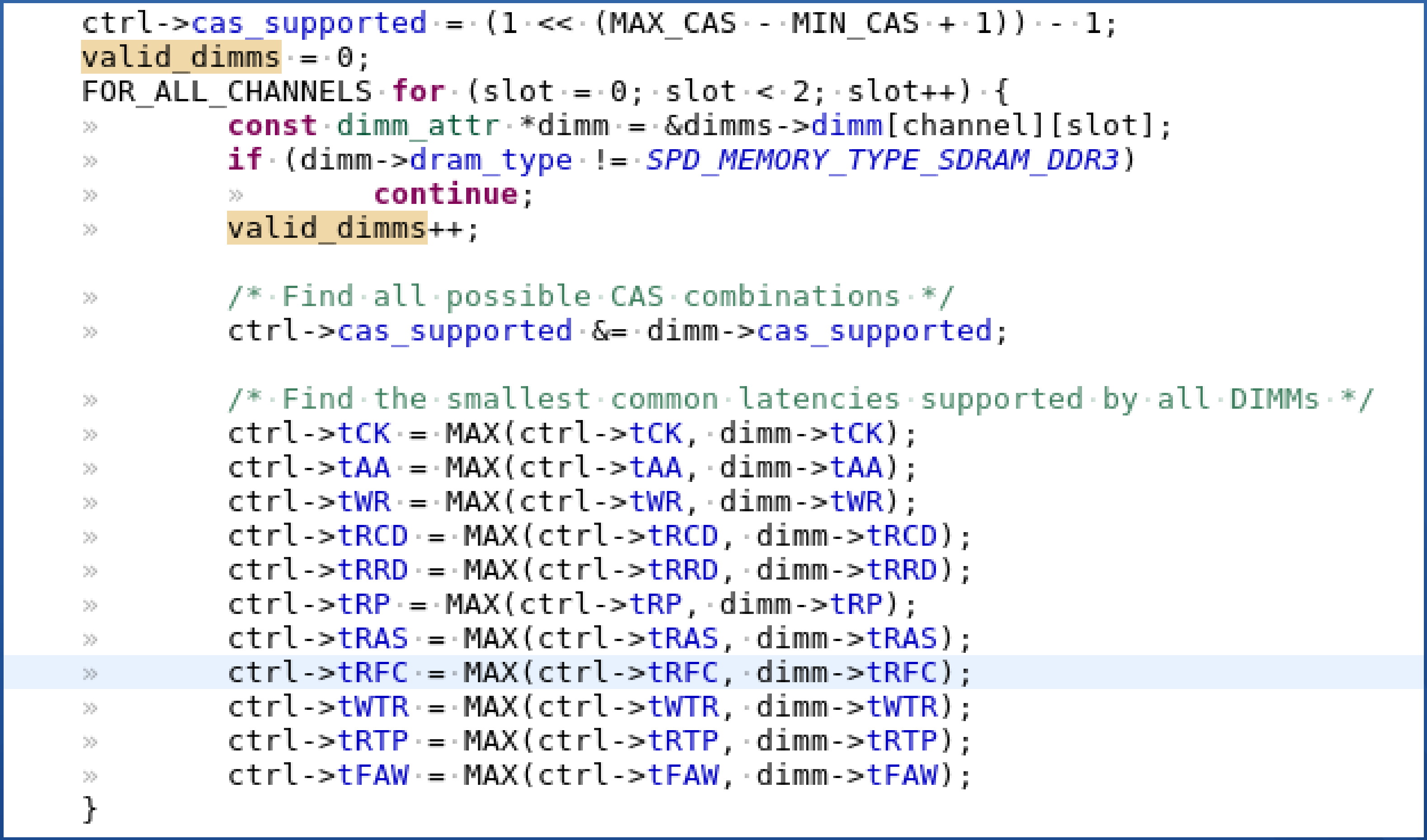
Picture Source: Coreboot src/device/dram/ddr3.c

Coreboot – Raminit

Finding common timings

Coreboot – Raminit

Find common timings



```
ctrl->cas_supported = (1 << (MAX_CAS - MIN_CAS + 1)) - 1;
valid_dimms = 0;
FOR_ALL_CHANNELS for (slot = 0; slot < 2; slot++) {
	const dimm_attr *dimm = &dimms->dimm[channel][slot];
	if (dimm->dram_type != SPD_MEMORY_TYPE_SDRAM_DDR3)
		continue;
	valid_dimms++;

	/* Find all possible CAS combinations */
	ctrl->cas_supported &= dimm->cas_supported;

	/* Find the smallest common latencies supported by all DIMMs */
	ctrl->tCK = MAX(ctrl->tCK, dimm->tCK);
	ctrl->tAA = MAX(ctrl->tAA, dimm->tAA);
	ctrl->tWR = MAX(ctrl->tWR, dimm->tWR);
	ctrl->tRCD = MAX(ctrl->tRCD, dimm->tRCD);
	ctrl->tRRD = MAX(ctrl->tRRD, dimm->tRRD);
	ctrl->tRP = MAX(ctrl->tRP, dimm->tRP);
	ctrl->tRAS = MAX(ctrl->tRAS, dimm->tRAS);
	ctrl->tRFC = MAX(ctrl->tRFC, dimm->tRFC);
	ctrl->tWTR = MAX(ctrl->tWTR, dimm->tWTR);
	ctrl->tRTP = MAX(ctrl->tRTP, dimm->tRTP);
	ctrl->tFAW = MAX(ctrl->tFAW, dimm->tFAW);
}
```

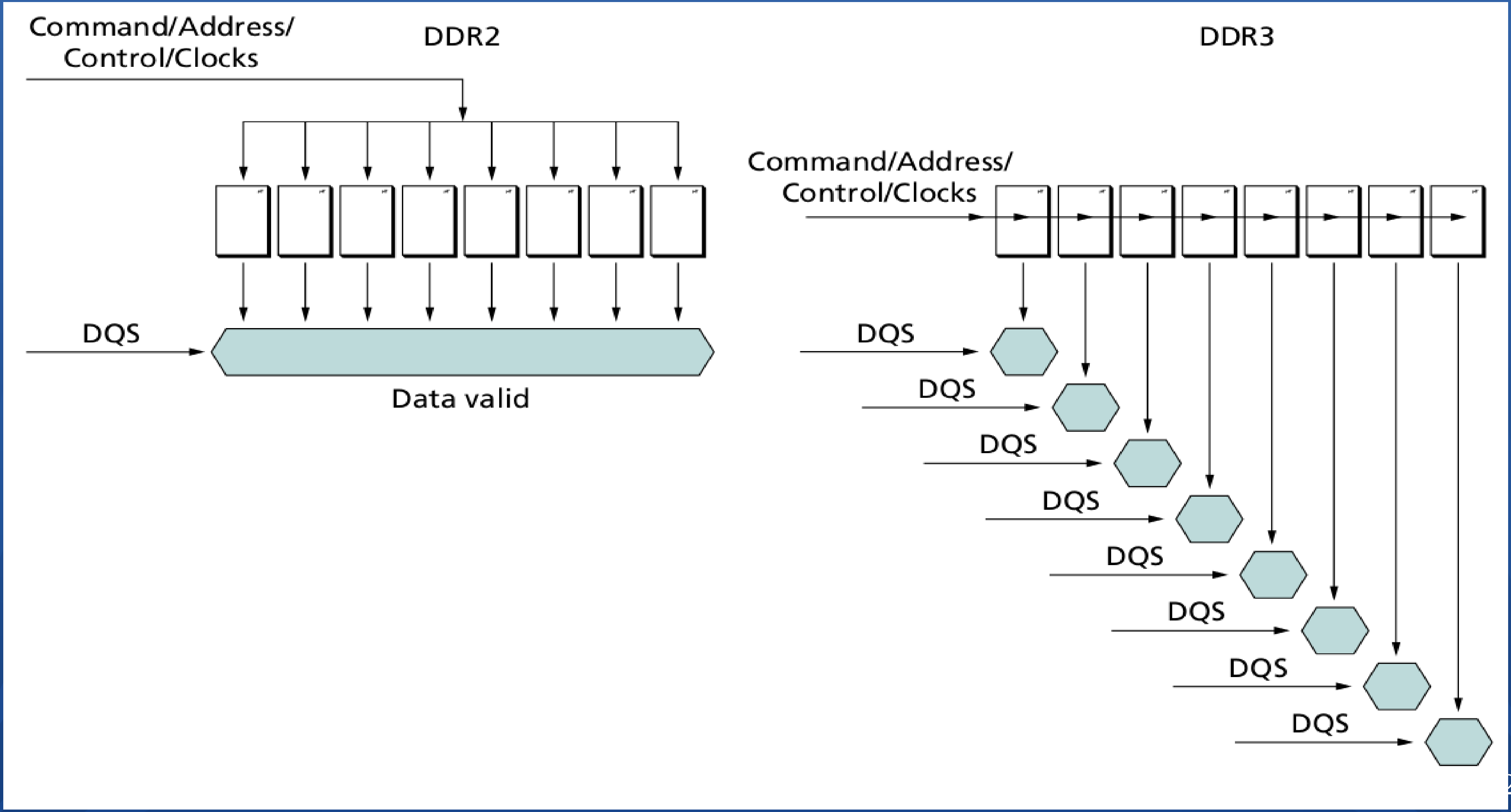
Source: Coreboot src/northbridge/intel/sandybridge/raminit_common.c

Coreboot – Raminit

Read training

Coreboot – Raminit

DDR3 Flyby design



Picture Source: Micron TN-41-13: DDR3 Point-to-Point Design Support

Coreboot – Raminit

DDR2

- High resistive DC load
- Variable load
- Difficult routing design
- High noise due to reflections on splicing
- No training required

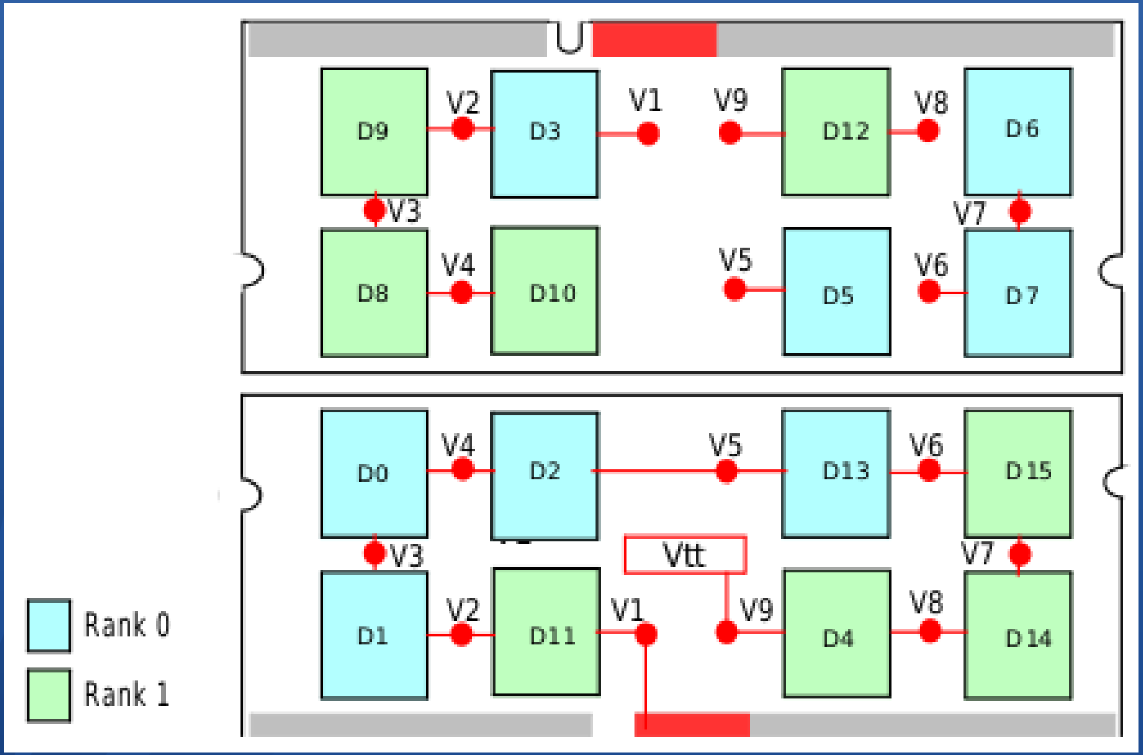
DDR3

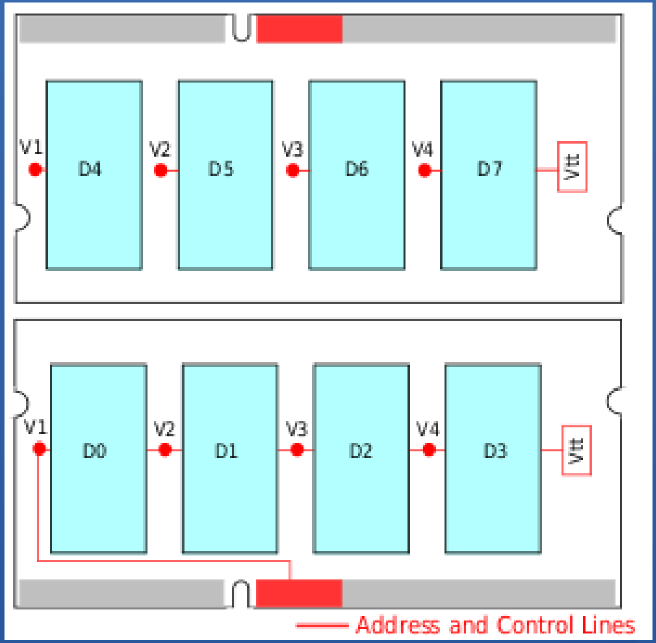
- Low resistive DC load
- Constant load
- Easy routing
- Low noise
- Requires additional training sequence (in software)

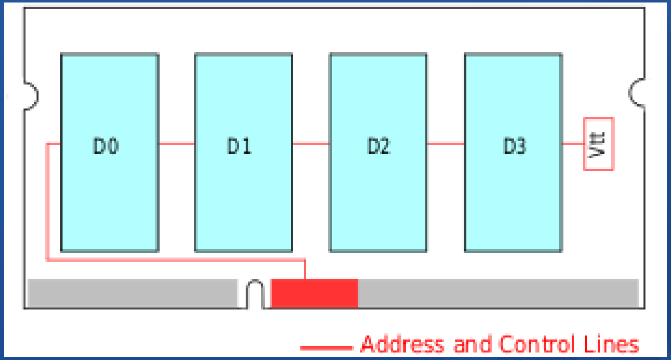
Coreboot – Raminit

6 Reference Design

- Single / Dual Rank



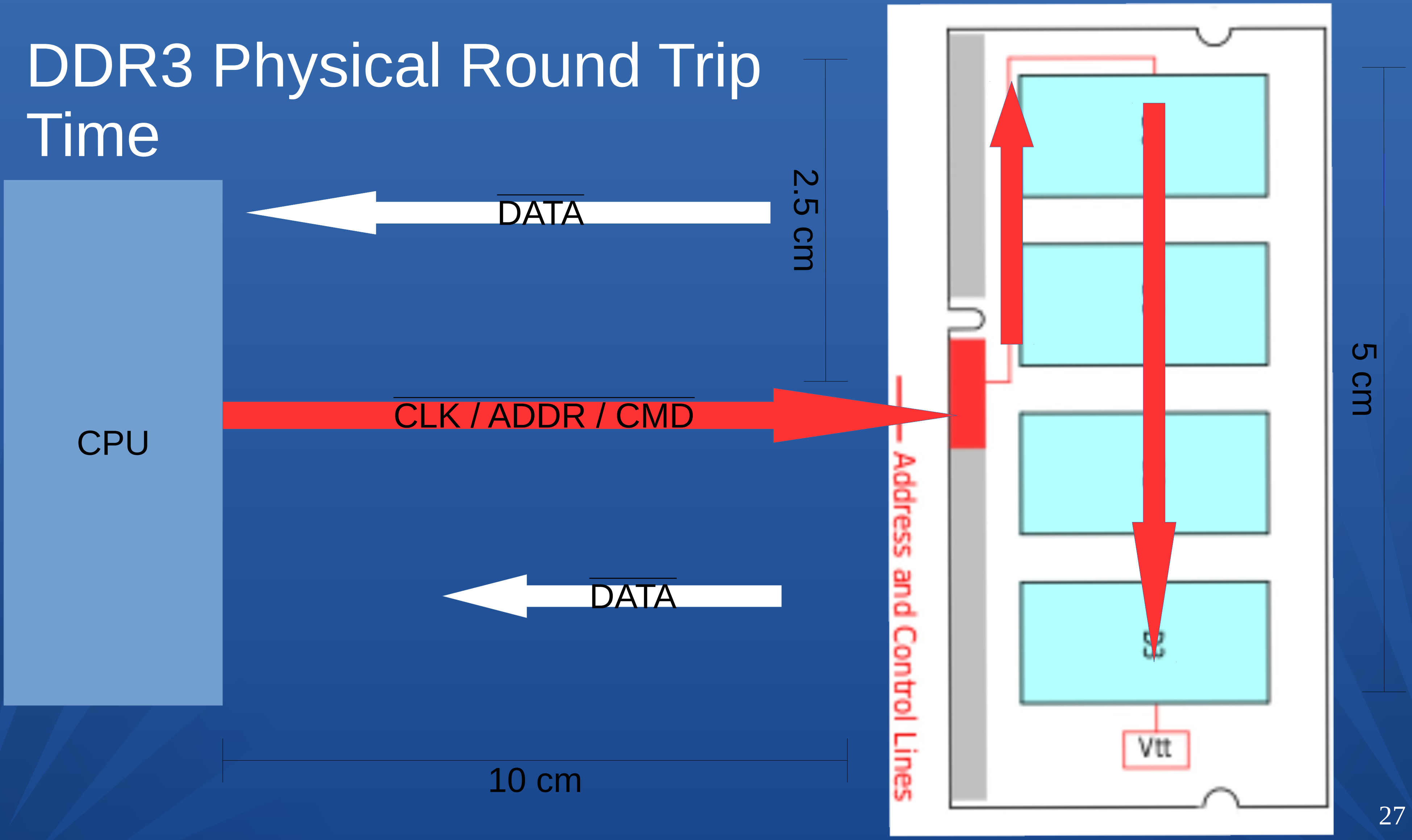




Picture Source: Hynix:DDR3 SDRAM Unbuffered SODIMMs Based on 2Gb B-die

Coreboot – Raminit

DDR3 Physical Round Trip Time



Picture Source: By Patrick Rudolph

Coreboot – Raminit

Signal propagation in FR4:

Trace length:

10cm + 2,5cm + 0cm + 10cm = 22,5cm = 0,225 m

10cm + 2,5cm + 5cm + 10cm = 27,5cm = 0,275 m

Signal velocity of propagation in FR4 ~ ½ SOL

v(FR4) ~ ½ * 300.000 km/s = ½ * 300.000.000 m/s

0,225 m / (½ * 300.000.000 m/s) = 1,50ns

0,275 m / (½ * 300.000.000 m/s) = 1,83ns

Coreboot – Raminit

Signal propagation in FR4:

DDR3 clock: 800Mhz

DDR3 DCLK: 1600Mhz → 625 ps

Signal propagation delay

1,50 ns / 625 ps = 2,4 DCLK

1,83 ns / 625 ps = 2,93 DCLK

Not that synchronous at all...

Coreboot – Raminit

DDR3 Round Trip Time

Required for Memory Controller to switch DQ from Tri-State to Input

Physical Round Trip Time (RTT)

CAS Latency (CL)

Phase compensation blocks (PLL)

Delay compensation blocks (DLL)

→ 55 DCLKs

Coreboot – Raminit

Read Training:

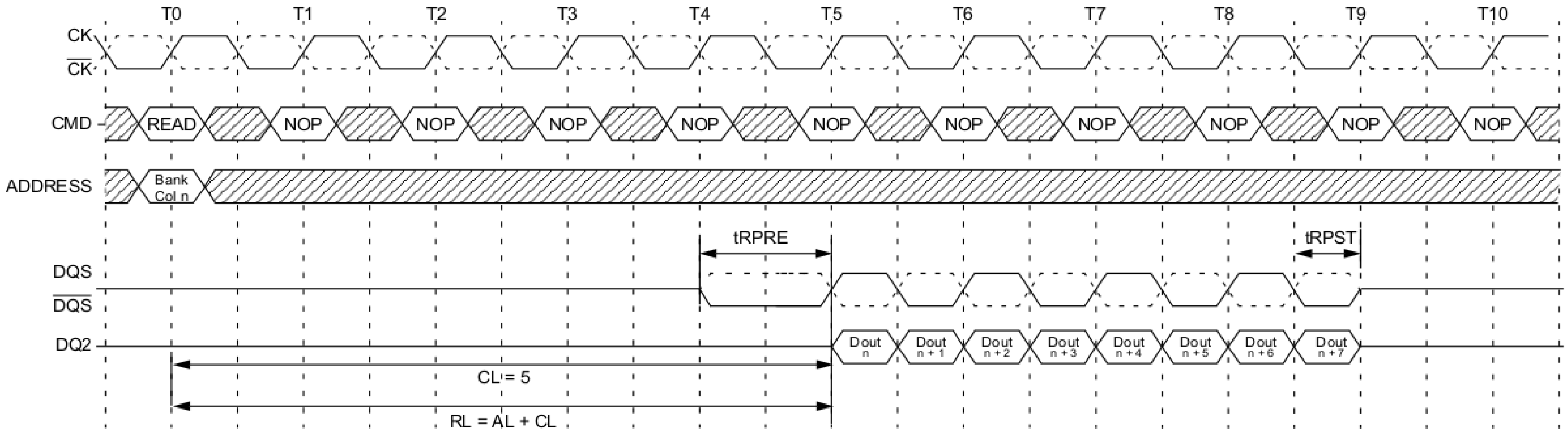
- Special DRAM mode
- Sends a predefined pattern
- Memory controller synchronizes to preamble
- But ...

Coreboot – Raminit

Read BURST Operation

2.13.1 READ Burst Operation

During a READ or WRITE command, DDR3 will support BC4 and BL8 on the fly using address A12 during the READ or WRITE (AUTO PRECHARGE can be enabled or disabled).
A12 = 0, BC4 (BC4 = burst chop, tCCD = 4)
A12 = 1, BL8
A12 is used only for burst length control, not as a column address.



NOTE :
1. BL8, RL = 5, AL = 0, CL = 5.
2. DOUT n = data-out from column n.
3. NOP commands are shown for ease of illustration; other commands may be valid at these times.
4. BL8 setting activated by either MR0[A1:0 = 00] or MR0[A1:0 = 01] and A12 = 1 during READ command at T0.

Figure 22. READ Burst Operation RL = 5 (AL = 0, CL = 5, BL8)

Picture Source: Samsung DDR3 Device Operations Rev1.4_Nov11.book

Coreboot – Raminit

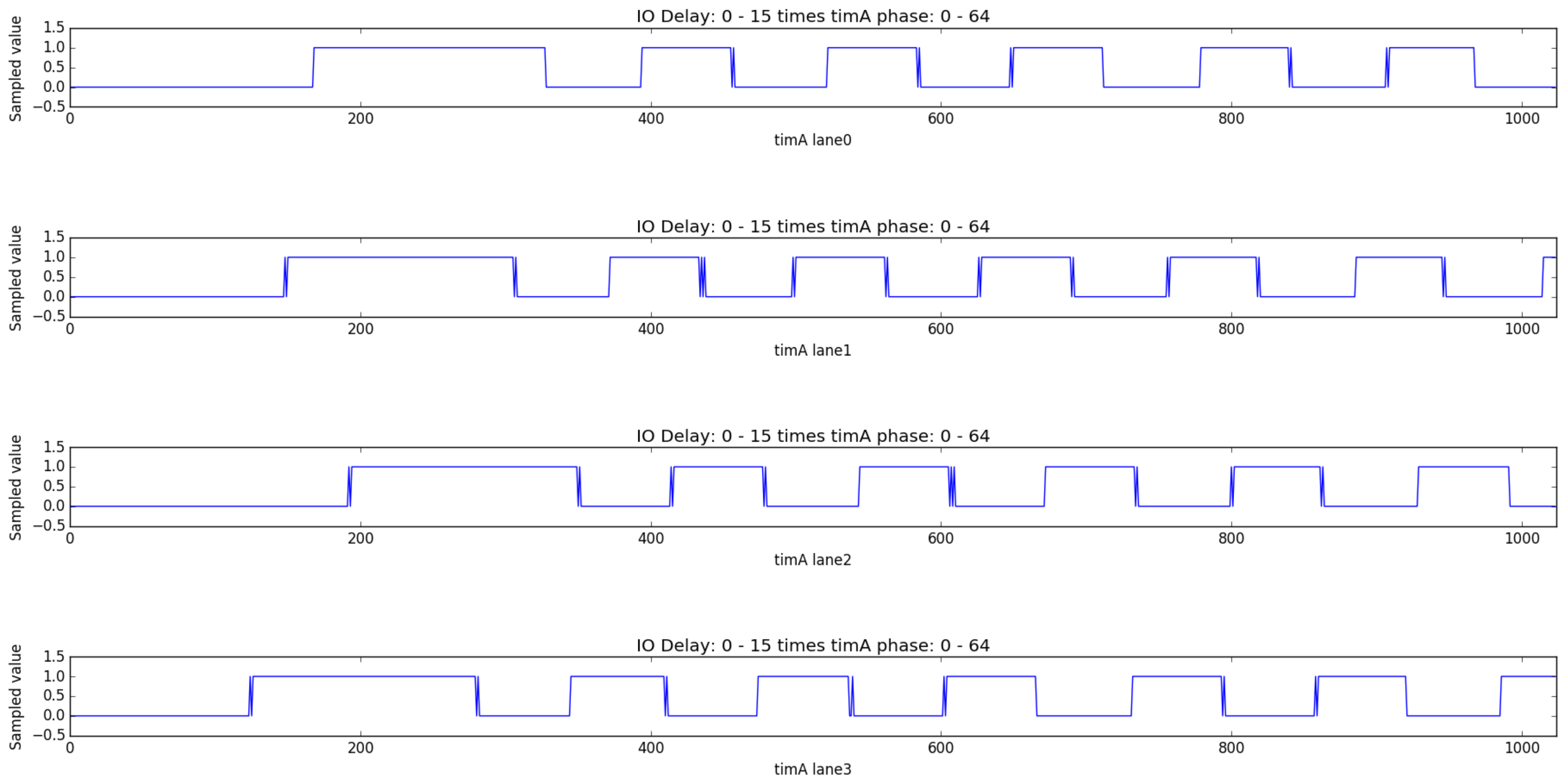
DRAM has to be cheap:

- No PLL integrated
- No ¼ phase shift possible
- DQS needs to be ¼ phase shifted by memory controller

Coreboot – Raminit

Real world measurements

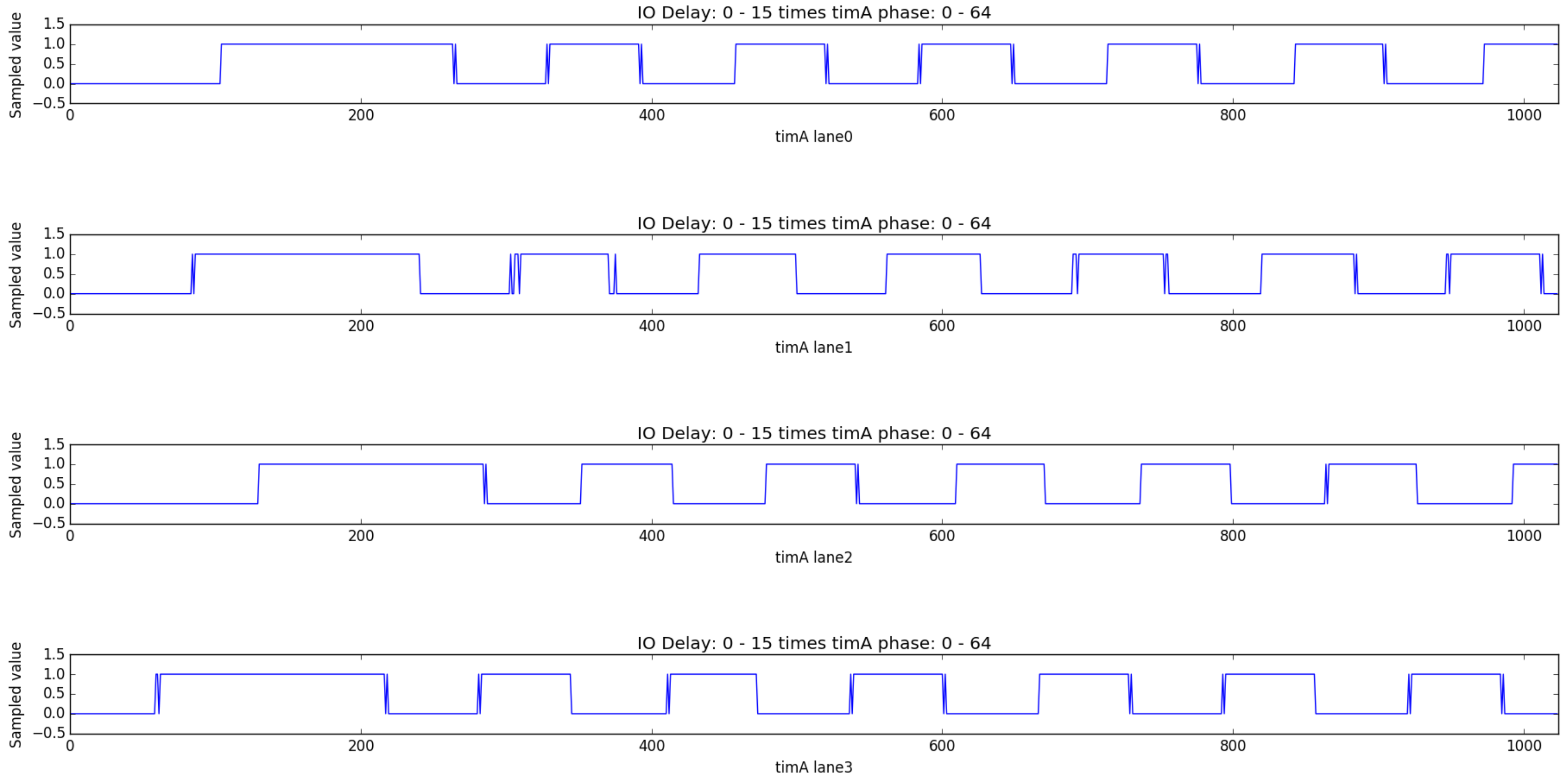
Coreboot – Raminit



Read Training, RTT: 53

Picture Source: By Patrick Rudolph

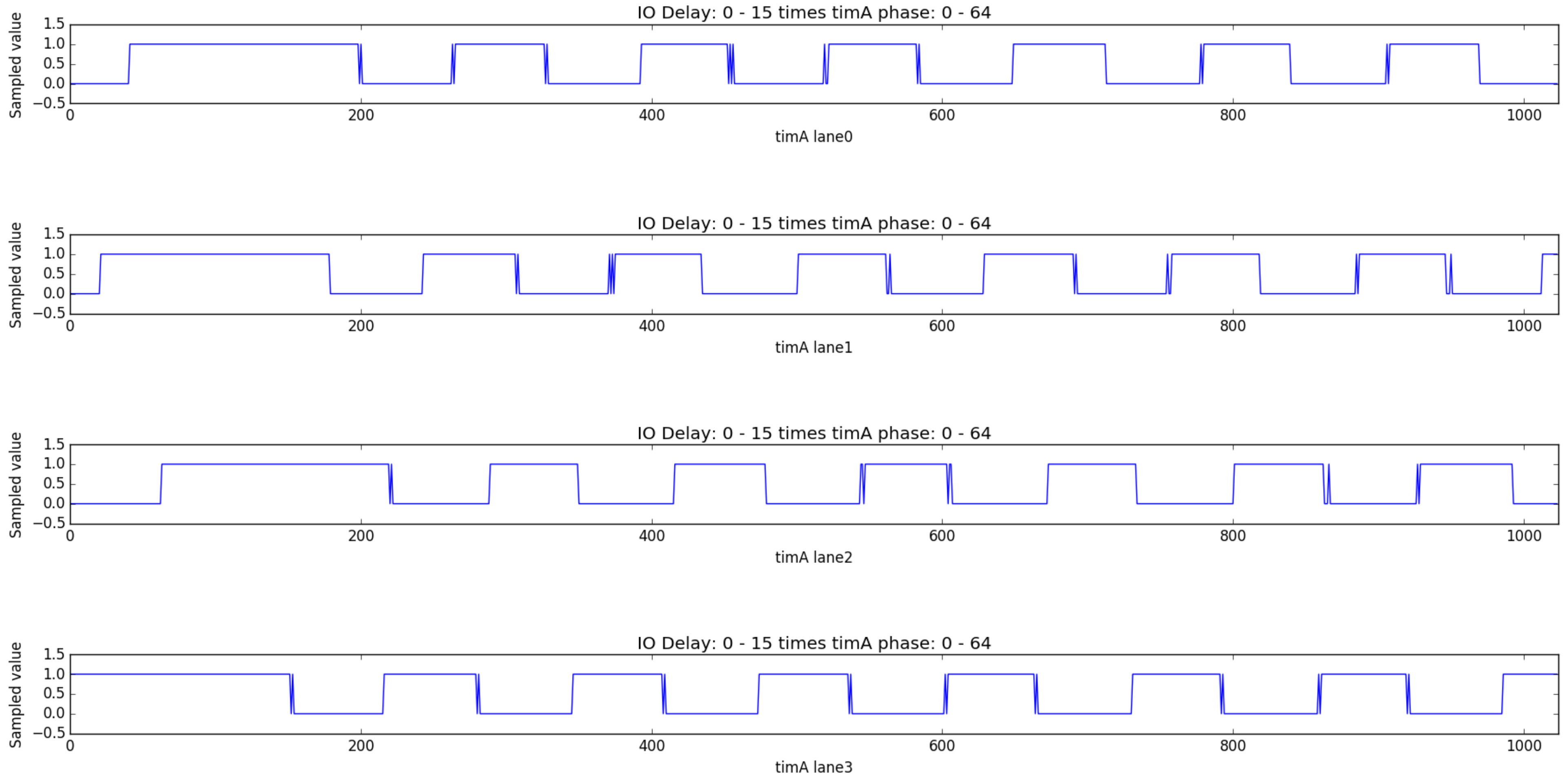
Coreboot – Raminit



Read Training, RTT: 54

Picture Source: By Patrick Rudolph

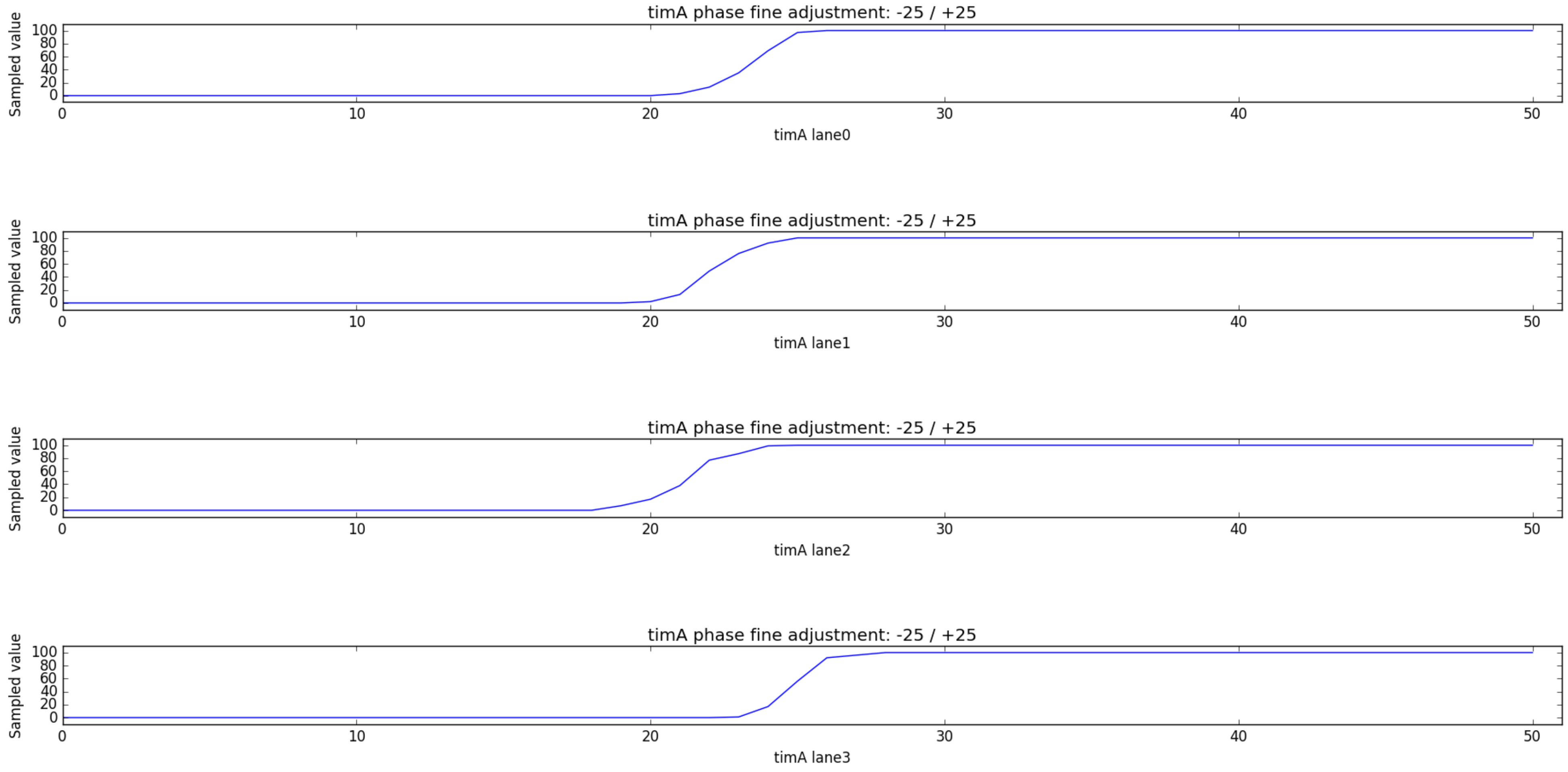
Coreboot – Raminit



Read Training, RTT: 55

Picture Source: By Patrick Rudolph

Coreboot – Raminit



Read Training, End of Preamble

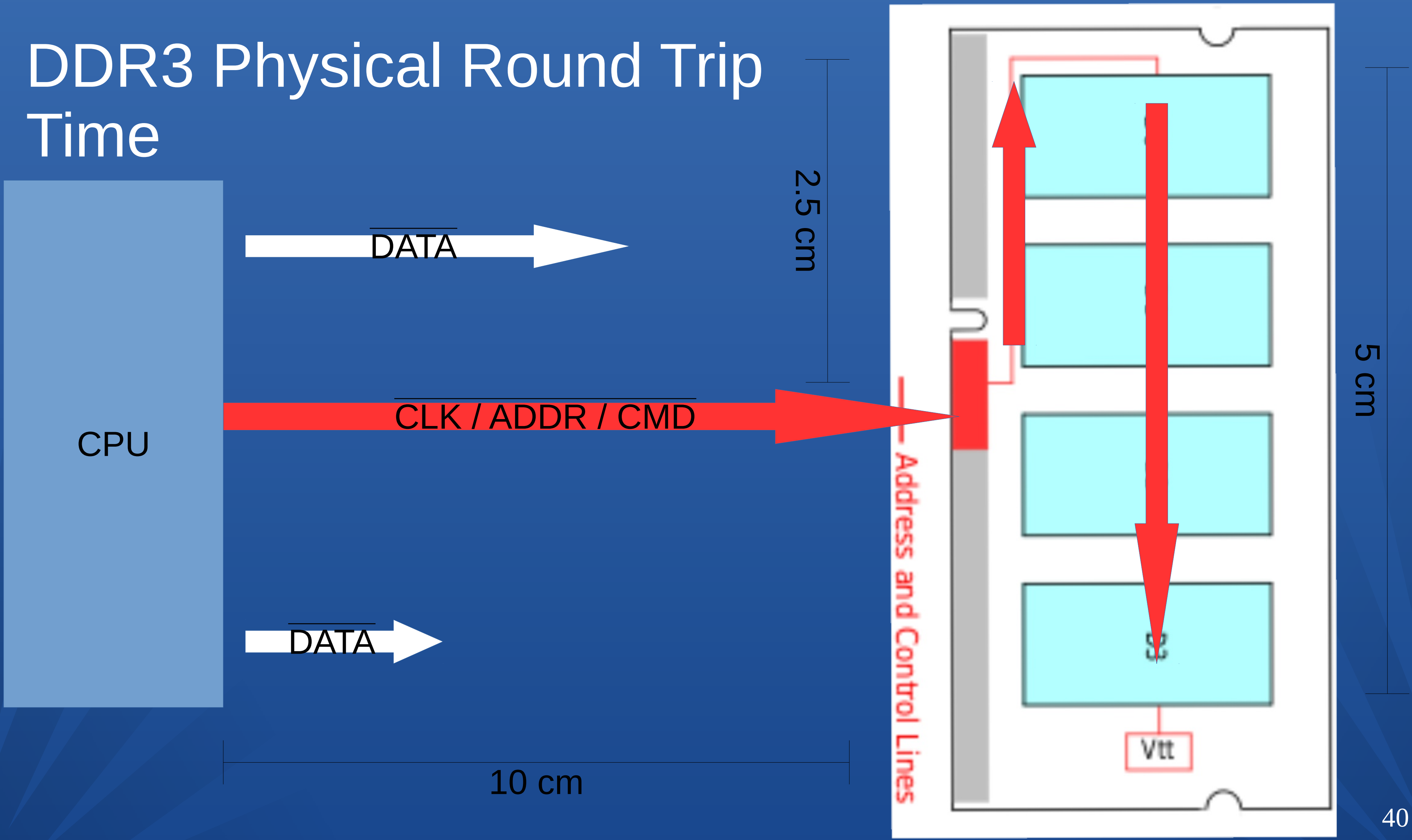
Picture Source: By Patrick Rudolph

Coreboot – Raminit

Write training

Coreboot – Raminit

DDR3 Physical Round Trip Time



Picture Source: Patrick Rudolph

Coreboot – Raminit

Signal propagation in FR4:

Trace length:

10cm + 2,5cm + 0cm – 10cm = 02,5cm = 0,025 m

10cm + 2,5cm + 5cm – 10 cm = 07,5cm = 0,075 m

Signal velocity of propagation in FR4 ~ ½ SOL

v(FR4) ~ ½ * 300.000 km/s = ½ * 300.000.000 m/s

0,025 m / (½ * 300.000.000 m/s) = 0,16ns

0,075 m / (½ * 300.000.000 m/s) = 0,50ns

Coreboot – Raminit

Signal propagation in FR4:

DDR3 clock: 800Mhz

DDR3 DCLK: 1600Mhz → 625 ps

Signal propagation delay

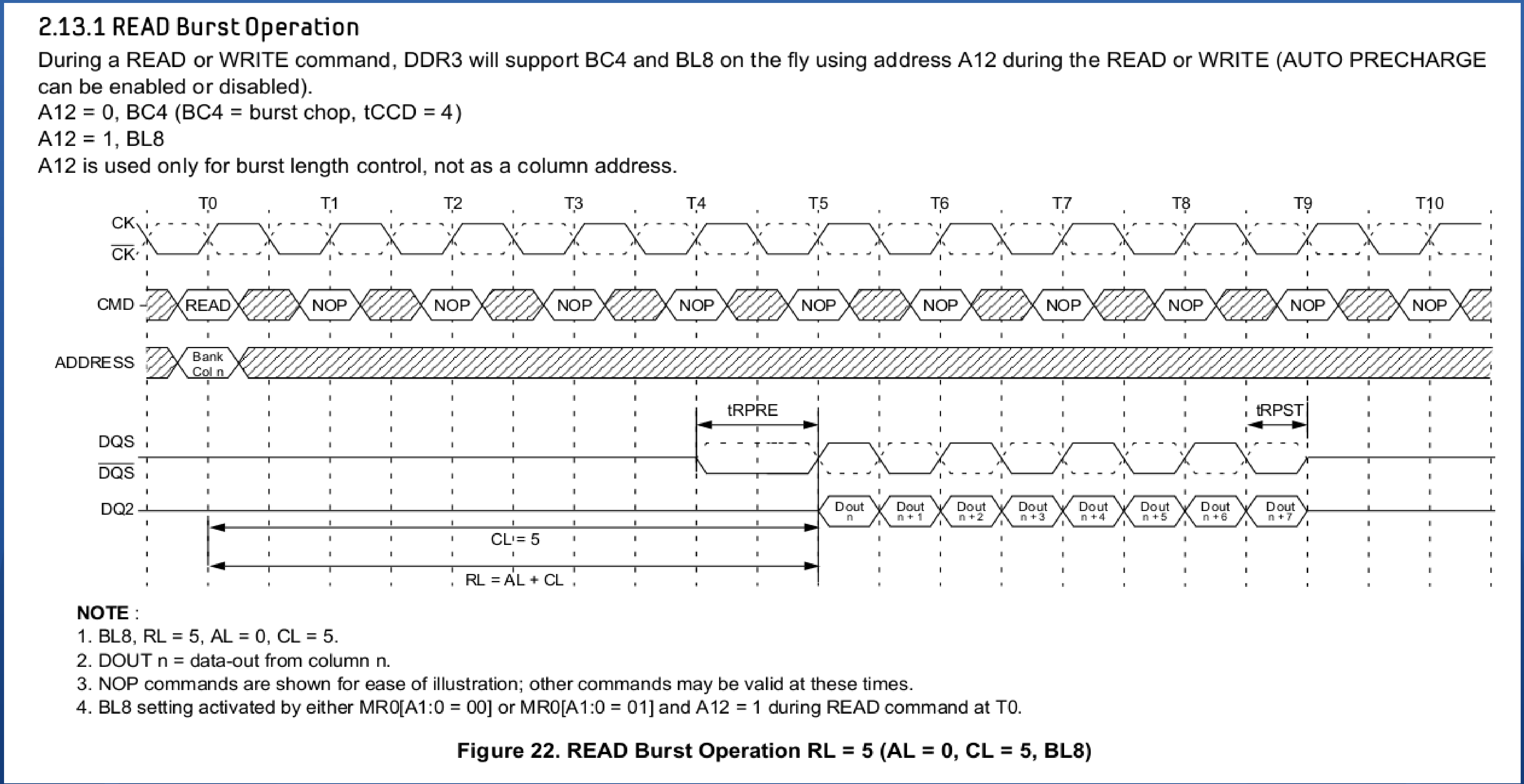
0,16 ns / 625 ps = 0,25 DCLK

0,50 ns / 625 ps = 0,8 DCLK

Not that synchronous at all...

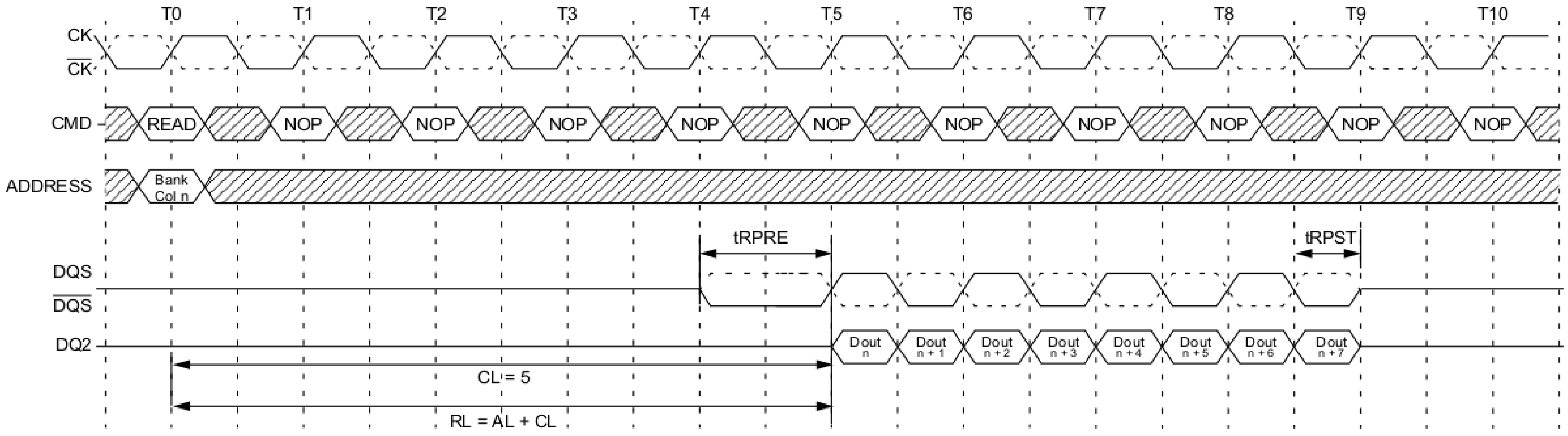
Coreboot – Raminit

Read BURST Operation



2.13.1 READ Burst Operation

During a READ or WRITE command, DDR3 will support BC4 and BL8 on the fly using address A12 during the READ or WRITE (AUTO PRECHARGE can be enabled or disabled).
A12 = 0, BC4 (BC4 = burst chop, tCCD = 4)
A12 = 1, BL8
A12 is used only for burst length control, not as a column address.



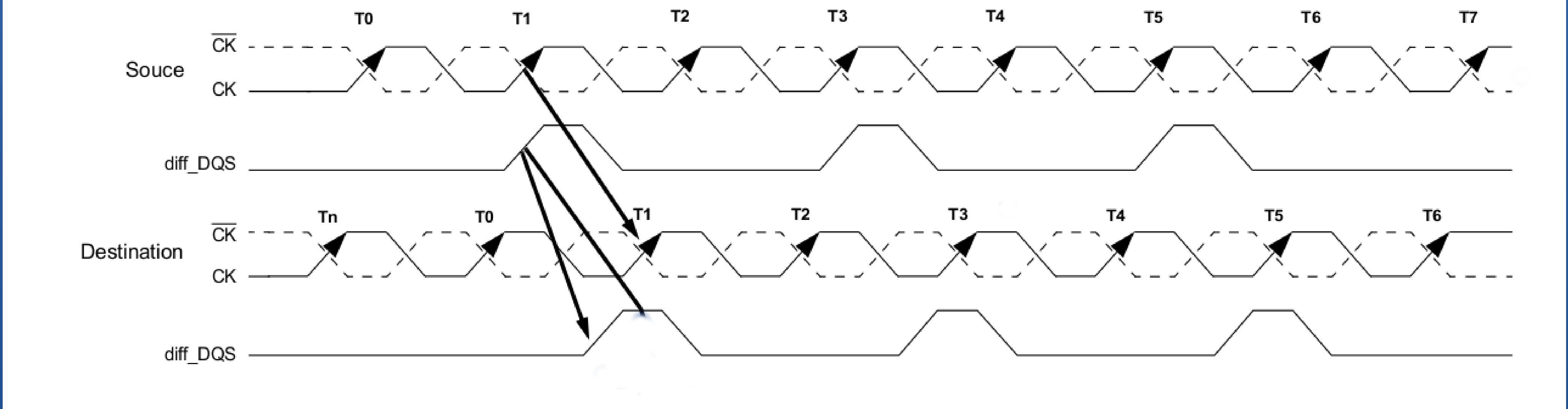
NOTE :
1. BL8, RL = 5, AL = 0, CL = 5.
2. DOUT n = data-out from column n.
3. NOP commands are shown for ease of illustration; other commands may be valid at these times.
4. BL8 setting activated by either MR0[A1:0 = 00] or MR0[A1:0 = 01] and A12 = 1 during READ command at T0.

Figure 22. READ Burst Operation RL = 5 (AL = 0, CL = 5, BL8)

Picture Source: Samsung DDR3 Device Operations Rev1.4_Nov11.book

Coreboot – Raminit

Write Leveling – Mechanism Part 1



Picture Source: Samsung DDR3 Device Operations Rev1.4_Nov11.book

Coreboot – Raminit

Write Leveling - Mechanism Part 2

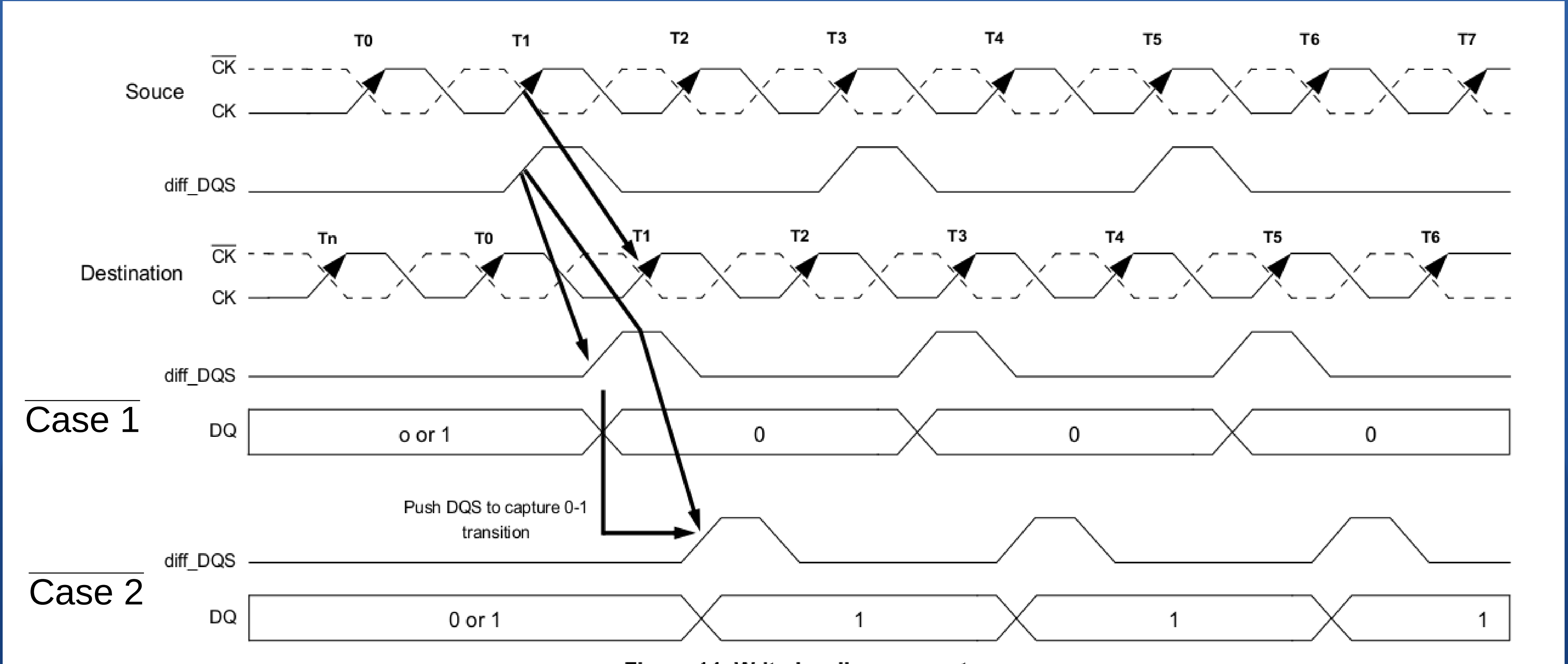


Figure 14. Write leveling concept

Picture Source: Samsung DDR3 Device Operations Rev1.4_Nov11.book

Coreboot – Raminit

Command / DQS training

Coreboot – Raminit

Next time...

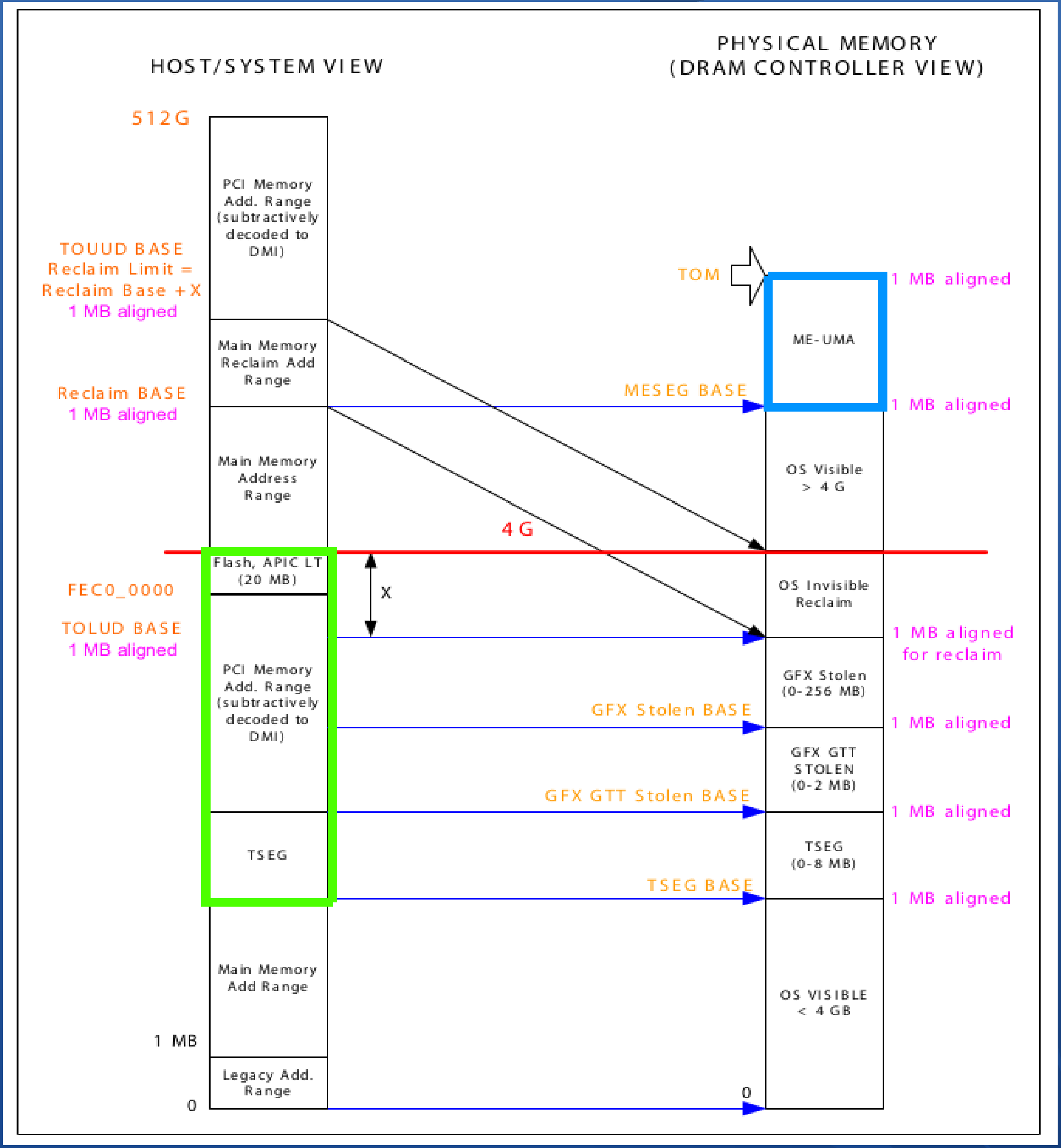
Coreboot – Raminit

Physical memory map

Coreboot – Raminit

Host physical memory map

- Lots of holes
- ME steals DRAM
- GFX steals DRAM
- SMM steals DRAM



Coreboot – Raminit

Security

Coreboot – Raminit

Cold boot attack:

1) Force reset and boot from USB to dump DRAM contents
2) Power off, “freeze” the memory and dump in second device

Coreboot – Raminit

Cold boot attack 1):

Solutions:

- Firmware password → Payload task
- Bootguard with custom keys → Requires Tianocore
- Clear all memory at boot → TODO: Add support in coreboot

Coreboot – Raminit

Raminit sequence includes:

DRAM Reset Gate (started with DDR3):

- Resets MRS registers and disables self refresh
- Data integrity not guaranteed any more
- Only takes microseconds …

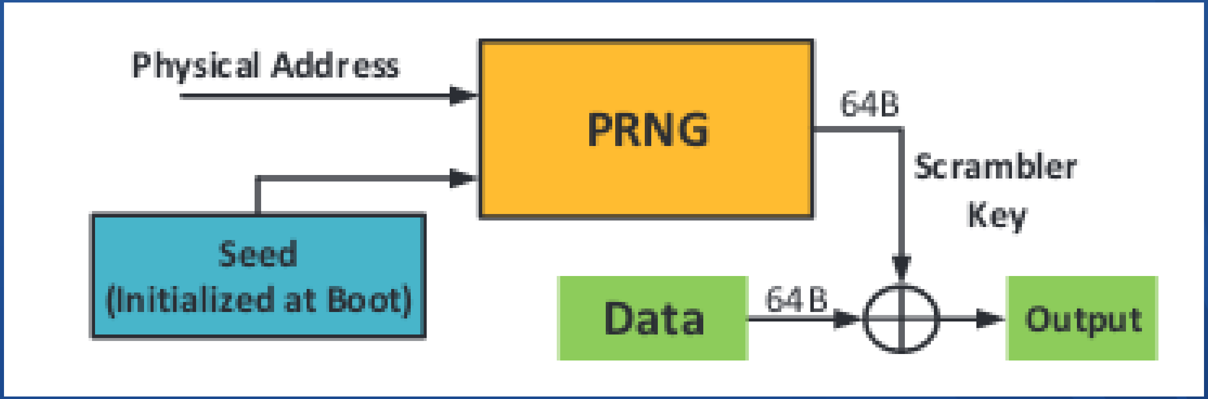
Memory scrambling:

- Decreases electrical current peaks
- 64 Byte blocks
- Seed initialized on boot

Coreboot – Raminit

Memory scrambling:

- 64 Byte block on SandyBridge
- Easy to find using known plain text attack
- Seed is constant in coreboot 4.6 → TODO: Use new seed on every cold boot
- 4096 x 64 Byte blocks on Skylake



Picture Source: “Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors”, University of Michigan, Ann Arbor

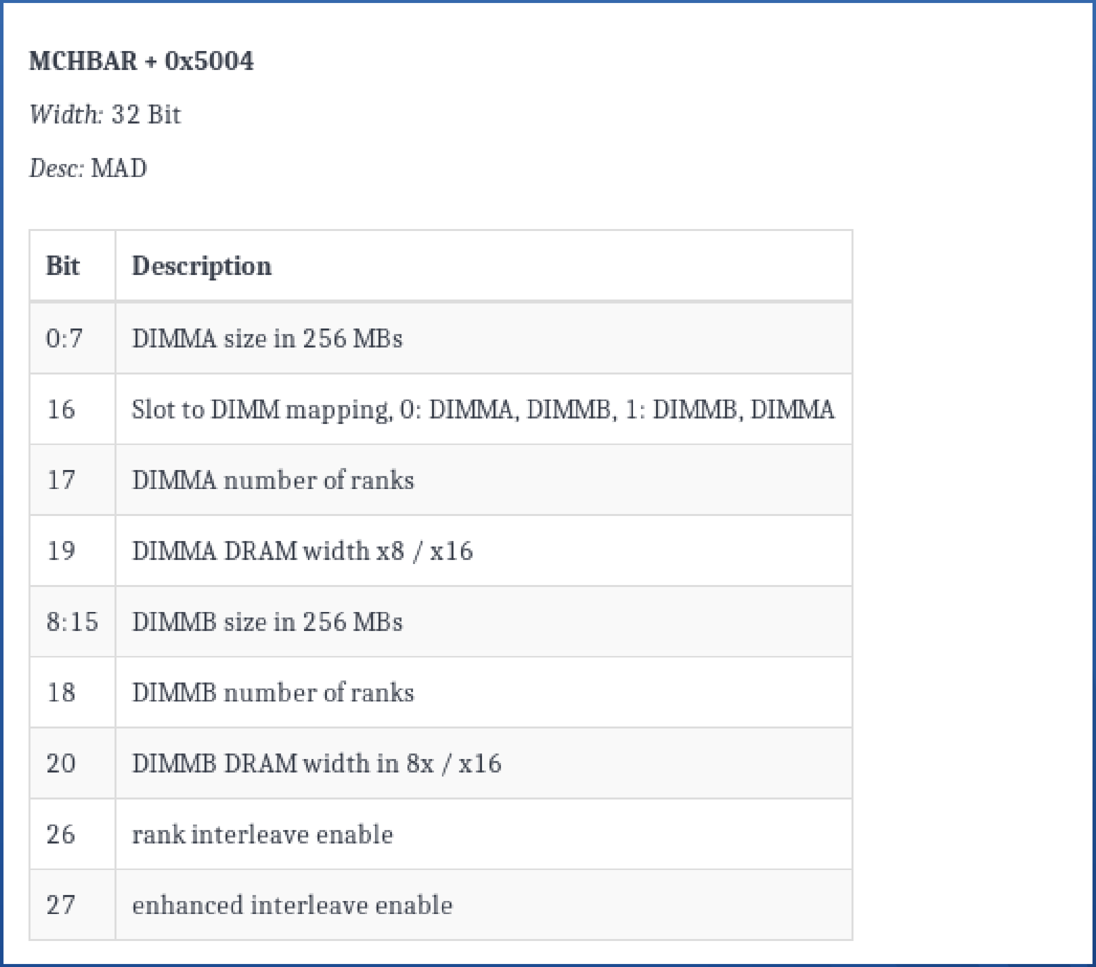
Coreboot – Raminit

(Inofficial) Documentation

Coreboot – Raminit

Documentation done so far:

- Read Training
- Memory Controller Registers



MCHBAR + 0x5004

Width: 32 Bit

Desc: MAD

Bit	Description
0:7	DIMMA size in 256 MBs
16	Slot to DIMM mapping, 0: DIMMA, DIMMB, 1: DIMMB, DIMMA
17	DIMMA number of ranks
19	DIMMA DRAM width x8 / x16
8:15	DIMMB size in 256 MBs
18	DIMMB number of ranks
20	DIMMB DRAM width in 8x / x16
26	rank interleave enable
27	enhanced interleave enable

Picture Source: By Patrick Rudolph

Coreboot – Raminit

Conclusion and outlook

Coreboot – Raminit

TODOs:

- Improve security
- Improve stability
- Improve memtest (using memtest86+ ?)
- ODT training ?
- Do documentation !
- Haswell raminit ?

Coreboot – Raminit

Questions ?