FTF FREESCALE TECHNOLOGY FORUM 2014

DRAM Controller Optimization for i.MX

FTF-SDS-F0170

Mark Middleton

APR.2014

freescale™

External Use

Overview

- This presentation is covering the tools used by the factory to optimize and debug DRAM interface on i.MX
  - This is not a deep dive in to various i.MX DRAM controller designs
  - This is not a training on various DRAM technologies
    - Please refer to the JEDEC specs
- These tools are available for use by our customers through the assigned Freescale FAE's.
- These tools are the precursors to the tests that will be available through the Processor Expert tool that will be made publically available.

Agenda

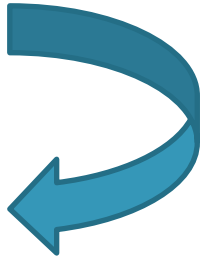
- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

Checklist Item	Details	owner	Findings & Status
The following items need to be completed serially			
Visual Inspection	Check major components to make sure nothing has been misplaced or rotated before applying power.		
Verify all SoC voltage rails	Confirm that the voltages match to what is required in the data sheet. Be sure to check voltages not only at the voltage source, but also as close to the SoC as possible, like on a bypass capacitor. This will reveal any IR drops on the board which will later cause issues. Ideally all of the SoC voltage rails should be checked, but noteworthy voltages include those that power the core, internal logic, boot devices, and DRAM (which is often overlooked).		
Verify power up sequence	Verify that Power On Reset (POR) is de-asserted (high) after all power rails have come up and are stable. Refer to the SoC data sheet for details on power up sequencing. This is an important if not overlooked process as many complex processors may be very sensitive to the proper power up sequencing.		
Measure/probe input clocks (32kHz, 24MHz, others)	Without a properly running clock, the SoC will not function properly. Look for jitter and noise.		
JTAG connectivity (RV-ICE, Lauterbach, Macraigor, etc)	This is one of the most fundamental and basic access points to the SoC to allow the debug and execution of low level code.		
Access internal RAM	Verify basic operation of the SoC in system. The on chip internal RAM starts at an address defined in the reference manual (normally in the Memory Map chapter) and includes the density of the on chip RAM. A basic test would simply be to perform a write-read-verify to the internal RAM via a JTAG debugger. No software initialization should be necessary to access internal RAM.		
Run basic DDR initialization and test memory	**Assuming the use of a JTAG debugger, run the DDR initialization and open a debugger memory window pointing to the DDR memory map starting address. Try writing a few words and verify if they can be read correctly. If not, re-check the DDR initialization sequence and if the DDR has been correctly soldered onto the board. It is also recommended to re-check the schematic to ensure the DDR memory has been connected to the SoC correctly. In some cases, a DRAM calibration routine may need to be executed, see next row.**		
Run DRAM Stress test (some SoC's include a DRAM calibration routine)	**A unit test that focuses on the robustness of the DRAM interface. Downloaded through JTAG debugger into internal RAM. Some SoC's DRAM stress test, like MX6Q, includes option to run DRAM calibration.**		
The following items may be worked on in parallel with other bring up tasks			
Verify CLKO outputs (measure and verify default clock frequencies for desired clock output options); this assumes that the board design supports probing of the CLKO pin.	This ensures that the corresponding clock is working and that the PLLs are working. This step does require some chip initialization, for example via the JTAG debugger, to properly set up the IOMUX to output CLKO and to set up the Clock Control Module to output the desired clock. Refer to the External Signals and Pin Multiplexing, CCM, and IOMUX sections of the SoC's reference manual for further details.		
Measure boot mode frequencies (set the boot mode switch for each boot mode and measure the following, depending on what is available in the system): - NAND (probe CE to verify boot, measure RE frequency) - SPI-NOR (probe slave select and measure clock freq) - MMC/SD (measure clock freq)	This verifies connectivity (at least for a few signals) between the SoC and boot device and that the boot mode signals are properly set.		
Run other unit tests	Once the DRAM interface has been verified as stable, the next step is to run other stand-alone unit tests to ensure the robustness of other peripherals and external components.		

Tools for DRAM Bring-up and Debug

DRAM Register Programming aid

Run basic DDR initialization and test memory	Assuming the use of a JTAG debugger, run the **DDR initialization** and open a debugger memory window pointing to the DDR memory map starting address. Try writing a few words and verify if they can be read correctly. If not, re-check the **DDR initialization** sequence and if the DDR has been correctly soldered onto the board. It is also recommended to re-check the schematic to ensure the DDR memory has been connected to the SoC correctly. In some cases, a DRAM calibration routine may need to be executed, see next row.
Run DRAM Stress test (some SoC's include a DRAM calibration routine)	**A unit test that focuses on the robustness of the DRAM interface**. Downloaded through JTAG debugger into internal RAM. Some SoC's **DRAM stress test**, like MX6Q, includes option to run DRAM calibration.



DRAM Stress Test

Agenda

- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

Agenda

- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

DRAM Register Programming Aid – Intro

- Tool to help create DRAM init scripts for specific memory types
  - Mainly used to help program JEDEC timing parameters
    - tRCD, tRC, tRFC, etc…
    - and for different DRAM parameters like rows, cols, and chip selects
  - Internal tool: Contact FAE for information
    - Customer Version will be found in Processor Expert
  - Excel spread sheet based, transparent, ease-of-use
  - “Automatically” creates RVD init script (.inc file)
    - To convert RVD to Lauterbach script format, Contact FAE.
    - How to modify *.inc script to *.ds for use with D5-Stream:
      - Change file name to *.ds
      - Substitute “mem set <reg_add> 32 <reg_val>”
      - For “set mem /32 <reg_add> = <reg_val>”

DRAM Register Programming Aid – Intro (Continued)

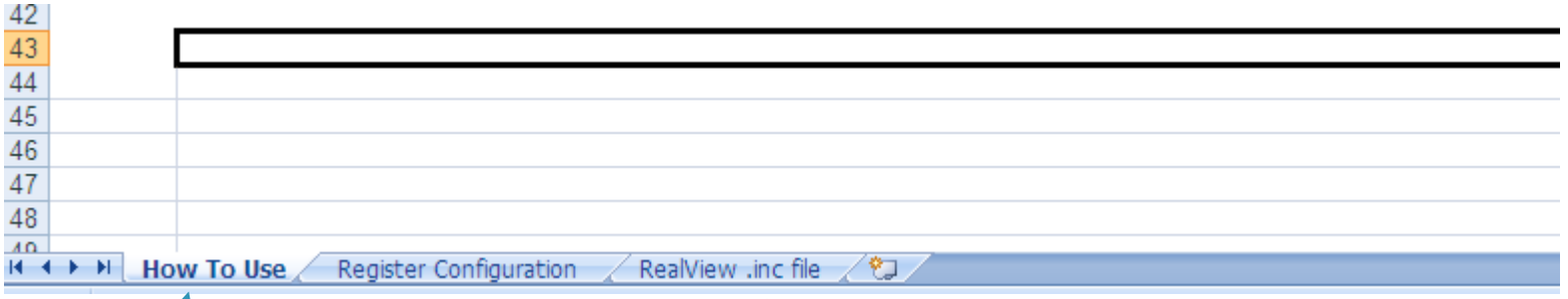
- Based on scripts provided by design/validation
- Anyone can use it, change it, fix it, etc…
- Each Programming Aid tool based on DRAM tech (DDR3, DDR2, LPDDR2, etc)
- What's been created to date:
  - MX6DQ: DDR3, LPDDR2; MX6DL: DDR3,LPDDR2; MX6SL: DDR3,LPDDR2
  - MX50: mDDR, LPDDR2, DDR2
  - MX28: mDDR, DDR2
  - What about other i.MX? No plans yet, need to resource this if enough interest

DRAM Register Programming Aid – Intro (Continued)

- Originated due to Denali controller on MX28 and MX50
  - Denali controller complex, many registers to program
  - Required use of Denali-specific tools available only to factory engineers (due to Denali license); burden on factory support
  - Even with Denali tools, the DRAM init scripts required further “tweaking” due to i.MX design integration
  - Register programming aid takes into account any “tweaking” and incorporates i.MX design specifics (based on scripts from design/validation)
  - Register programming aid offers more visibility on how the DRAM controller is being programmed
- Register programming aid tool concept carried over to i.MX FIL base controllers, starting with MX53 and MX6 series
- As of today, tools are available through assigned FAE’s
- In the near future, Customer versions available through PEx.

DRAM Register Programming Aid – Usage Overview

- Programming aid tool Excel Spreadsheet based
  - There are three tabs (worksheets)



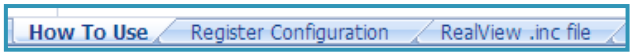
Start with “How To Use” tab for overview on how to use the tool

“Register Configuration” tab is where the user inputs various DRAM parameters

“RealView .inc file” tab contains the automatically generated DRAM initialization in the ARM RVD vernacular

DRAM Register Programming Aid – Usage Overview (Continued)

How to use the DRAM register programming aid outline



Step 1. Obtain the desired DRAM data sheet from the DRAM vendor

The following are to be completed in the Register Configuration Worksheet tab.

Note, each spread sheet is for a specific i.MX SoC and DRAM memory type.

Step 2. Update the Device Information table to include the DRAM information and system usage

Device Information	
Memory type:	mDDR
Manufacturer:	Micron
Memory part number:	MT46H64M16LF-5
Memory timing info:	5.0ns
Total DRAM Density per CS (Gb):	1
Number of ROW Addresses	14
Number of COLUMN Addresses	10
Number of BANKS	4
Number of Chip Selects used	1
Total DRAM density (Gb):	1
Bus Width	x16
Clock Cycle Freq (MHz):	200
Clock Cycle Time (ns)	5

Step 3. Go through the various shaded cells in the spread sheet to update with data from the DRAM sheet (take special note of the "Legend" table to ascertain the meaning of different shaded cells; in many cases, the cells may not need to be updated).

Instructions	Legend	
Shaded cells may require updating per the DRAM memory data sheet parameters. Certain registers should not need to be modified by the user. If a register is not provided then it is assumed this parameter is not to be changed per the provided initialization script or that the register is read-only. Certain registers are provided though they may be noted as recommended to not change.	On Register Configuration Tab, this color indicates the bitfields that would commonly require updating.	
	On Register Configuration Tab, this color indicates the bitfields that may be updated, but should typically not require it.	
	On Register Configuration Tab, this color indicates the bitfields that are updated automatically from setting provided in the "Device Information" table or other cells, and should not be changed manually.	Automatically Updated Setting
	On Register Configuration Tab, an unshaded cell means that the value should remain as is and should not be modified. In these cases, the settings are provided for completeness.	
	On other tabs, this color indicates the cells that are affected by changes on the Register Configuration tab. Note, this cell shading should not used in this worksheet Register Configuration tab, only in other tabs that are affected by cells in this tab.	

Pay attention to shaded cells – this is where you input data

Don't touch un-shaded cells

This is relevant to the "RealView .inc file" tab

The following refers to the RealView .inc file Worksheet tab. In this tab, the entire DRAM initialization can be obtained. This initialization can be used as a RealView include file (see below) or are reference for the bootloader DRAM initialization.

Step 4. Go to the RealView .inc file Worksheet tab and copy and paste this into a text document (make sure to rename the document with a ".inc" file ending); this is ready to use with the RealView development system.

Step 5. This .inc file can also be used as a reference for other debugger tools and bootloaders.

DRAM Register Programming Aid – Usage Overview (Continued)

- These columns are where the user inputs various DRAM parameters

How To Use | Register Configuration | RealView .inc file

dram parameter	ns	clk	binary setting within register	parameter description	Register name	Register address (HEX)	Register value (HEX)
TFAW (tFAW)	50	10	0A000000	Four bank activate window. This parameter could not be found in several mDDR data sheets. However, to maintain consistency with previous versions of the mDDR initialization, recommend to keep this value set to 50 as it has no affect on performance or functionality.	HW_DRAM_CTL39	0x800E009C	0x0A000000
TDLL	-	0	00000000	For mDDR, there is no DLL. *Keep this set this to 0.*			
TMRD (tMRD)	-	2	02000000	DRAM TMRD parameter in cycles. Defines the minimum number of cycles required between two mode register write commands. This is the time required to complete the write operation to the mode register.	HW_DRAM_CTL40	0x800E00A0	0x02009C40
TINIT (tINIT)	200000	40000	00009C40	DDR data sheet does not have a tINIT parameter however, this timing specifies the wait time after stable power up and stable clock before raising CKE high. JEDEC recommends waiting a minimum of 200us. *No need to change this parameter.*			
TPDEX (tXP)	-	2	00020000	DRAM TPDEX parameter in cycles. This is the power down exit time.	HW_DRAM_CTL41	0x800E00A4	0x0002030B
TRCD_INT (tRCD)	15	3	00000300	DRAM TRCD parameter in cycles. Defines the DRAM RAS to CAS delay, in cycles.			
TRC (tRC)	55	11	0000000B	DRAM TRC parameter in cycles. Defines the DRAM period between active commands for the same bank, in cycles. This is the sum of tRAS+tRP (for tRP, use the all banks parameter, tRP_AB for safer margin)			
TRAS_MAX (tRAS (max))	70000	14000	0036B000	DRAM TRAS_MAX parameter in cycles. Defines the DRAM maximum row active time, in cycles.	HW_DRAM_CTL42	0x800E00A8	0x0036B000
TRAS_MIN (tRAS (min))	40	8	00000008	DRAM TRAS_MIN parameter in cycles. Defines the DRAM minimum row activate time, in cycles.			
TRP (tRP)	15	3	03000000	DRAM TRP (single bank) parameter in cycles. Defines the DRAM pre-charge command time, in cycles.			
TRFC (tRFC)	110	22	00160000	DRAM TRFC parameter in cycles. Defines the DRAM refresh command time, in cycles.			
				DRAM TREF parameter in cycles. This is the auto refresh duty cycle (also called tREFI in some data sheets). This is the maximum time allowed between auto refresh commands to guarantee that a specified number of auto refresh commands are sent with a 64ms time period. The DDR data sheet should specify this parameter in "us" (micro seconds). In some cases, this value may have to be calculated if the DDR data sheet only specifies the number of auto refresh commands in a 64ms time period, in this case, simply take 64ms and divide it by the	HW_DRAM_CTL43	0x800E00AC	0x03160612

Example tRC:
1. User inputs '55' (in 'ns')
2. Tool calculates that 55ns=11clks*
3. Then configures binary setting for register
4. Tool takes all binary settings to create final register value

* Clock: 200MHz, clock period: 5ns

Example based on MX28

- Register values are calculated automatically from user inputs
- These values are also automatically updated to the "RealView .inc file" tab

ACTIVE-to-PRECHARGE command	^{t}RAS	40	70,000	42	70,000	42	70,000	45	70,000	ns	22
ACTIVE to ACTIVE/ACTIVE to AUTO REFRESH command period	^{t}RC	55	–	58.2	–	60	–	67.5	–	ns	
Active to read or write delay	^{t}RCD	15	–	16.2	–	18	–	22.5	–	ns	

Example Micron mDDR data sheet spec

DRAM Register Programming Aid – Usage Overview (Continued)

- Changing device information, automatic update to register fields

How To Use | Register Configuration | RealView .inc file

1. Type in device parameters here

Device Information	
Memory type:	DDR3
Manufacturer:	Micron
Memory part number:	MT41J128M16HA-15E
Density of each DDR3 device (Gb):	2
Number of DRAM devices per chip select	4
Density per chip select (Gb)[1]:	8
Number of Chip Selects used[2]	2
Total DRAM density (Gb)	16
Number of ROW Addresses[2]	14
Number of COLUMN Addresses[2]	10
Number of BANKS[2]	8
Bus Width (input 16, 32, or 64 bits)[2]	64
Clock Cycle Freq (MHz)[3]	533
Clock Cycle Time (ns)	1.876

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0xC31A0000
SDE_1	-	1	40000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW	-	3	03000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	1	00100000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	2	00020000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

2. Gets updated here

3. Final register setting here

1. ROW, COL, Bus width are now changed from above

Device Information	
Memory type:	DDR3
Manufacturer:	someone
Memory part number:	something
Density of each DDR3 device (Gb):	2
Number of DRAM devices per chip select	4
Density per chip select (Gb)[1]:	8
Number of Chip Selects used[2]	2
Total DRAM density (Gb)	16
Number of ROW Addresses[2]	15
Number of COLUMN Addresses[2]	11
Number of BANKS[2]	8
Bus Width (input 16, 32, or 64 bits)[2]	32
Clock Cycle Freq (MHz)[3]	533
Clock Cycle Time (ns)	1.876

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0xC4290000
SDE_1	-	1	40000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW	-	4	04000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	2	00200000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	1	00010000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

2. Gets updated here

BTW, these cells are not to be touched either, they are automatically updated given previously entered parameters

DRAM Register Programming Aid – Usage Overview (Continued)

- Another detailed look…
- Let's say on MX6DQ, you had only one chip select populated (CS0)

How To Use | Register Configuration | RealView .inc file

Before, the MX6DQ register was as follows for the validation board which has DDR3 on both chip selects

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0xC31A0000
SDE_1	-	1	40000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW		3	03000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	1	00100000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	2	00020000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

Now, after the change to enable CS0 only, here's what the register looks like

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0x831A0000
SDE_1	-	0	00000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW	-	3	03000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	1	00100000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	2	00020000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

Clearing this means we are not enabling chip select 1

Register setting gets automatically updated

Don't touch. When BL cleared, burst length is 4 which is setting for LPDDR2

DRAM Register Programming Aid – Usage Overview (Continued)

//==========		
// DDR Controller Registers		
//==========		
// Manufacturer:	Micron	
// Device Part Number:	MT46H64M16LF-5	
// Clock Freq.:	200MHz	
// Density per CS (Gb):	1	
// Chip Selects:	1	
// Number of Banks:	4	
// Row address:	14	
// Column address:	10	
//==========		
setmem /32	0x800E0000=	0x00000000
setmem /32	0x800E0040=	0x00000000
setmem /32	0x800E0054=	0x00000000
setmem /32	0x800E0058=	0x00000000
setmem /32	0x800E005C=	0x00000000
setmem /32	0x800E0060=	0x00000000
setmem /32	0x800E0064=	0x00000000
setmem /32	0x800E0068=	0x00010101
setmem /32	0x800E006C=	0x01010101
setmem /32	0x800E0070=	0x000f0f01
setmem /32	0x800E0074=	0x0102010A
setmem /32	0x800E007C=	0x00000101
setmem /32	0x800E0080=	0x00000100
setmem /32	0x800E0084=	0x00000100
setmem /32	0x800E0088=	0x01000000
setmem /32	0x800E008C=	0x00000002
setmem /32	0x800E0090=	0x01010000
setmem /32	0x800E0094=	0x08060301
setmem /32	0x800E0098=	0x06000001
setmem /32	0x800E009C=	0x0A000000
setmem /32	0x800E00A0=	0x02009C40
setmem /32	0x800E00A4=	0x0002030B
setmem /32	0x800E00A8=	0x0036B068

How To Use | Register Configuration | RealView .inc file

- DRAM initialization automatically generated per template
- Values in yellow are automatically updated from the "Register Configuration" tab (previous slide)
- No user input required in this tab

A few slides back, we saw an example of programming tRC and how the tool created the final value for this register address 0x800E00A4. The value is automatically transferred to this location.

TPDEX (tXP)	-	2	00020000	DRAM TPDEX parameter in cycles. This is the power down exit time.	HW_DRAM_CTL41	0x800E00A4	0x0002030B
TRCD_INT (tRCD)	15	3	00000300	DRAM TRCD parameter in cycles. Defines the DRAM RAS to CAS delay, in cycles.			
TRC (tRC)	55	11	0000000B	DRAM TRC parameter in cycles. Defines the DRAM period between active commands for the same bank, in cycles. This is the sum of tRAS+tRP (for tRP, use the all banks parameter, tRP_AB for safer margin)			

How To Use | Register Configuration | RealView .inc file

DRAM Register Programming Aid – Usage Overview (Continued)

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0xC31A0000
SDE_1	-	1	40000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW	-	3	03000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	1	00100000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	2	00020000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

- This example from the MX6DQ DDR3 register programming aid illustrates that the register value for MDCTL in the “RealView .inc file” tab is taken directly from the MDCTL Register value cell in the “Register Configuration” tab

```
124 setmem /32  0x021b0004 =  0x00020036  // MMDC0_MDPDC
125 setmem /32  0x021b0008 =  0x09444040  // MMDC0_MDOTC
126 setmem /32  0x021b000c =  0x555A7975  // MMDC0_MDCFG0
127 setmem /32  0x021b0010 =  0xFF538F64  // MMDC0_MDCFG1
128 setmem /32  0x021b0014 =  0x01FF00DB  // MMDC0_MDCFG2
129 setmem /32  0x021b0018 =  0x00081740  // MMDC0_MDMISC
130
131 setmem /32  0x021b001c =  0x00008000  // MMDC0_MDSCR, set th
132 setmem /32  0x021b002c =  0x000026d2  // MMDC0_MDRWD; recor
133 setmem /32  0x021b0030 =  0x005A0E21  // MMDC0_MDOR
134 setmem /32  0x021b0040 =  0x00000027  // CS0_END
135 setmem /32  0x021b0000 =  0xC31A0000  // MMDC0_MDCTL
136 // Mode register writes
137 setmem /32  0x021b001c =  0x04088032  // MMDC0_MDSCR, MR2 v
138 setmem /32  0x021b001c =  0x00008033  // MMDC0_MDSCR, MR3 v
139 setmem /32  0x021b001c =  0x00048031  // MMDC0_MDSCR, MR1 v
140 setmem /32  0x021b001c =  0x09408030  // MMDC0_MDSCR, MR0 v
141 setmem /32  0x021b001c =  0x04008040  // MMDC0_MDSCR, ZQ ca
142
143 setmem /32  0x021b001c =  0x0408803A  // MMDC0_MDSCR, MR2 v
144 setmem /32  0x021b001c =  0x0000803B  // MMDC0_MDSCR, MR3 v
145 setmem /32  0x021b001c =  0x00048039  // MMDC0_MDSCR, MR1 v
```

DRAM Register Programming Aid Walkthrough.

- MX6DQ DDR3 Example
  (based on DDR3 validation board)

DRAM Register Programming Aid - Walkthrough

Instructions

Shaded cells may require updating per the DRAM memory data sheet parameters. Certain registers should not need to be modified by the user. If a register is not provided then it is assumed this parameter is not to be changed per the provided initialization script.Certain registers are provided though they may be noted as recommended to not change.

Legend

On Register Configuration Tab, this color indicates the bitfields that would commonly require updating.	
On Register Configuration Tab, this color indicates the bitfields that may be updated, but should typically not require it.	
On Register Configuration Tab, this color indicates the bitfields that are updated automatically from setting provided in the "Device Information" table or other cells, and should not be changed manually	Automatically Updated Setting
On Register Configuration Tab, an unshaded cell means that the value should remain as is and should not be modified. In these cases, the settings are provided for completeness.	
On other tabs, this color indicates the cells that are affected by changes on the Register Configuration tab.	

For Sabre boards with CS1 not populated, set this to 1

Device Information	
Memory type:	DDR3
Manufacturer:	Micron
Memory part number:	MT41J128M16HA-15E
Density of each DDR3 device (Gb):	2
Number of DRAM devices per chip select	4
Density per chip select (Gb)[1]:	8
Number of Chip Selects used[2]	2
Total DRAM density (Gb)	16
Number of ROW Addresses[2]	14
Number of COLUMN Addresses[2]	10
Number of BANKS[2]	8
Bus Width (input 16, 32, or 64 bits)[2]	64
Clock Cycle Freq (MHz)[3]	533
Clock Cycle Time (ns)	1.876

User inputs clock speed "Clock Cycle (ns)" will change accordingly, value used throughout tool

1. Important: it is necessary to populate this field with the density in Gbits as it is used later in the CS0_END calculation. This field is cacluated from the previous two fields. The user can also simply type in the total density as well in this field.
2. Important, these fields need to be filled out correctly as these values are used later in this tool for register settings.
3. Even though i.MX6Q runs at 528MHz, set timings according to 533MHz to allow for a little more margin

Revision History

0.9 - Fixed issue with SDCKE0/1 IOMUX programming (DSE set in CTLDS)
0.8 - Made calibration values update-able in the Register Configuration tab to easily update the values in the RealView .inc file tab
0.7 - added DDR_SEL field to DRAM_RESET pad, changed MR1 write (needed to align with DDR_SEL field set to 11).
0.6 - added power savings features to align with validation scripts
0.5 - added MPODTCTRL register programming option
0.4 - updated tMOD field: In DDR3 mode this field shoud be set to max (tMRD,tMOD).
0.3 - fixed DDR_4_BANK description
0.2 - grammactical corrections
0.1 - initial

Pull down menu for DSE

From the ref manual

000	DISABLED — Output driver disabled.
001	240OHM — 240 Ohm
010	120OHM — 120 Ohm
011	80OHM — 80 Ohm
100	60OHM — 60 Ohm
101	48OHM — 48 Ohm
110	40OHM — 40 Ohm
111	34OHM — 34 Ohm

DRAM Pad Name	Field (i.e. DSE)	Binary Setting	bit setting within register	Notes			Pad Register value (HEX)
SDQS[7:0]	DSE	111	00000038	000: output driver disabled 001: Weakest ... 111: Strongest	IOMUXC_SW_PAD_CTL_PAD_DRAM_SDQS[7:0]	0x020e05a8 0x020e05b0 0x020e0524 0x020e051c 0x020e0518 0x020e050c 0x020e05b8 0x020e05c0	0x00000038
DQM[7:0]	DDR_INPUT	0	00000000	0: CMOS mode (recommended) 1: Differential input mode	IOMUXC_SW_PAD_CTL_PAD_DRAM_DQM[7:0]	0x020e05ac 0x020e05b4 0x020e0528 0x020e0520 0x020e0514 0x020e0510 0x020e05bc 0x020e05c4	0x00000030
	DSE	000 001 ... 111	00000030	000: output driver disabled 001: Weakest ... 111: Strongest			

DRAM Register Programming Aid - Walkthrough

SDCLK_0, SDCLK_1	DDR_INPUT	0	00000000	0: CMOS mode (recommended) 1: Differential input mode	IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_0 IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCLK_1	0x020e0588 0x020e0594	0x00000030
	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest			
DRAM_RESET	DDR_SEL	11	000C0000	DDR Select Field Select one out of next values for pad: DRAM_RESET. 00 Reserved 01 Reserved 10 LPDDR2 mode (240Ohm driver unit calibration, 240, 120, 80, 60, 48, 40, 32 Ohm drive strengths at 1.2V) 11 DDR3 mode (240Ohm driver unit calibration, 240, 120, 80, 60, 48, 40, 32 Ohm drive strengths at 1.5V	IOMUXC_SW_PAD_CTL_PAD_DRAM_RESET	0x020e057c	0x000C0030
	DDR_INPUT	0	00000000	0: CMOS mode (recommended) 1: Differential input mode			
	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest			
SDCKE0, SDCKE1 (DSE (drive strength) configured in CTLDS)	DDR_INPUT	0	00000000	0: CMOS mode (recommended) 1: Differential input mode	IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE0 IOMUXC_SW_PAD_CTL_PAD_DRAM_SDCKE1	0x020e0590 0x020e0598	0x00003000
	HYS	0	00000000	0: Hysteresis disable (recommended) 1: Hysteresis enable			
	PUS	0	00000000	Pull Up / Down Config. Field 00: 100KOhm Pull Down (recommended) 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up			
	PUE	1	00002000	Pull / Keep Select Field 0: Keeper			
	PKE	1	00001000	Pull / Keep Enable Field 0: Pull/Keeper Disabled 1: Pull/Keeper Enabled (recommended)			
	ODT	000	00000000	On Die Termination Field 000: off (recommended) 001 120 Ohm ODT ... 110 20 Ohm ODT 111 RESERVED			
SDODT0, SDODT1	DDR_INPUT	0	00000000	0: CMOS mode (recommended) 1: Differential input mode	IOMUXC_SW_PAD_CTL_PAD_DRAM_SDODT0 IOMUXC_SW_PAD_CTL_PAD_DRAM_SDODT1	0x020e059c 0x020e05a0	0x00003030
	HYS	0	00000000	0: Hysteresis disable (recommended) 1: Hysteresis enable			
	PUS	0	00000000	Pull Up / Down Config. Field 00: 100KOhm Pull Down (recommended) 01: 47KOhm Pull Up 10: 100KOhm Pull Up 11: 22KOhm Pull Up			
	PUE	1	00002000	Pull / Keep Select Field 0: Keeper			
	PKE	1	00001000	Pull / Keep Enable Field 0: Pull/Keeper Disabled 1: Pull/Keeper Enabled (recommended)			
	ODT	000	00000000	On Die Termination Field 000: off (recommended) 001 120 Ohm ODT ... 110 20 Ohm ODT 111 RESERVED			
	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest			

Continuation of “IOMUX part” of DRAM Register Programming aid

Note, for Sabre boards with only one CS populated, we found we can reduce all DRAM IO DSE to 101

DRAM Register Programming Aid - Walkthrough

Many IO's are grouped together

Continuation of "IOMUX part" of DRAM Register Programming aid

B0DS, B1DS, B2DS, B3DS, B4DS, B5DS, B6DS, B7DS (DRAM data byte groups)	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest	IOMUXC_SW_PAD_CTL_PAD_DRAM_B[7:0]DS	0x020e0784 0x020e0788 0x020e0794 0x020e079c 0x020e07a0 0x020e07a4 0x020e07a8 0x020e0748	0x00000030
ADDDS (DRAM Address pads A[15:0] and SDBA[2:0])	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest	IOMUXC_SW_PAD_CTL_PAD_DRAM_ADDDS	0x020e074c	0x00000030
CTLDS (DRAM Control pads CS0, CS1, SDBA2, SDCKE0, SDCKE1, SDWE)	DSE	110	00000030	000: output driver disabled 001: Weakest ... 111: Strongest	IOMUXC_SW_PAD_CTL_GRP_CTLDS	0x020e078c	0x00000030
DDRMODE_CTL	DDR_INPUT	1	00020000	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE_CTL (Pads: DRAM_SDQS0 DRAM_SDQS1 DRAM_SDQS2 DRAM_SDQS3 DRAM_SDQS4 DRAM_SDQS5 DRAM_SDQS6 DRAM_SDQS7). 0: CMOS input type 1: Differential input mode (recommended)	IOMUXC_SW_PAD_CTL_GRP_DDRMODE_CTL	0x020e0750	0x00020000
DDRMODE	DDR_INPUT	1	00020000	DDR / CMOS Input Mode Field Select one out of next values for group: DDRMODE (Pads: DRAM_D[63:0]). 0: CMOS input type 1: Differential input mode (recommended)	IOMUXC_SW_PAD_CTL_GRP_DDRMODE	0x020e0774	0x00020000
DDR_TYPE	DDR_SEL	11	000C0000	DDR Select Field Select one out of next values for group: DDR_TYPE for all DDR address, data, and control pads 00: reserved 01: reserved 10: LPDDR2 mode (240Ohm driver unit calibration, 240, 120, 80, 60, 48, 40, 34 Ohm drive strengths) at 1.2V power 11: DDR3 mode (240Ohm driver unit calibration, 240, 120, 80, 60, 48, 40, 34 Ohm drive strengths) at 1.5V power (required for DDR3)	IOMUXC_SW_PAD_CTL_GRP_DDR_TYPE	0x020e0798	0x000C0000
DDRPKE	PKE	0	00000000	Pull / Keep Enable Field for DDR pads 0: Pull/Keeper Disabled (recommended) 1: Pull/Keeper Enabled	IOMUXC_SW_PAD_CTL_GRP_DDRPKE	0x020e0758	0x00000000

DDR3 Register Programming Aid – Walkthrough

For Sabre boards with CS1 not populated, we need to clear this bit (SDE_1)

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
SDE_0	-	1	80000000	SDE_0: Enable Chip Select 0, set to 0 (disable) or 1 (enable)	MDCTL	0x021B0000	0xC31A0000
SDE_1	-	1	40000000	SDE_1: Enable Chip Select 1, set to 0 (disable) or 1 (enable)			
ROW	-	3	03000000	ROW: number of ROW addresses. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
COL	-	1	00100000	COL number of Column addresses. NOTE: : this value is taken from the Device Information table above. Modify this value only in the table above.			
BL	-	1	00080000	BL: Burst length. For DDR3, set to 1 for burst length 8.			
DSIZ	-	2	00020000	DSIZ: Data bus size. Note: this value is taken from the Device Information table above. Modify this value only in the table above.			

These are automatically updated from previous parameter settings

MMDC timing parameter (DDR device timing parameter)	value from DDR data sheet (ns)	Clock Cycle or Binary Setting	bit setting within register	Notes	Register name	Register address (HEX)	Register value (HEX)
tCKE	5.625	3	00020000	tCKE - CKE minimum pulse width. Obtain this value from DDR3 data sheet.	MDPDC	0x021B0004	0x00020036
tCKSRX	10	6	00000030	tCKSRX - This field determines the amount of clock cycles for selfrefresh exit. Obtain this value from DDR3 data sheet.			
tCKSRE	10	6	00000006	tCKSRE - This field determines the amount of clock cycles for self-refresh entry. Obtain this value from DDR3 data sheet.			
PWDT_1	-	5	00005000	Power Down Timer - Chip Select 1. Freescale validation determined a value of 5 was the most optimal. For systems with only one chip select with devices on CS0, thess bits don't apply, but keeping them set will allow backwards compatibility with systems with two chip selects.			0x00025576
PWDT_0	-	5	00000500	Power Down Timer - Chip Select 0. Freescale validation determined a value of 5 was the most optimal.			
BOTH_CS_PD	-	1	00000040	Parallel power down entry to both chip selects. Leave this set so that both chip selects will enter power down together. For systems with only one chip select, this bit doesn't apply, but keeping it set will allow backwards compatibility with systems with two chip selects.			

There are two separate settings, first for initialization, the second enables power savings, it gets placed at end of initialization

JEDEC standard timing parameters, obtain value from DDR3 data sheet

For Sabre boards with CS1 not populated, we can keep these parameters untouched as they have no adverse affect and will minimize differences between init scripts

DRAM Register Programming Aid - Walkthrough

tAOFPD	2	2	08000000	tAOFPD - This field determines the time between termination cuircuit starts to turn off the ODT resistance till termination has reached high impedance. Obtain this value from DDR3 data sheet.	MDOTC	0x021B0008	0x09444040
tAONPD	2	2	01000000	tAONPD - This field determines the time between termination cuircuit gets out of high impedance and begins to turn on till ODT resistance are fully on. Obtain this value from DDR3 data sheet.			
tANPD	-	5	00400000	tANPD - Asynchronous ODT to power down entry delay. In DDR3 should be set to tCWL-1. Obtain this value from DDR3 data sheet. This is also calcuated from MDCGFG1[tCWL].			
tAXPD	-	5	00040000	tAXPD - Asynchronous ODT to power down exit delay. In DDR3 should be set to tCWL-1. Obtain this value from DDR3 data sheet. This is also calcuated from MDCGFG1[tCWL].			
tODTLon	-	4	00004000	tODTLon - This field determines the delay between ODT signal and the associated RTT, where according to JEDEC standard it equals WL(write latency) - 2. Therefore, the value that is configured to tODTLon field should correspond the value that is configured to MDCGFG1[tCWL]. Obtain this value from DDR3 data sheet. This is also calcuated from MDCGFG1[tCWL].			
tODT_idle_off	-	4	00000040	memory ODT off. Obtain this value from DDR3 data sheet. Usually just tCWL-2.			
tRFC	160	86	55000000	tRFC - Refresh command to Active or Refresh command time. Obtain this value from DDR3 data sheet.	MDCFG0	0x021B000C	0x555A7975
tXS	170	91	005A0000	tXS - Exit self refresh to non READ command. Obtain this value from DDR3 data sheet.			
tXP	6	4	00006000	tXP - Exit power down with DLL-on to any valid command. Obtain this value from DDR3 data sheet.			
tXPDLL	24	13	00001800	tXDPLL - Exit precharge power down with DLL frozen to commands requiring DLL. Obtain this value from DDR3 data sheet.			
tFAW	45	24	00000170	tFAW - Four Active Window (all banks). Obtain this value from DDR3 data sheet.			
tCL	13.5	8	00000005	tCL - CAS Read Latency. Obtain this value from DDR3 data sheet.			

These are automatically updated from other parameter settings

JEDEC standard timing parameters, obtain value from DDR3 data sheet

DRAM Register Programming Aid – Walkthrough

JEDEC standard timing parameters, obtain value from DDR3 data sheet

tRCD	13.5	8					
tRP	13.5	8	1C000000	tRP - Precharge command period (same bank). Obtain this value from DDR3 data sheet.			
tRC	49.5	27	03400000	tRC - Active to Active or Refresh command period (same bank). Obtain this value from DDR3 data sheet.			
tRAS	36	20	00130000	tRAS - Active to Precharge command period (same bank). Obtain this value from DDR3 data sheet.			
tRPA	-	1	00008000	tRPA - Precharge-all command period. Obtain this value from DDR3 data sheet.	MDCFG1	0x021B0010	0xFF538F64
tWR	15	8	00000E00	tWR - WRITE recovery time (same bank). Obtain this value from DDR3 data sheet.			
tMRD	-	12	00000160	tMRD - Mode Register Set command cycle (all banks). Obtain this value from DDR3 data sheet. In DDR3 mode this field shoud be set to max (tMRD,tMOD).			
tCWL	-	6	00000004	tCWL - CAS Write Latency. Obtain this value from DDR3 data sheet			
tDLLK	-	512	01FF0000	tDLLK - DLL locking time. 512			
tRTP	7.5	4	000000C0	tRTP - Internal READ comman bank). Obtain this value from D			
tWTR	7.5	4	00000018	tWTR - Internal WRITE to REA value from DDR3 data sheet.			
tRRD	6	4	00000003	tRRD - Active to Active comma from DDR3 data sheet.			
tXPR	170	91	005A0000	tXPR - Obtain this value from D			
SDE_to_RST	-	13	00000E00	SDE_to_RST			
RST_to_CKE	-	32	00000021	RST_to_CKE			

MMDC pin	Chip select 0 pin	Chip select 1 pin
A3	A3	A4
A4	A4	A3
A5	A5	A6
A6	A6	A5
A7	A7	A8
A8	A8	A7
B0	B0	B1
B1	B1	B0

Needs to align with board layout
Sabre boards don't populate CS1, so clear this bit

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register				
CALIB_PER_CS	-	0	00000000	CALIB_PER_CS: determines which chip select the associated calibration is targetted to. Set to 0 for CS0 or 1 for CS1. Note if both chip selects are populated, recommend to set to 0.			
ADDR_MIRROR	-	1	00080000	ADDR_MIRROR: for DDR3, address bits in the following pairs [A3,A4], [A5,A6], [A7,A8], [B0,B1] are swapped from the MMDC to CS1 to aid in routing of the DDR device on CS1. As DDR devices on CS1 are placed on the opposite side of the board from CS0, swapping these signals aids in the board routing. If the board enables this feature and routes these signals accordingly, then set this bit. Otherwise, this bit should be cleared.			
LHD	-	0	00000000	LHD: Latency hiding disable. Recommend to clear this bit to enable latency hiding and improve performance.			
WALAT	-	0	00000000	WALAT: Write Additional latency. Recommend to clear these bits. Proper board design should ensure that the DDR3 devices are placed close enough to the MMDC to ensure the shew between CLK and DQS is less than 1 cycle.	MDMISC	0x021B0018	0x00081740
BI_ON	-	1	00001000	BI_ON: Bank Interleaving On. Recommend to set this bit to enable bank interleaving; improves performance.			
LPDDR2_S2	-	0	00000000	Set to 0; not applicable to DDR3.			
MIF3_MODE	-	3	00000600	MIF3_MODE: Command prediction working mode; set to 0x3 for optimal performance.			
RALAT	-	5	00000140	RALAT: Read Additional Latency. Set to 0x5 for optimal performance.			
DDR_4_BANK	-	0	00000000	DDR_4_BANK: set to 0 for 8 banks, 1 for 4 banks. NOTE: this value is taken from the Device Information table above. Modify this value only in the table above.			
DDR_TYPE	-	0	00000000	DDR3 mode - set to 0. Do not change.			
LPDDR2_2CH	-	0	00000000	Set to 0; not applicable to DDR3.			
RST	-	0	00000000	Set to 0 for normal operation.			

If any WL calibration parameter >0x20, should set WALAT = 1.

Because this is a dedicated DDR3 programming aid, these values are pre-set and not to be changed

DRAM Register Programming Aid – Walkthrough

MMDC Parameter	Calculated End Address (starting at offset 0x10000000)	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
CS0_END	0x50000000	39	00000027	Note: DO NOT change the values directly in these cells, instead, program the Density of each DDR device and number of devices in the cells at the top of the page. End Address is calculated from cells above: Density of each DDR device multiplies by the number of devices per chip select, then add offset of 256MB. Note that the total DDR density on CS0 is offset by 0x1000000, which is the starting address of the CS0 memory region. Hence the CS0_END is the sum of the DDR density on CS0 with offset 0x10000000.	MDASP	0x021B0040	0x00000027

MMDC Control Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
ODT3_INT_RES	-	1	00020000	On chip ODT byte3 resistor - This field determines the Rtt_Nom of the on chip ODT byte3 resistor during read accesses. For x64 configuration, this applies to byte 7 with register address at 0x021B4818. 000 Rtt_Nom Disabled. 001 Rtt_Nom 120 Ohm/75 Ohm 010 Rtt_Nom 60 Ohm/150 Ohm (default setting for FSL boards) 011 Rtt_Nom 40 Ohm/50 Ohm 100 Rtt_Nom 30 Ohm/37.5 Ohm 101 Rtt_Nom 24 Ohm/30 Ohm 110 Rtt_Nom 20 Ohm/25 Ohm 111 Rtt_Nom 17 Ohm/21 Ohm	MPODTCTRL	0x021B0818 0x021B4818	0x00011117
ODT2_INT_RES	-	1	00002000	On chip ODT byte2 resistor - This field determines the Rtt_Nom of the on chip ODT byte2 resistor during read accesses. For x64 configuration, this applies to byte 6 with register address at 0x021B4818.			
ODT1_INT_RES	-	1	00000200	On chip ODT byte1 resistor - This field determines the Rtt_Nom of the on chip ODT byte1 resistor during read accesses. For x64 configuration, this applies to byte 5 with register address at 0x021B4818.			
ODT0_INT_RES	-	1	00000020	On chip ODT byte0 resistor - This field determines the Rtt_Nom of the on chip ODT byte0 resistor during read accesses. For x64 configuration, this applies to byte 4 with register address at 0x021B4818.			
ODT_RD_ACT_EN	-	0	00000000	Active read CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during read accesses. 0 Active CS ODT pin is disabled during read access. 1 Active CS ODT pin is enabled during read access. This is generally not set for Freescale boards			
ODT_RD_PAS_EN	-	1	00000004	Inactive read CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during read accesses. 0 Inactive CS ODT pin is disabled during read accesses to other CS. 1 Inactive CS ODT pin is enabled during read accesses to other CS. For Freescale boards with devices on both chip selects, this bit is generally set as the board layout considers termination applied for the non-active chip select device. For boards with a device on only one chip select, this bit can be cleared, however, leaving it set should not cause any issues.			
ODT_WR_ACT_EN	-	1	00000002	Active write CS ODT enable. The bit determines if ODT pin of the active CS will be asserted during write accesses. 0 Active CS ODT pin is disabled during write access. 1 Active CS ODT pin is enabled during write access. For Freescale boards with devices on both chip selects, this bit is generally set. In some cases, the board may be designed to account for the termination of only the other (passive) device during writes, hence this bit can be cleared. However, even in such a case, leaving it set does not cause any issues. For boards with only one chip select populated, it is recommended to set this bit. Hence, by default, this bit remains set.			
ODT_WR_PAS_EN	-	1	00000001	Inactive write CS ODT enable. The bit determines if ODT pin of the inactive CS will be asserted during write accesses. 0 Inactive CS ODT pin is disabled during write accesses to other CS. 1 Inactive CS ODT pin is enabled during write accesses to other CS. For Freescale boards with devices on both chip selects, this bit is generally set as the board layout considers termination applied for the non-active chip select device. For boards with a device on only one chip select, this bit can be cleared, however, leaving it set should not cause any issues.			

This gets calculated from the “density parameter per chip select “(cell C21).
You can click on the cells to see the following formulas:

```
="0x"&DEC2HEX(((C21*1024*1024*1024)/8+ 256*1024*1024), 8)
=((((C19*C20*1024*1024*1024)/8+ 256*1024*1024)/(32*1024*1024))-1)
```

More details of this in the next slide

DRAM Register Programming Aid – Walkthrough

- **MDASP: CS0_END** defines the ***absolute*** last address associated with CS0 with increments of 256Mb (32MB)

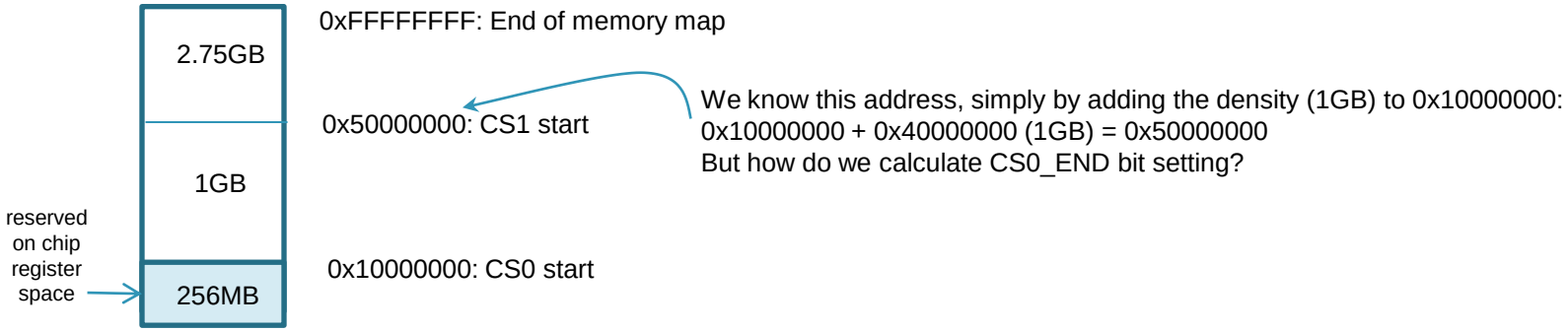
CS0_END bit settings:

```
Register bit setting
  0000000 – 256Mb(32MB)
  0000001 – 512Mb
  0000010 – 768Mb
  . . .
  0011111 – 8Gb  (1GB)
  0111111 – 16Gb (2GB)
  1111111 – 32Gb (4GB)
```

➢ Note, these offsets *start* from address 0x0, in general CS0 starts at 0x10000000; these are not offsets from 0x10000000!

➢ So, a CS0_END setting of 0000000 means that CS0 ends at 0x02000000 (before it begins).. Yeah, this setting is meaningless

- Actually, any setting from 0000000 to 0000111 is meaningless (0000111 is 256MB (2Gb) which is 0x10000000)
- Example, assume there is 1GB (8Gbits) of DDR3 memory on CS0



To calculate CS0_END setting, first remember this is the absolute address starting from 0x0.
Next, as seen in the figure above, calculate the end address of CS0 by adding the DRAM density to 0x10000000.
Now that we have the end (absolute) address, take this value and divide by 256MB (32MB) then subtract 1 (since CS0_END = 0 starts at 256Mb).

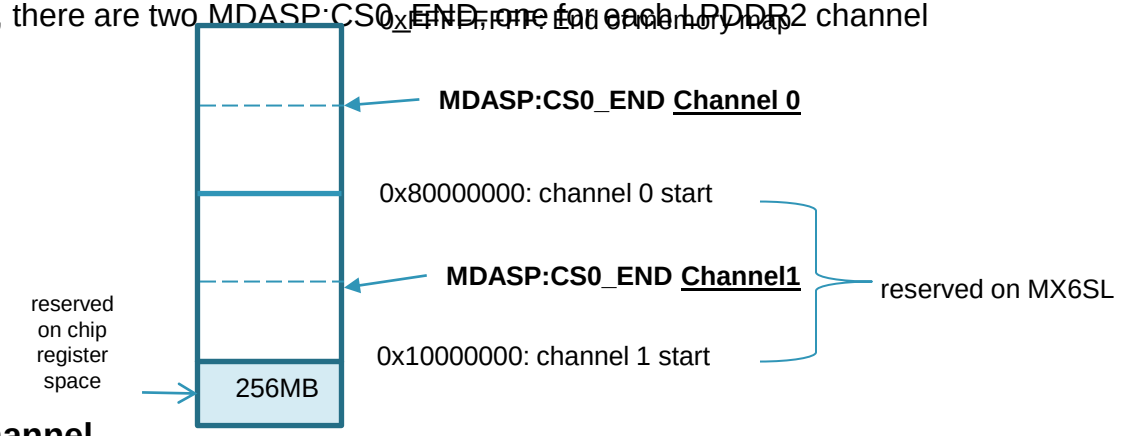
Taking the example above, density = 1GB, so CS0 end is 0x10000000 + 0x40000000 = 0x50000000
(0x50000000 / 0x2000000) - 1 = 39 decimal (or 0x27) (or 0100111)

Generically: [(0x10000000 + density_MB) / 0x2000000] – 1 = CS0_END setting

32MB = 0x2000000

DRAM Register Programming Aid - Walkthrough

- **MDASP: CS0_END 2-channel LPDDR2**
  - Each channel has a set starting address
    - Channel 0: 0x80000000
    - Channel 1: 0x10000000
  - You can use a 1-channel LPDDR2 device, need to pick a channel (channel 0 is most common choice)
    - MX6SL (aka Megrez) only has channel 0, so MMDC DRAM memory space starts at 0x80000000
  - For MX6DQ (Arik) and MX6DL (Rigel), there are two MDASP:CS0_END, one for each LPDDR2 channel



- **Calculate CS0_END for each channel**
  - Channel 0: [(0x80000000 + density_MB) / 0x2000000] – 1

Example1: 2 chip select, each with 512MB, CS0_END=[(0x80000000+0x20000000)/0x2000000]-1=0x4F

Example2: 2 chip select, each with 256MB, CS0_END=[(0x80000000+0x10000000)/0x2000000]-1=0x47

  - Channel 1: [(0x10000000 + density_MB) / 0x2000000] – 1 (not for MX6SL)

Example1: 2 chip select, each with 512MB, CS0_END=[(0x10000000+0x20000000)/0x2000000]-1=0x17

Example2: 2 chip select, each with 256MB, CS0_END=[(0x10000000+0x10000000)/0x2000000]-1=0xF

DRAM Register Programming Aid - Walkthrough

DDR3 MR2 Parameter or MMDC MDSCR Parameter	N/A	control bit setting (decimal)	bit setting within register	Notes	Register name	Register address	Register value (HEX)
MR2: RTT	-	2	04000000	Dynamic ODT (RTT(WR)). 00-disable; 01-RZQ/4; 10-RZQ/2; 11-reserved (RZQ=240ohm)	MDSCR	0x021B001C	0x04088032
MR2: SRT	-	0	00000000	SRT: Self refresh temperature, set to 0 for normal operation			
MR2: ASR	-	0	00000000	ASR: Auto self refresh, set to 0 for normal operation			
MR2: CWL	-	6	00080000	CAS Write Latency. This value is taken from the MMDC tCWL parameter setting above. Do not modify this bit as it is automatically programmed.			
CON_REQ	-	1	00008000	Configuration request - set to 1 for this operation.			
WL_EN	-	0	00000000	Set to 0; not applicable to mode register programming.			
CMD	-	3	00000030	CMD: set to 0x3 for load mode register command.			
CMD_CS	-	0	00000000	Determines which chip select command is targetted to.			
CMD_BA	-	2	00000002	CMD_BA - set to 0x2 for MR2			
DDR3 MR3 Parameter or MMDC MDSCR Parameter	**N/A**	**control bit setting (decimal)**	**bit setting within register**	**Notes**	**Register name**	**Register address**	**Register value (HEX)**
MR3: MPR	-	0	00000000	MPR enable - set to 0 for normal operation	MDSCR	0x021B001C	0x00008033
MR3: MPR_RF	-	0	00000000	Set to 0 for normal operation			
CON_REQ	-	1	00008000	Configuration request - set to 1 for this operation.			
WL_EN	-	0	00000000	Set to 0; not applicable to mode register programming.			
CMD	-	3	00000030	CMD: set to 0x3 for load mode register command.			
CMD_CS	-	0	00000000	Determines which chip select command is targeted to.			
CMD_BA	-	3	00000003	CMD_BA - set to 0x3 for MR3			
DDR3 MR1 Parameter or MMDC MDSCR Parameter	**N/A**	**control bit setting (decimal)**	**bit setting within register**	**Notes**	**Register name**	**Register address**	**Register value (HEX)**
MR1: Q Off	-	0	00000000	Set to 0 for normal operation	MDSCR	0x021B001C	0x00048031
MR1: TDQS	-	0	00000000	Termination data strobe (TDQS) is a function of the x8 DDR3 SDRAM configuration; set to 0 for x16 and x32 memories			
MR1: RTT (M9)	-	0	00000000	On-die termination (ODT) resistance RTT. 000-disabled; 001-RZQ/4; 010-RZQ/2; 011-RZQ/6; 100-RZQ/12; 101-RZQ/8; 110&111-reserved (RZQ=240ohm)			
MR1: RTT (M6)	-	0	00000000				
MR1: RTT (M2)	-	1	00040000				
MR1: WL	-	0	00000000	Write leveling enable - set to 0 for normal operation			
MR1: ODS (M5)	-	0	00000000	Output Drive Strength: 00-RZQ/6 (40ohm); 01-RZQ/7 (34ohm); 10&11-reserved			
MR1: ODS (M1)	-	0	00000000				
MR1: AL	-	0	00000000	AL: Additive Latency, set to 0			
MR1:DLL	-	0	00000000	DLL Enable - set to 0			
CON_REQ	-	1	00008000	Configuration request - set to 1 for this operation.			
WL_EN	-	0	00000000	Set to 0; not applicable to mode register programming.			
CMD	-	3	00000030	CMD: set to 0x3 for load mode register command.			
CMD_CS	-	0	00000000	Determines which chip select command is targeted to.			
CMD_BA	-	1	00000001	CMD_BA - set to 0x1 for MR1			
DDR3 MR0 Parameter or MMDC MDSCR Parameter	**N/A**	**control bit setting (decimal)**	**bit setting within register**	**Notes**	**Register name**	**Register address**	**Register value (HEX)**
MR0: PD	-	0	00000000	Precharge power-down (PD) - set to 0 for normal operation	MDSCR	0x021B001C	0x09408030
MR0: WR	-	4	08000000	Write recovery. 000-16; 001-5; 010-6; 011-7; 100-8; 101-10; 110-12; 111-14. Make sure to match MMDC tWR.			
MR0: DLL	-	1	01000000	DLL reset - set to 1 to reset DLL; self clearing			
MR0: BT	-	0	00000000	Burst type: set to 0			
MR0: CL (M6)	-	1	00400000	CAS latency: 0010-5; 0100-6;0110-7;1000-8; 1010-9; 1100-10; 1110-11; 0001-12; 0011-13. Make sure to match CAS to MMDC tCL			
MR0: CL (M5)	-	0	00000000				
MR0: CL (M4)	-	0	00000000				
MR0: CL (M2)	-	0	00000000				
MR0:BL	-	0	00000000	Burst length - set to 00 for fixed 8 burst length			
CON_REQ	-	1	00008000	Configuration request - set to 1 for this operation.			
WL_EN	-	0	00000000	Set to 0; not applicable to mode register programming.			
CMD	-	3	00000030	CMD: set to 0x3 for load mode register command.			
CMD_CS	-	0	00000000	Determines which chip select command is targeted to.			
CMD_BA	-	0	00000000	CMD_BA - set to 0x0 for MR0			

Mode Register #

DRAM Register Programming Aid - Walkthrough

If CS1 populated							
DDR3 MR2 Parameter or MMDC MDSCR Parameter	**N/A**	**control bit setting (decimal)**	**bit setting within register**	**Notes**	**Register name**	**Register address**	**Register value (HEX)**
MR2: RTT	-	2	04000000	Dynamic ODT (RTT(WR)). 00-disable; 01-RZQ/4; 10-RZQ/2; 11-reserved (RZQ=240ohm)	MDSCR	0x021B001C	0x0408803A
MR2: SRT	-	0	00000000	SRT: Self refresh temperature, set to 0 for normal operation			
MR2: ASR	-	0	00000000	ASR: Auto self refresh, set to 0 for normal operation			
MR2: CWL	-	6	00080000	CAS Write Latency. This value is taken from the MMDC tCWL paramter setting above. Do not modify this bit as it is automatically programmed.			
CON_REQ	-	1	00008000	Configuration request - set to 1 for this operation.			
WL_EN	-	0	00000000	Set to 0; not applicable to mode register programming.			
CMD	-	3	00000030	CMD: set to 0x3 for load mode register command.			
CMD_CS	-	1	00000008	Determines which chip select command is targetted to.			
CMD_BA	-	2	00000002	CMD_BA - set to 0x2 for MR2			

DDR3 MR3 Parameter or MMDC MDSCR Parameter	**N/A**	**control bit setting (decimal)**	**bit setting within register**	**Notes**	**Register name**	**Register address**	**Register value (HEX)**

Rest of CS1 mode register programming is exactly the same as CS0

- If board doesn't populate CS1, can still perform mode register writes to CS1 even though nothing there, but preferably…
- Other option, remove CS1 mode register writes in "RealView .inc file" tab:

```
// Mode register writes
setmem /32    0x021b001c =    0x04088032    // MMDC0_MDSCR, MR2 write, CS0
setmem /32    0x021b001c =    0x00008033    // MMDC0_MDSCR, MR3 write, CS0
setmem /32    0x021b001c =    0x00048031    // MMDC0_MDSCR, MR1 write, CS0
setmem /32    0x021b001c =    0x09408030    // MMDC0_MDSCR, MR0 write, CS0
setmem /32    0x021b001c =    0x04008040    // MMDC0_MDSCR, ZQ calibration command sent to device on CS0

setmem /32    0x021b001c =    0x0408803A    // MMDC0_MDSCR, MR2 write, CS1
setmem /32    0x021b001c =    0x0000803B    // MMDC0_MDSCR, MR3 write, CS1
setmem /32    0x021b001c =    0x00048039    // MMDC0_MDSCR, MR1 write, CS1
setmem /32    0x021b001c =    0x09408038    // MMDC0_MDSCR, MR0 write, CS1
setmem /32    0x021b001c =    0x04008048    // MMDC0_MDSCR, ZQ calibration command sent to device on CS1

setmem /32    0x021b0020 =    0x00005800    // MMDC0_MDREF
setmem /32    0x021b0818 =    0x00022227    // DDR_PHY_P0_MPODTCTRL
setmem /32    0x021b4818 =    0x00022227    // DDR_PHY_P1_MPODTCTRL
```

DRAM Register Programming Aid - Walkthrough

- The final configurable parameters in the Register Configuration tab – calibration values

	Register name	Register address	Register value (HEX)
These parameters are determined after running calibration. The parameters provided here are from Freescale's development board and will work as initial values. Update these values after running calibration.	MPDGCTRL0 PHY0	0x021b083c	0x434b0350
	MPDGCTRL1 PHY0	0x021b0840	0x034c0359
	MPDGCTRL0 PHY1	0x021b483c	0x434b0350
	MPDGCTRL1 PHY1	0x021b4840	0x03650348
	MPRDDLCTL PHY0	0x021b0848	0x4436383b
	MPRDDLCTL PHY1	0x021b4848	0x39393341
	MPWRDLCTL PHY0	0x021b0850	0x35373933
	MPWRDLCTL PHY1	0x021b4850	0x48254a36
These are for write leveling calibration, which is needed for fly-by board layout topology	MPWLDECTRL0 PHY0	0x021b080c	0x001F001F
	MPWLDECTRL1 PHY0	0x021b0810	0x001F001F
	MPWLDECTRL0 PHY1	0x021b480c	0x00440044
	MPWLDECTRL1 PHY1	0x021b4810	0x00440044

Read DQS gating

Read delay line:
read DQS-to-DQ delay

Write delay line:
write DQS-to-DQ delay

Write leveling

If you run calibration, then you can update these values with the values found in calibration

more on calibration later …

Agenda

- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

DRAM Stress Test – Overview and History

- Took years to develop, constantly evolving to catch elusive DRAM failures
- Compilation of various DRAM sub tests
  - Each sub test contains various data patterns/methods to stress the DRAM interface
  - Started with a few tests using memcpy and various data patterns (1's and 0's; A's and 5's; pseudo-random, etc)
  - Each new SoC/board presented new DRAM challenges/issues
    - New tests were created to debug
  - Historically, each test was run one-by-one, took time
  - Tests were compiled together into one overall test
    - Each test now called sub-test, executed through a function call
    - Marked beginning of 'stress test', sub-tests run in a loop, overnight
  - Cache enabled – important, needed to mimic OS-type transactions; more stress
  - Increment DRAM frequency – method to stress interface accounting for variations in PVT
    - How much above frequency max is considered ample?
      - Historically 30MHz or more seemed good
    - Useful for gathering statistical data; outliers may point to other issues

DRAM Stress Test – Overview and History (Continued)

- Non-OS test to exercise DRAM interface
  - Non-OS: easier than OS to catch/debug DRAM failures
  - Used by factory as part of DRAM validation
  - Helps diagnose but doesn't fix DDR problems
- Purpose: Root out potential signal integrity issues due to inadequate board layout
  - Primarily uses sequential bursts of back-to-back data looking for simultaneous switching noise (SSN)
  - Validation vehicle that reports how robust DRAM interface is given current set of parameters (i.e. drive strength settings, timing parameters, board layout, etc)
- Runs from internal RAM
  - Device under test is DRAM itself, don't execute out of same memory being tested
  - Download to IRAM via JTAG debugger tools (RVD, Lauterbach, Macraigor)
  - Now available in a USB version.
- FAE's are able to provide tailored code if necessary.
  - Any debugger that supports specific SoC ARM core and elf should work
  - Factory not responsible to test every debugger or debug it if doesn't work on other debuggers

DRAM Stress Test – Overview and History (Continued)

- Once DRAM stress test passes with ample margin, are we guaranteed the OS will never fail due to DRAM issues?
  - High degree of confidence DRAM robust enough, but…
    - OS is still the most stressful, particularly an OS stress test like Bonnie++
    - Recommend to run any OS stress tests to double check
  - Currently Supported SoC:
    - MX28, MX508, MX51, MX53, MX6DQ (Arik) MX6DL (Rigel), MX6SL (Megrez), MX6SX (Pele)
  - No plans to back port to older legacy processors
    - Issues encountered as some only have 16KB of IRAM
- Challenges
  - Test becoming too big to fit inside IRAM (128KB becoming a limiting factor)
  - When new sub-tests are created, no plan in place to back port to older processors

DRAM Stress Test – How to Run
Debugger Setup

- Stress test outputs .elf; can be run from various debuggers
  - Debuggers necessary for new SoC bring up; highly useful for new board bring up
  - Debuggers don't rely on bootloaders or on anything running
    - Simply having a debugger connect means the SoC is powered up and running
  - Debuggers allow user to quickly test fundamental DRAM init
    - Open a debugger memory window to perform simple write-read-verify
  - There are less expensive debugger options than ARM (like Macraigor)
    - May give up some functionality, but good enough to download and run stress test
- To run from a debugger
  - Run DRAM init (`.inc`, `.ds`, `.cmm`, `.mac`)
    - Refer to DRAM Register Programming Aid tool
  - Load and run the .elf file

DRAM Stress Test – How to Run
Serial Port Setup

- DRAM Stress Test uses serial port output (UART) for user interaction
- Terminal program needed on the Host PC
  - Host PC: Tera Term or Hyperterminal
    - Ensure the correct COM port usage
  - Set up:

```
BAUD RATE - 115200
DATA - 8 bit
PARITY- none
STOP - 1bit
FLOW CONTROL – none
```

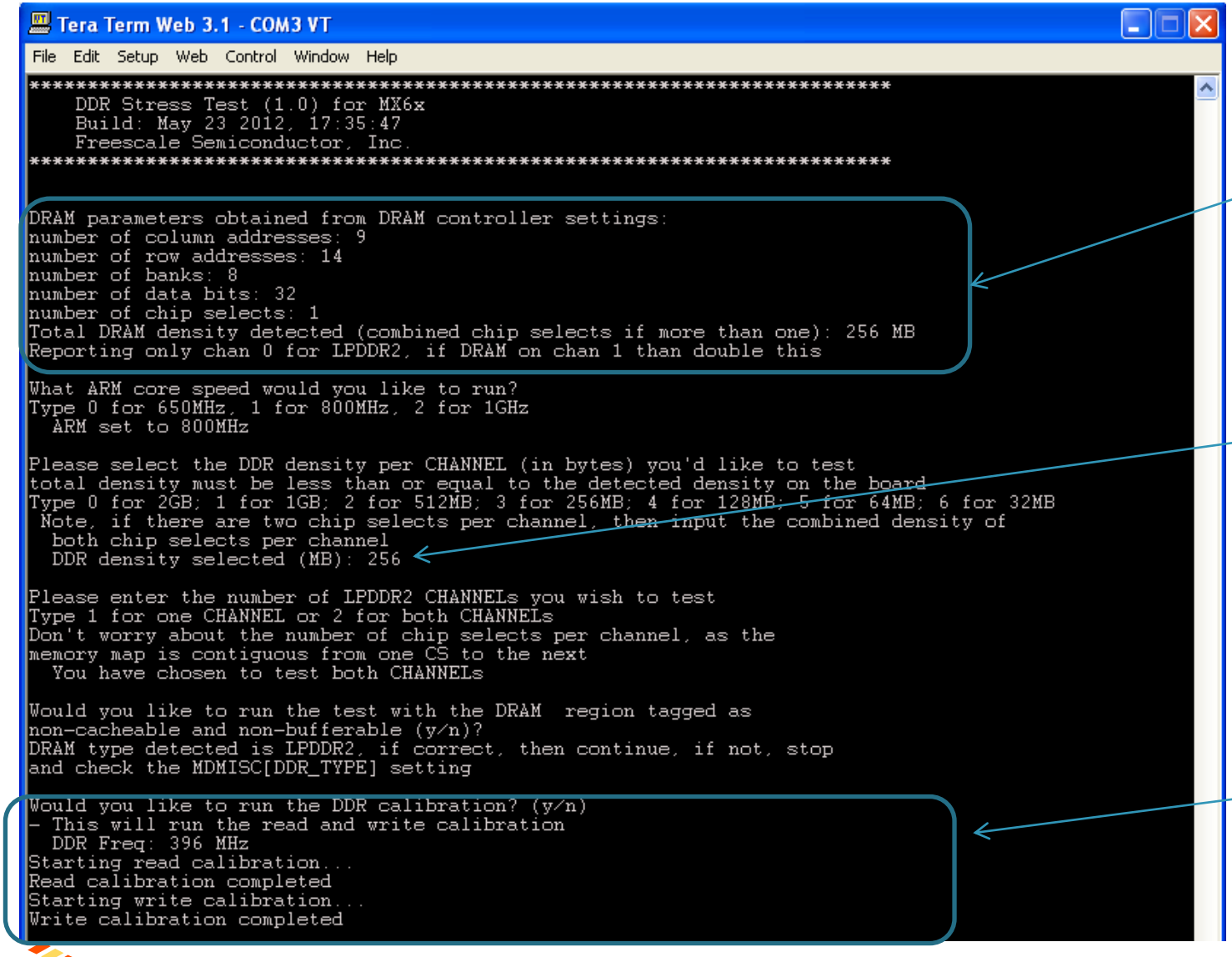
- Once test runs, look for output messages on the terminal
  - Various run control options
  - DRAM density to test, frequency range, etc

DRAM Stress Test – How to Run
USB Option

- Makes use of *.bin file output.
- Need all in the same windows folder:
  - USB executable file (DDR_Stress_Tester)
  - *.bin file generated from source code (ddr-stress-test-<TGT>.bin)
  - *.inc script file (from Register Programming Aid)
- Enter at the command prompt:
  - DDR_STRESS_TESTER –t <tgt> -df <prog_aid> -usb
  - DDR_STRESS_TESTER –h (Help Menu)
- Allowable targets are:
  - mx53
  - Mx6x
- Test runs from Windows command prompt just like Serial UART Terminal.

M Stress Test – How to Run

Serial Port Screen Shot – MX6Q LPDDR2 validation board example



Newer DRAM stress tests (MX6 and MX508) include this; needed for newer “row hop” test described later

For quick testing, you can choose a smaller density than is on the board

MX53 and MX6 DRAM controllers support calibration
MX508 only does ZQ calibration

DRAM Stress Test – Overview of DRAM Subtests

- memcpy11 SSN test (*DDRtest_memcpy11_SSN.c*)

```
int DDRtest_memcpy11_SSN(u32 density, u32 number_of_cs)
int DDRtest_memcpy10_SSN_x64(u32 density, u32 number_of_cs)
```

  - This test utilizes a custom written memory copy function that issues 11-word load and store multiple instructions, to test the bursting behavior of the DDR interface. This test uses data patterns to help root out SSN. This test also breaks up the total DDR density into four “banks”, where each bank contains a different SSN data pattern. The test uses various stress patterns such as walking ones and walking zeros, and 0xA’s followed by 0x5’s.
  - For a 64-bit data bus size, a memcpy10 routine is used instead to better align with the 64-bit interface. Also, this test uses data words of size “long long”, or 64-bit data words. This is necessary to effectively test the SSN data pattern.

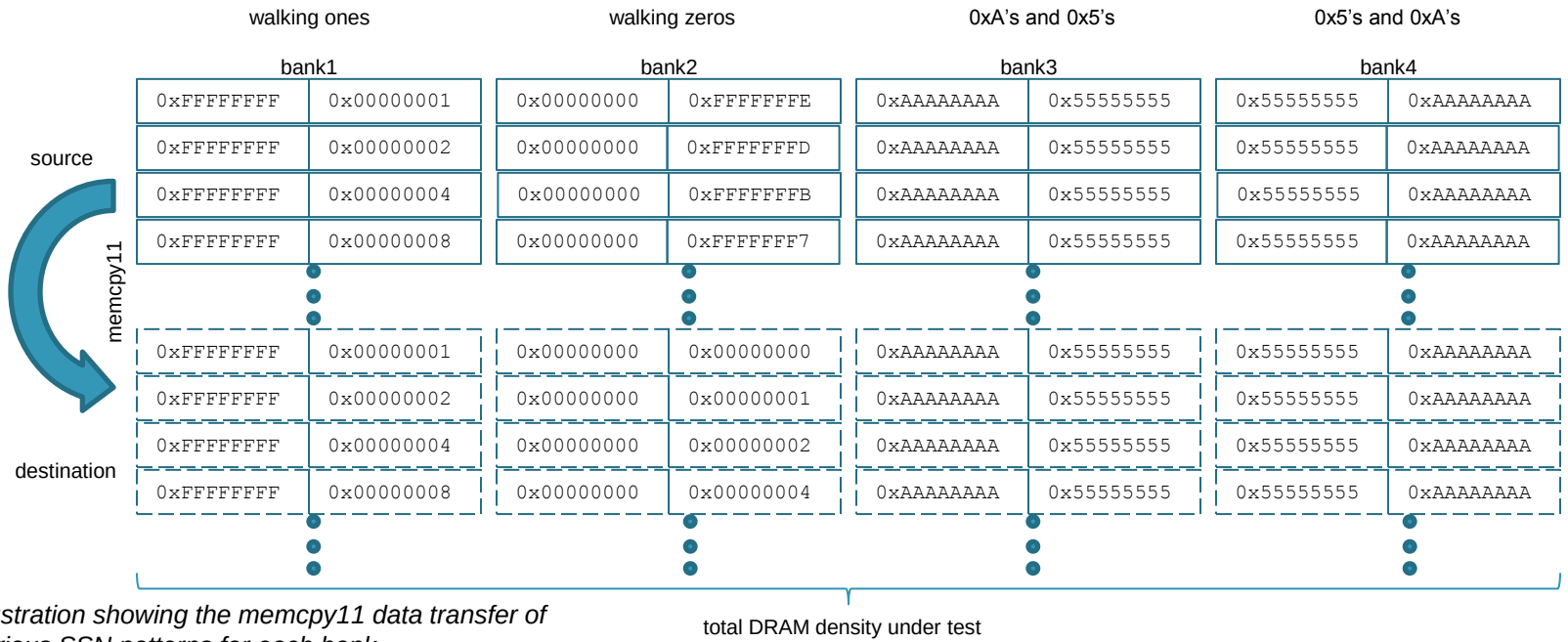


Illustration showing the memcpy11 data transfer of various SSN patterns for each bank

DRAM Stress Test – Overview of DRAM Subtests (Continued)

- memcpy8 SSN test (*DDRtest_SSN_memcpy_test.c*)

```
int DDRtest_SSN_memcpy_test(u32 density, u32 number_of_cs)
int DDRtest_SSN_memcpy_test_x64(u32 density, u32 number_of_cs)
```

  - This test performs “memcpy” bursts of various stress patterns (walking ones and walking zeros, 0xA’s followed by 0x5’s, etc) using *copy_words_using_eight_registers* to help determine or weed out simultaneous switching signal noise. In this case, the memory copy function used issues 8-word load and store multiple instructions.
  - For a 64-bit data bus size, each data pattern above is expanded into a 64-bit data size

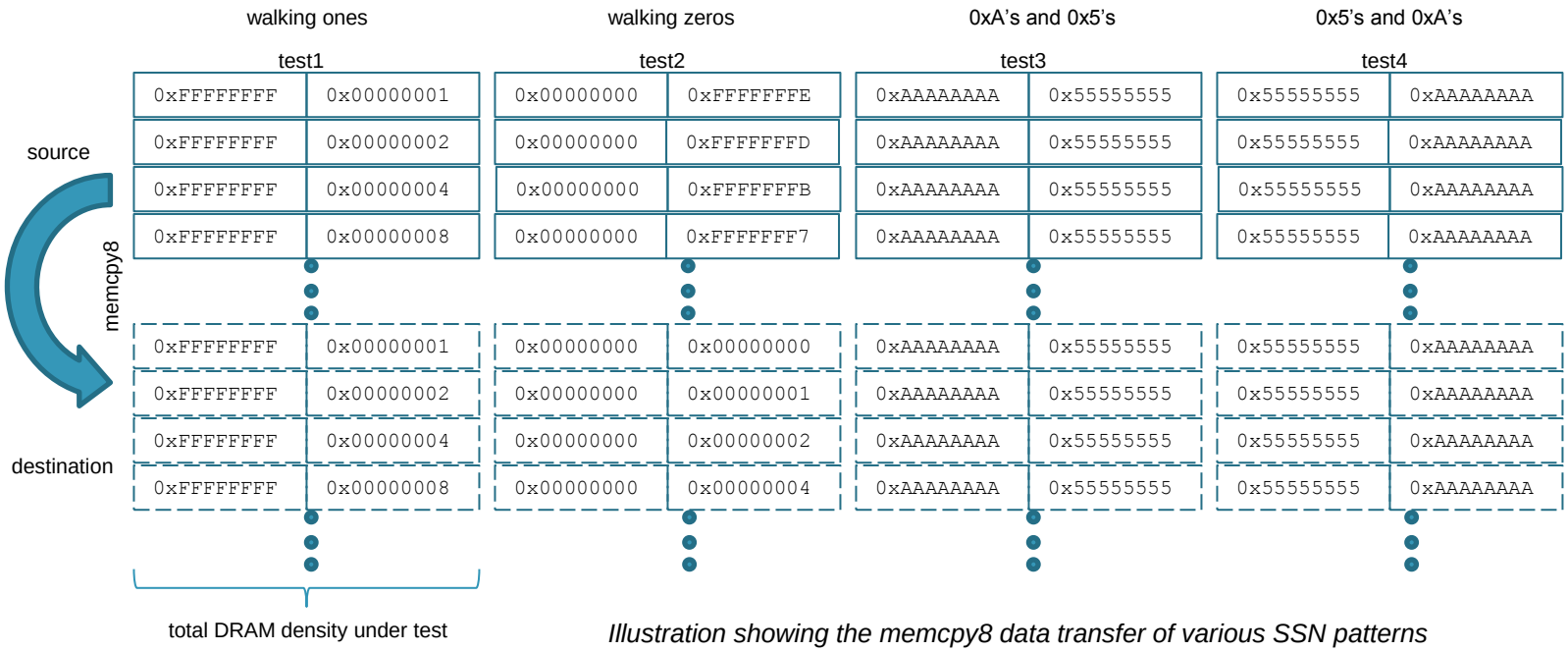


Illustration showing the memcpy8 data transfer of various SSN patterns

DRAM Stress Test – Overview of DRAM Subtests (Continued)

- byte-wise SSN test (*DDRtest_byte_SSN_test.c)*

```
int DDRtest_byte_SSN_test(u32 density, u32 number_of_cs)

int DDRtest_byte_SSN_test_x64(u32 density, u32 number_of_cs)
```

  - The purpose of this test is to root out any SSN within byte lanes. It accomplishes this by writing byte-wise patterns to one location and the inverse of each byte to the subsequent location. Each of the four bytes values are equal to one another and the test increments the byte pattern as follows (with the inverse value in brackets: 0x00000000 [0xFFFFFFFF]; 0x01010101 [0xFEFEFEFE]; 0x02020202 [0xFDFDFDFD]; … 0xFFFFFFFF [0x00000000].
  - For a 64-bit data bus size, each data pattern above is expanded into a 64-bit data size

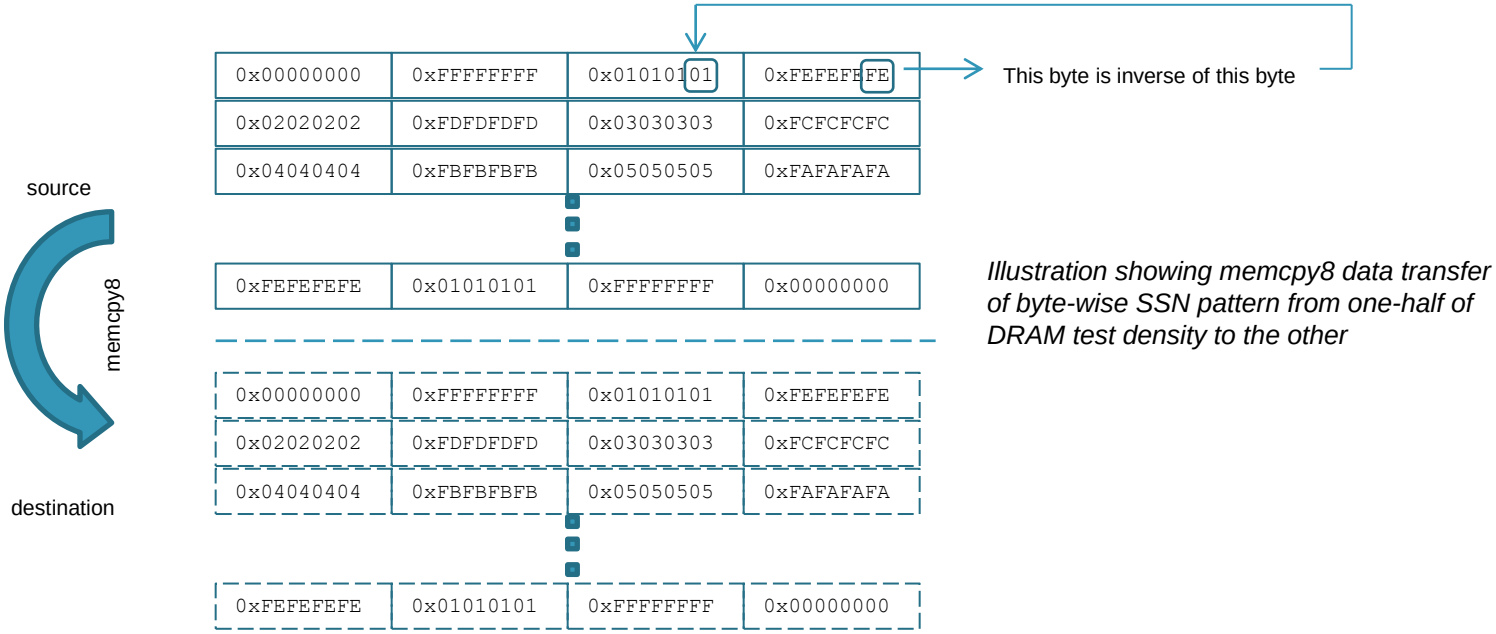


Illustration showing memcpy8 data transfer of byte-wise SSN pattern from one-half of DRAM test density to the other

DRAM Stress Test – Overview of DRAM Subtests (Continued)

- memcpy11 random pattern test (*DDRtest_memcpy11_pseudo_random_with_offset.c)*

```
int DDRtest_memcpy11_pseudo_random_with_offset(u32 density, u32 number_of_cs)
```

  - This test utilizes a custom written memory copy function that issues 11-word load and store multiple instructions, to test the bursting behavior of the DDR interface. Each time through the memcpy11 test, the start address is incremented in steps of four bytes to test different burst access types, especially important when caches are enabled to initiate burst wraps for cache line fills. The data pattern used is pseudo-random. This test also breaks up the total DDR density into four "banks", where each bank contains a different "seed" for each pseudo-random data pattern. Because the data patterns are pseudo-random in nature, the data bus size does not make a difference in the overall effectiveness of the pattern.

```
bank1: "pattern += 0x12345678" → 0x00000000, 0x12345678, 0x2468ACF0, etc …
bank2: "pattern += 0x11223344" → 0x00000000, 0x11223344, 0x22446688, etc …
bank3: "pattern += 0x87654321" → 0x00000000, 0x87654321, 0x0ECA8642, etc …
bank4: "pattern += 0x11112222" → 0x00000000, 0x11112222, 0x22224444, etc …
```

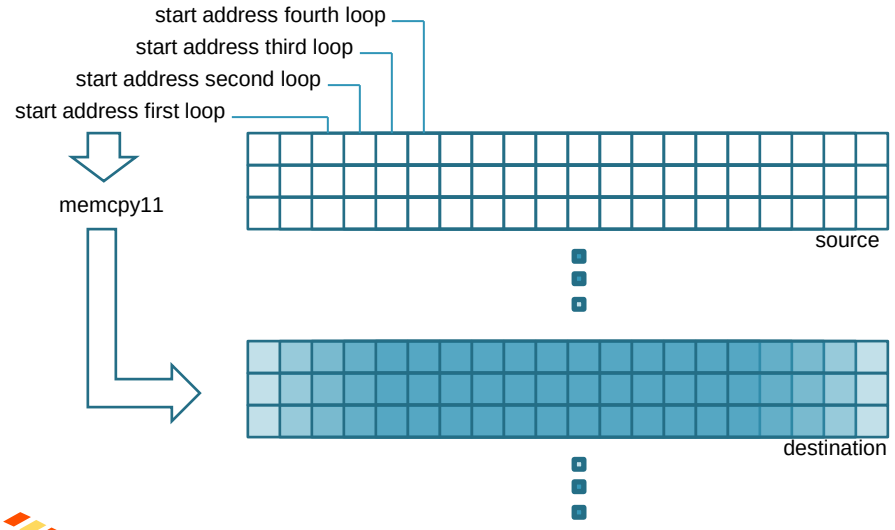


Illustration showing memcpy11 data transfers starting a different address offsets (only one bank shown)

DRAM Stress Test – Overview of DRAM Subtests (Continued)

- IRAM_to_DDRv2 test (*DDRtest_IRAM_to_DDRv2.c*)

```
int DDRtest_IRAM_to_DDRv2(u32 iram_start, u32 iram_test_length, u32 number_of_cs)
```

  - The purpose of this test is to root out any SSN and isolates the DDR read and write accesses by using the IRAM as an intermediate data storage location. This test moves a user defined length of data using the *copy_words_using_eight_registers* function from DDR to IRAM and then from IRAM to a different DDR location, then compares DDR location 1 and location 2. This test is similar to the IRAM_to_DDRv1 test (described next), but instead transfers the data 1000 times per test to ensure that the data never changes to root out random glitches. Also, the test uses various data patterns to root out SSN.
- IRAM_to_DDRv1 test (*DDRtest_IRAM_to_DDR.c*)

```
int DDRtest_IRAM_to_DDR(u32 iram_start, u32 iram_test_length, u32 number_of_cs)
```

  - The purpose of this test is to root out any SSN and isolates the DDR read and write accesses by using the IRAM as an intermediate data storage location. This test moves a user defined length of data using the *copy_words_using_eight_registers* function from DDR to IRAM and then from IRAM to a different DDR location, then compares DDR location 1 and location 2.

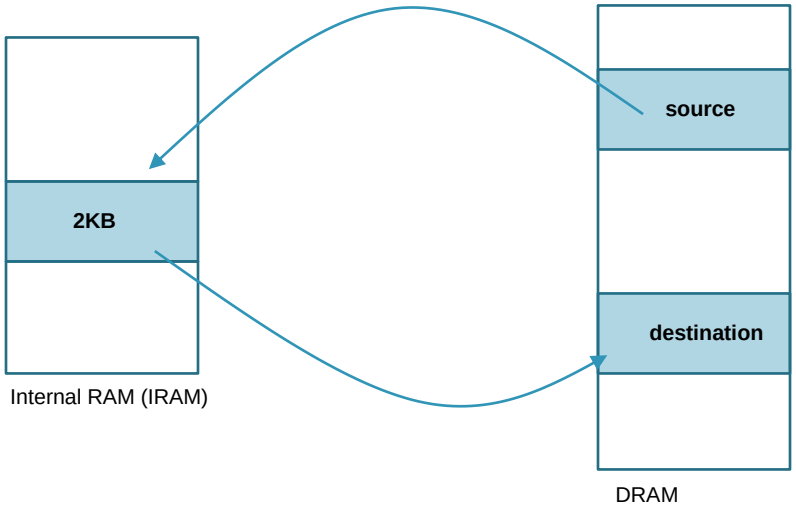


Illustration showing memcpy8 data transfers from DRAM location 1 to IRAM then to DRAM location 2

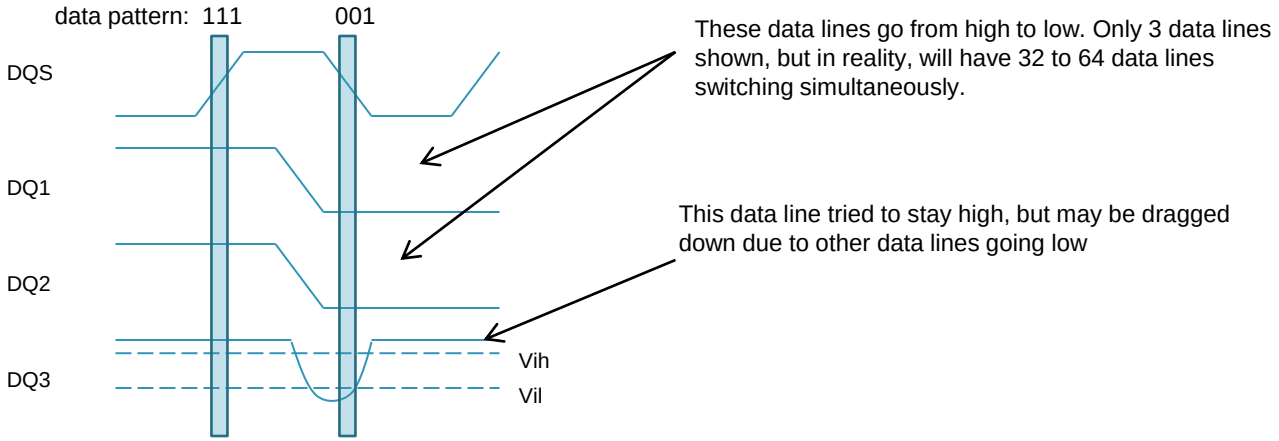
DRAM Stress Test – Overview of DRAM Subtests (Continued)

- read noise walking ones and zeros test (*DDRtest_read_noise_walking_test.c*)

```
int DDRtest_read_noise_walking_test(u32 density, u32 number_of_cs)
```

  - This test helps to identify if there any signal integrity issues or cross talk between data lines or other control signals by doing two sets of tests on both CSD0 and CSD1 (if CSD1 is populated) by performing either a walking ones test or walking zeros test. This is a quick test transferring only two back-to-back data words and less rigorous that previously described SSN tests, but is kept here for legacy purposes.
    - *Read from DDR walking ones* – this test writes a set of 0xFFFFFFFF followed by a walking ones pattern to memory and then reads it back in a two word burst (if there are issues, the read will not return a one)
    - *Read from DDR walking zeros test* – this test writes a set of 0x00000000 followed by a walking zeros pattern to memory and then reads it back in a two word burst (if there are issues, the read will not return zero)

Example of how the SSN patterns are used and what they are looking for

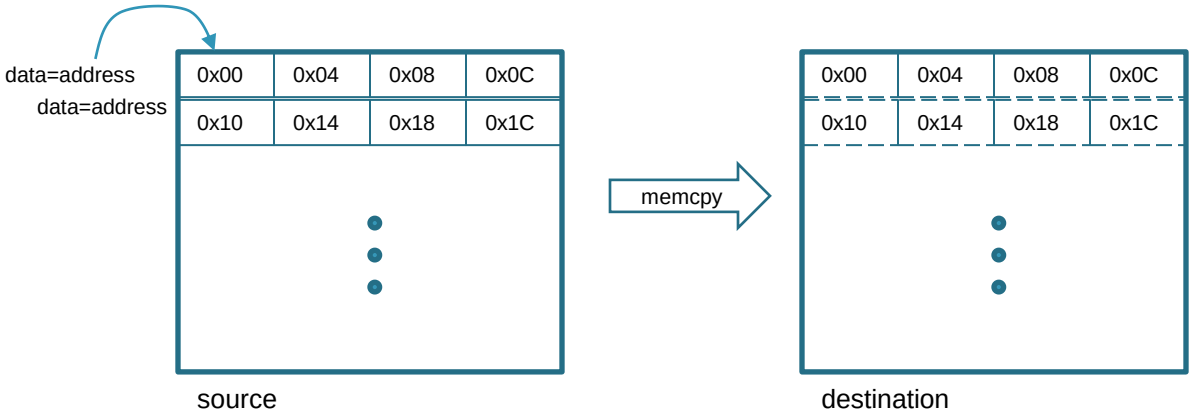


DRAM Stress Test – Overview of DRAM Subtests (Continued)

- data is addr test (*DDRtest_data_is_addr_memcpy.c*)

```
int DDRtest_data_is_addr_memcpy_test(u32 density, u32 number_of_cs)
```

  - This test basically programs the source buffer's data words with its address (and immediately read-verifies), then copies (memcpy) the source buffer to a destination buffer and verifies the data transferred correctly.
  - For example, if the source address being written to is "0x10000008", the data value of that address is "0x10000008". Once the source buffer is completely written, the data is transferred to the destination where it is verified.
  - This test is normally commented out in the stress test release due to the lengthy test time and can be re-enabled by un-commenting the line "#define TEST_ADDR_DATA" in the main_mx6x.c file and recompiling the stress test.
  - Not a very rigorous test, and due to lengthy test time, this test is not enabled in MX6 or MX508 by default
  - Test useful for new SoC bring up to ascertain simple read/writes work; helps to test cache since each write is immediately read-verified (which should hit in cache)



DRAM Stress Test – Overview of DRAM Subtests (Continued)

- row hop read test (*DDRtest_row_hop_read_test.c*)

```
int DDRtest_row_hop_read_test(u32 number_of_cs, u32 bank, u32 col, u32 row, u32 dsiz)
```

  - This test performs single word reads by hopping from one DRAM row to the next, reading the first column in each row and in each bank. Once all of the rows are read from the first column, the reads start again from the first row, and begin at the second column. This continues till all row, column, and bank data are read. The intent is to perform non-sequential reads and to force pre-charge and activate commands before each read access. A slight delay is added between reads to force idle time on the DRAM bus. This delay uses the ref_clock() function from drivers/timer.
  - In order to run this test, DRAM parameters such as number of row, column, and bank addresses, and data bus size must first be obtained. This test uses these parameters to determine the DRAM row, column, and bank boundaries to effectively perform single word reads by hopping from one DRAM row to the next.
  - These parameters are obtained in the main() function and passed into this function. These parameters are read from the SoC DRAM controller registers (which must first be initialized).
  - This test can be commented out in the stress test release due to the lengthy test time by commenting the line "#define ROW_HOP_READ_TEST" in the main_mx6x.c file and recompiling the stress test. The option to comment out this test is due to the lengthy test time. Unlike the other sub-tests whose test density can be less than the actual DRAM density in the system to shorten test time, this test will test the entire DRAM density in the system. The test is enabled by default for both MX6 and MX508.

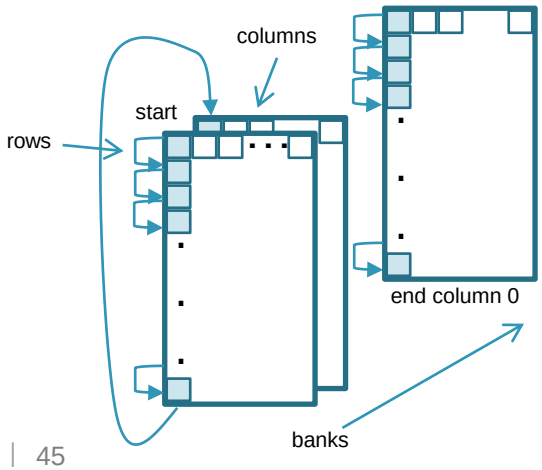


Illustration of the row hop read test

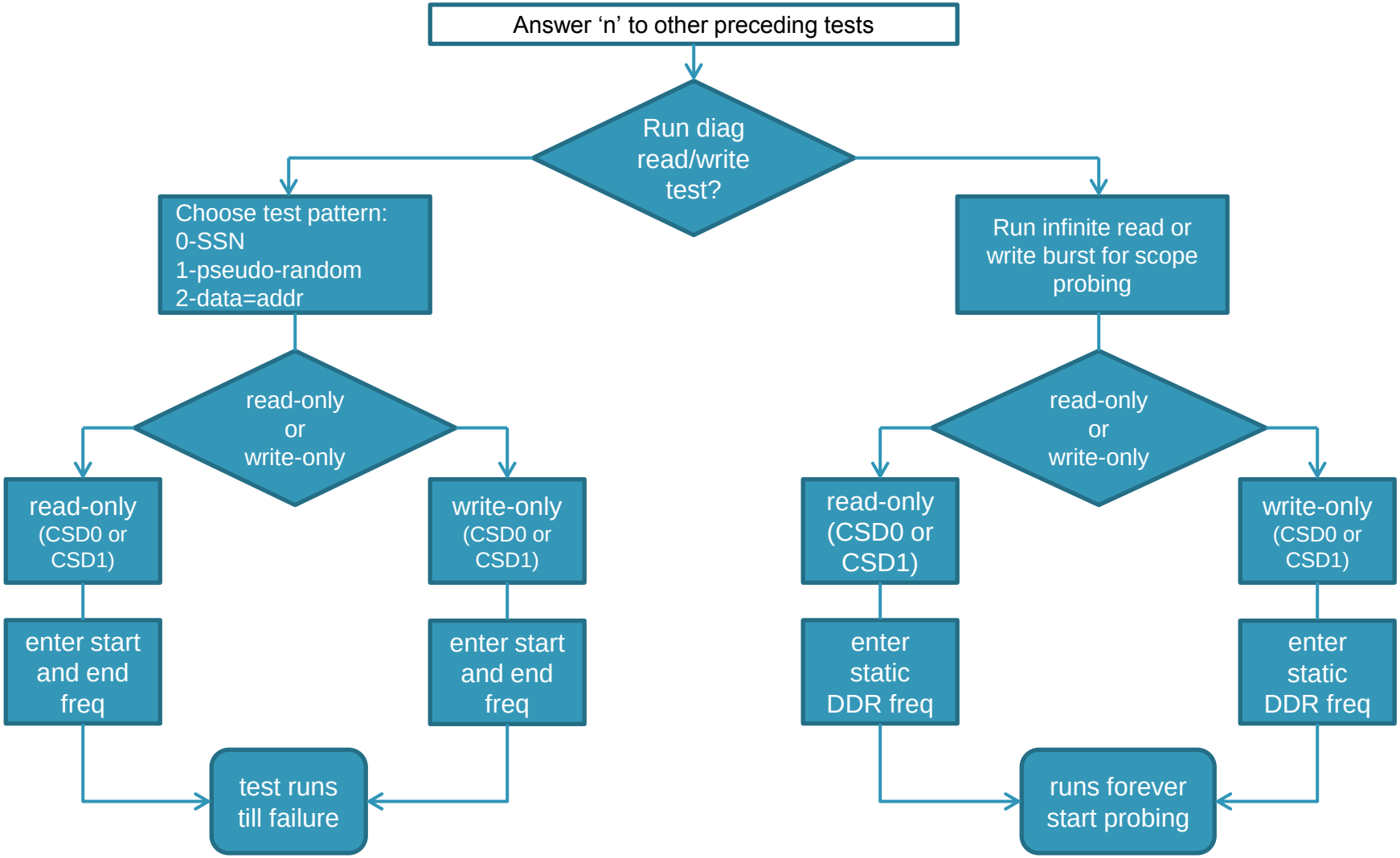
DRAM Stress Test – Overview of DRAM Subtests

mem_diag_read_write

- Memory diagnostics read- or write-only test
  - Runs either “read- or write-only test” or runs “read- or write-only infinite burst pattern” for scope probing
  - Not part of stress test’s “sub tests”; this is more of a “back door” test for more advanced users
  - To run diagnostic’s test, must select ‘n’ when asked to run other preceding tests (including “stress test”)
- C source file name: `main_mem_diag_read_write_only.c` (*recommend to go through this, code is best document*)
  - Found under supported SoC folder (mx6_common; mx53; mx508)
  - Function call: `mem_diag_read_write(u32 density, u32 dsiz);`
- Read-only or Write-only burst tests (memcpy) of various user-selected patterns
  - write-only test: source buffer in IRAM, destination in DRAM
  - read-only test: source buffer in DRAM, destination in IRAM
  - Patterns include: SSN, data-is-addr, pseudo-random
  - Like stress test, enter start and end freq for increasing freq test
  - Useful to help narrow down read or write issues, but not guaranteed; stress test is still de-facto tool
- Infinite read- or write-only burst transfers for scope probing
  - Isolates external bus transactions for scope probing purposes, pattern: 1’s and 0’s:
    - Write-only means i.MX is outputting signals
    - Read-only means DRAM is outputting signals

DRAM Stress Test – Overview of DRAM Subtests

mem_diag_read_write: flow chart



DRAM Stress Test – Interpreting DRAM Failures

- Failures observed may help narrow down a root cause; following are some pointers:
- Bit wise failures
  - Normally indicates one or more data lines experiencing glitch due to signal integrity issues (too slow rise/fall time, or too fast rise and fall time attributing to ringing)
  - Varying temperature is one method to narrow down the root cause
  - If cooling down the part causes more failures, then it is likely the drive strength is too high causing more overshoots and undershoots
  - If heating up the part causes more failures, then the drive strength is too low and the signals may not rise/fall fast enough
  - Playing around with drive strengths often help (start with i.MX side and then try DRAM side)
- Byte wise failures
  - This is usually indicative of a problem with the DQS signals: too slow a rise/fall time, there is a glitch, or over/under shoots (signal integrity issues)
  - Temperature testing (see bit-wise failures) to assist in narrowing down the issue with the DQS signal
  - Playing around with drive strengths help; also try playing around with DQS to data timing
- Entire word is wrong or random
  - This may indicate something more catastrophic either in the logic of the DRAM controller, or some issue with address and/or command signals
  - Could be a problem with board layout
  - Try playing around with DRAM controller's timings like 'RALAT'

DRAM Stress Test – Interpreting DRAM Failures (Continued)

- Keep in mind that these are only pointers to help diagnose DRAM related memory failures. In the past, DRAM failures were attributed to:
  - Poor board layout resulting in simultaneous switching noise (SSN) causing glitches.
  - Inadequate IO PAD design - often the IO PAD has poorly controlled impedance (either too high or too low a drive strength). This can be easily proven when observing a data signal or DQS signal being sourced by the DRAM memory (read access) or by the SoC (write access). One will often times observe that the DRAM memory provides a much cleaner waveform than that of the SoC.
  - Internal package issues (poor routing of power/ground signals, not providing proper ground returns, ground bounce, etc.)
  - DRAM controller logic bugs, often found in corner use cases where an internal bus master causes a transaction or series of subsequent transactions that were not anticipated or verified during the design process
  - Jitter on the SDCLK lines due to poor PLL design, or noise induced on the clock source itself (outside of the SoC from an external crystal or oscillator).
  - For LPDDR2, make sure i.MX IOMUX has pull down enabled for DQS signals
- Always remember to double-check DRAM initialization (DRAM register programming aid)

Agenda

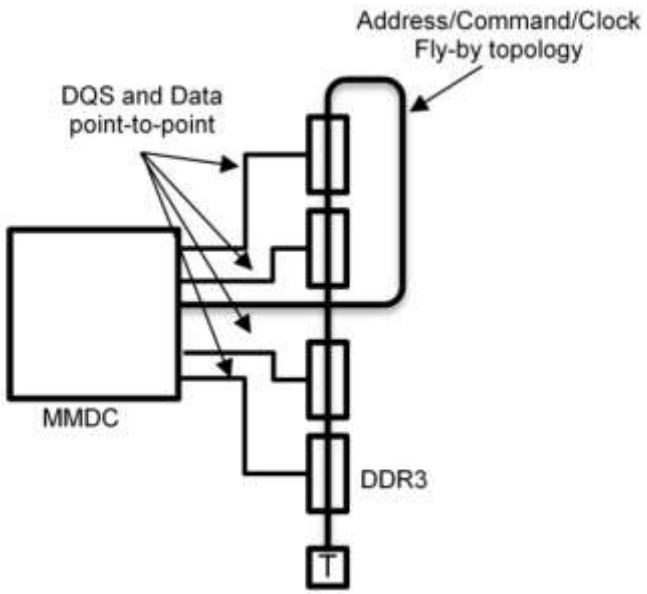
- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

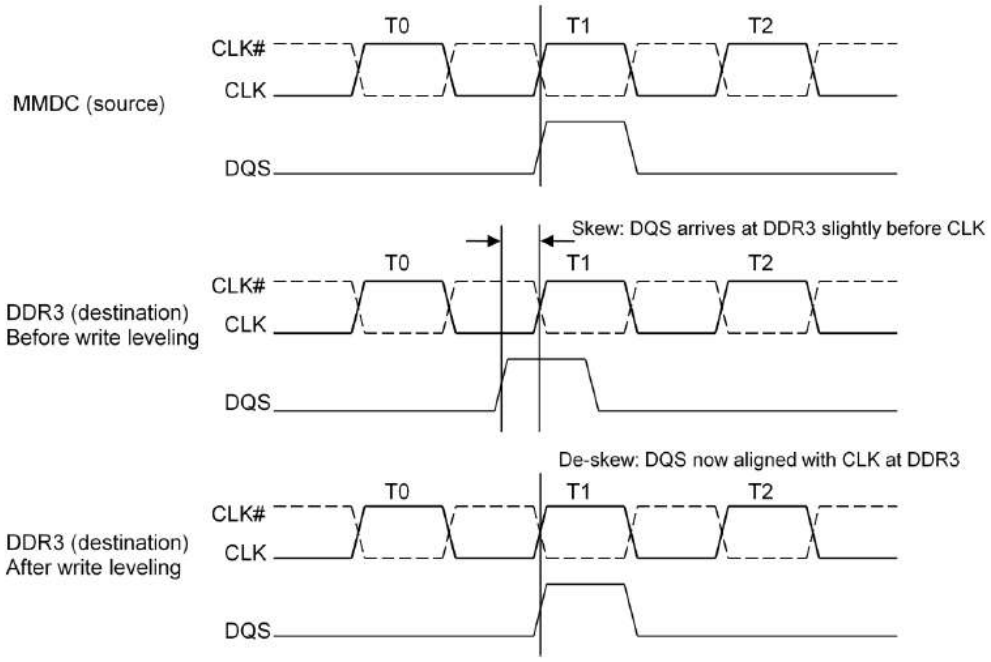
DRAM Calibration in the DRAM Stress Test

- MX6 MMDC Calibration App Note (ANXXXX): highly recommended reading to understand calibration concepts
- MX6 series DRAM controller (MMDC) features HW supported calibration methods:
  - Read DQS Gating calibration
  - Read DQS delay calibration
  - Write DQS delay calibration
  - Write-leveling calibration
- Previous i.MX SoC did not have this support in HW (with the exception of MX53)
  - That doesn't mean someone didn't come along and write their own type of SW calibration code for previous parts
- ZQ calibration is something simply enabled, no user interaction
  - Exception is MX508 where this requires a manual SW ZQ calibration routine
    - This routine is part of the MX508 DRAM Stress Test
    - This routine was obtained directly from design/validation and re-used in stress test
- DRAM Stress Tests that support calibration in some form or fashion:
  - MX6: Read DQS gating; read and write calibration; write-leveling
    - MX6 Calibration app note describes other calibration methods, but these are manual or for fine tuning and are not commonly performed and are not supported in the stress test
- MX508: manual ZQ calibration
- MX53: Read DQS gating; read and write calibration
- All code received directly from design/validation and ported to stress test

DRAM Calibration Conceptual Overview

Write Leveling Calibration





From the MX6DQ register programming aid example

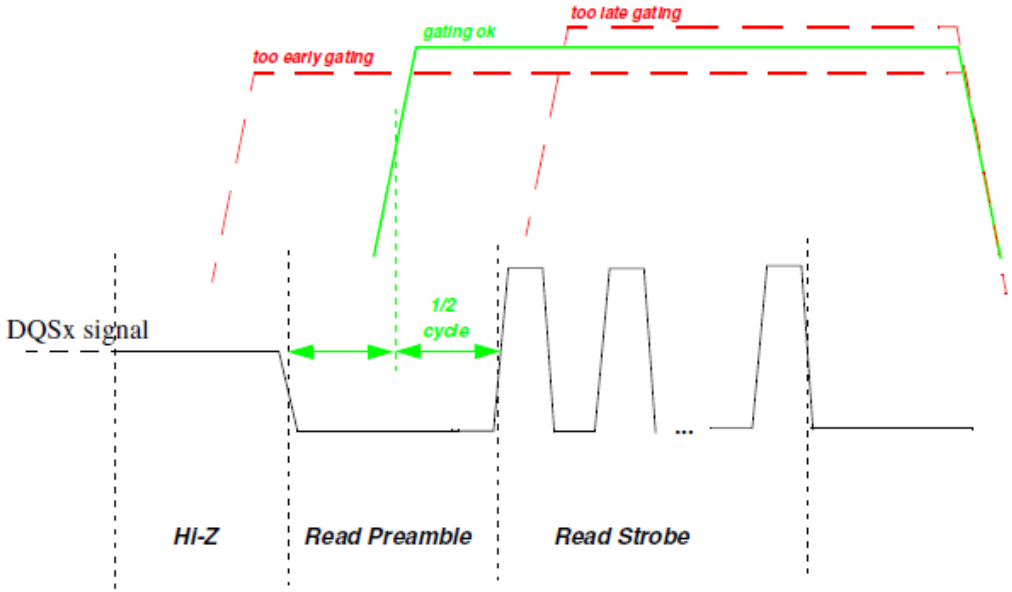
These are for write leveling calibration, which is needed for fly-by board layout topology	MPWLDECTRL0 PHY0	0x021b080c	0x001F001F
	MPWLDECTRL1 PHY0	0x021b0810	0x001F001F
	MPWLDECTRL0 PHY1	0x021b480c	0x00440044
	MPWLDECTRL1 PHY1	0x021b4810	0x00440044

Values from validation board. For Sabre boards, which is routed point-to-point, can set all values to 0x1F.

- Compensates for CLK-to-DQS skew incurred from fly-by topology
  - *Point-to-point memory layouts don't need this*
  - Point-to-point memory layouts normally don't use terminations, like FSL development boards
- Relevant to DDR3 memories only (not supported with LPDDR2)
- Write Leveling Calibration code in MX6 DRAM Stress Test

DRAM Calibration Conceptual Overview

Read DQS Gating Calibration

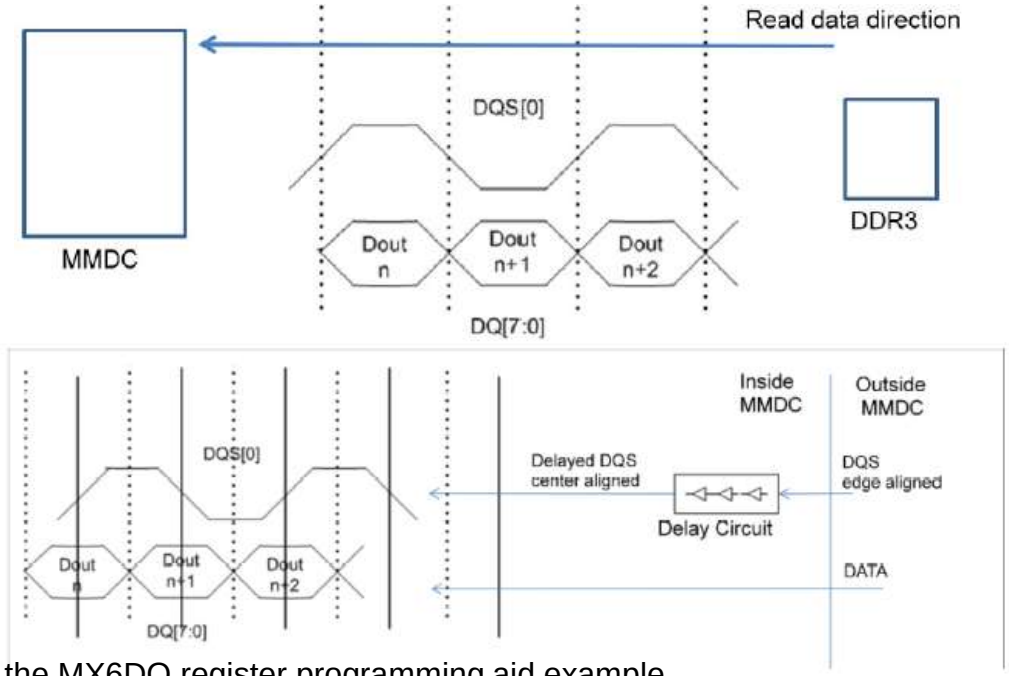


From the MX6DQ register programming aid example

These parameters are determined after running calibration. The parameters provided here are from Freescale's development board and will work as initial values. Update these values after running calibration.	MPDGCTRL0 PHY0	0x021b083c	0x434b0350
	MPDGCTRL1 PHY0	0x021b0840	0x034c0359
	MPDGCTRL0 PHY1	0x021b483c	0x434b0350
	MPDGCTRL1 PHY1	0x021b4840	0x03650348
	MPRDDLCTL PHY0	0x021b0848	0x4436383b
	MPRDDLCTL PHY1	0x021b4848	0x39393341
	MPWRDLCTL PHY0	0x021b0850	0x35373933
	MPWRDLCTL PHY1	0x021b4850	0x48254a36

- Not a JEDEC standard; controls i.MX internal DQS gate timing parameters
- Mechanism for our DRAM controllers to correctly sample incoming read DQS signal
- Relevant to DDR3 memories only (not supported with LPDDR2)
- Calibration code in MX6 and MX53 DRAM Stress Test

DRAM Calibration Conceptual Overview
Read DQS Delay Calibration (Continued)

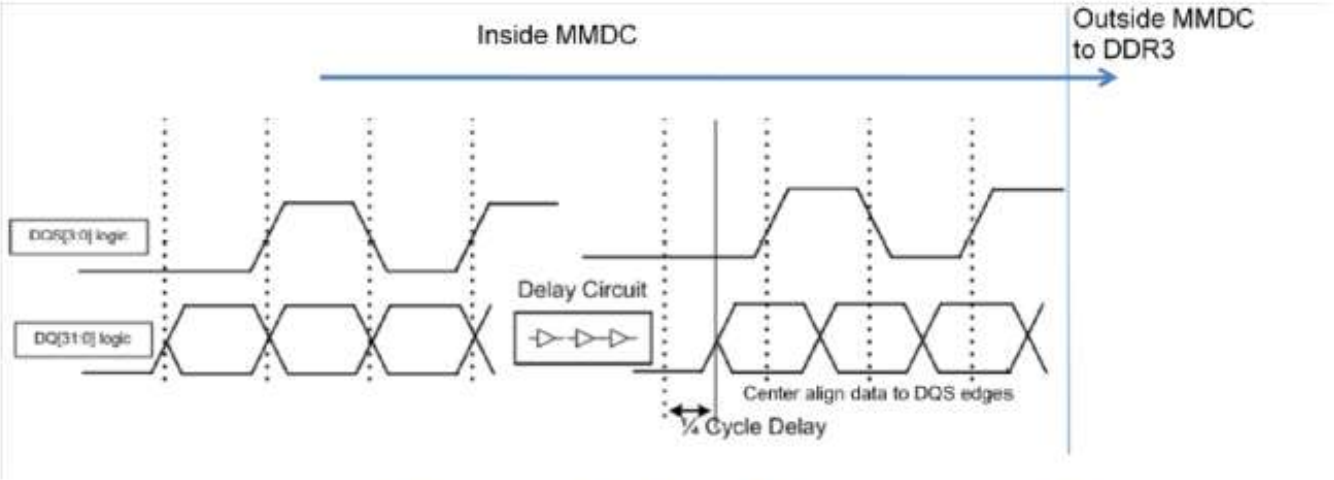


From the MX6DQ register programming aid example

These parameters are determined after running calibration. The parameters provided here are from Freescale's development board and will work as initial values. Update these values after running calibration.	MPDGCTRL0 PHY0	0x021b083c	0x434b0350
	MPDGCTRL1 PHY0	0x021b0840	0x034c0359
	MPDGCTRL0 PHY1	0x021b483c	0x434b0350
	MPDGCTRL1 PHY1	0x021b4840	0x03650348
	MPRDDLCTL PHY0	0x021b0848	0x4436383b
	MPRDDLCTL PHY1	0x021b4848	0x39393341
	MPWRDLCTL PHY0	0x021b0850	0x35373933
	MPWRDLCTL PHY1	0x021b4850	0x48254a36

- Used to adjust read-DQS within read-data byte
- Relevant to DDR3 and LPDDR2 memories
- Calibration code in MX6 and MX53 DRAM Stress Test

DRAM Calibration Conceptual Overview
Write DQS Delay Calibration

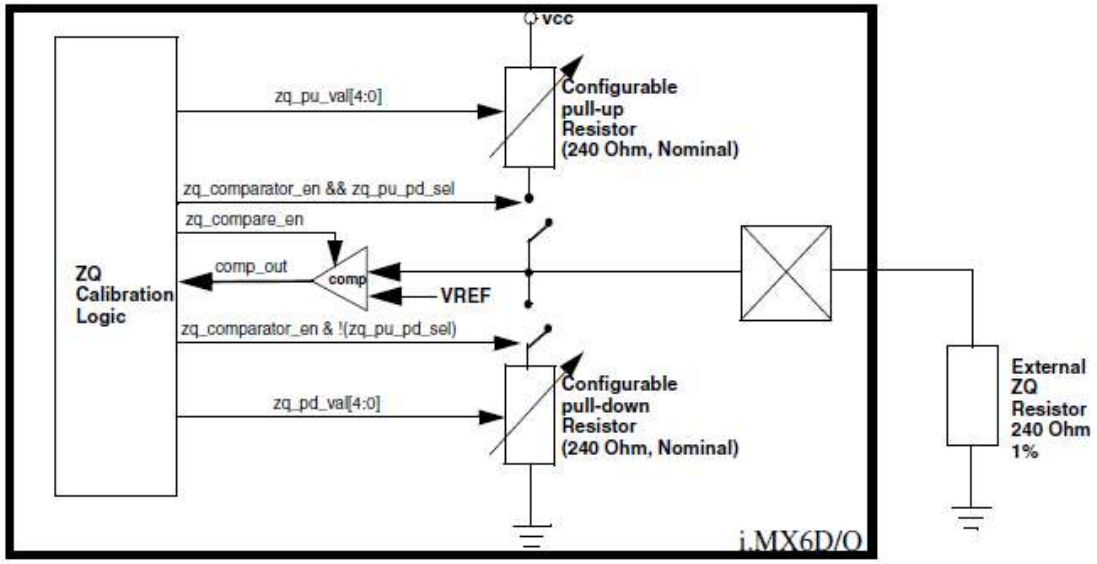


From the MX6DQ register programming aid example

These parameters are determined after running calibration. The parameters provided here are from Freescale's development board and will work as initial values. Update these values after running calibration.	MPDGCTRL0 PHY0	0x021b083c	0x434b0350
	MPDGCTRL1 PHY0	0x021b0840	0x034c0359
	MPDGCTRL0 PHY1	0x021b483c	0x434b0350
	MPDGCTRL1 PHY1	0x021b4840	0x03650348
	MPRDDLCTL PHY0	0x021b0848	0x4436383b
	MPRDDLCTL PHY1	0x021b4848	0x39393341
	MPWRDLCTL PHY0	0x021b0850	0x35373933
	MPWRDLCTL PHY1	0x021b4850	0x48254a36

- Used to center output write-DQS within write-data byte
- Relevant to DDR3 and LPDDR2 memories
- Calibration code in MX6 and MX53 DRAM Stress Test

DRAM Calibration Conceptual Overview
ZQ Calibration



- Feature of both i.MX* and DRAM (DDR3 and LPDDR2)
- Used to calibrate the pull-up/pull-down resistors of DRAM IO pads (tighter control of pad impedance)
- ZQ calibration occurs automatically, simply just need to enable it
  - Except on MX508, there's a SW routine to do this (in stress test)

* Only i.MX parts that support DDR3 and/or LPDDR2: MX6, MX53, MX508

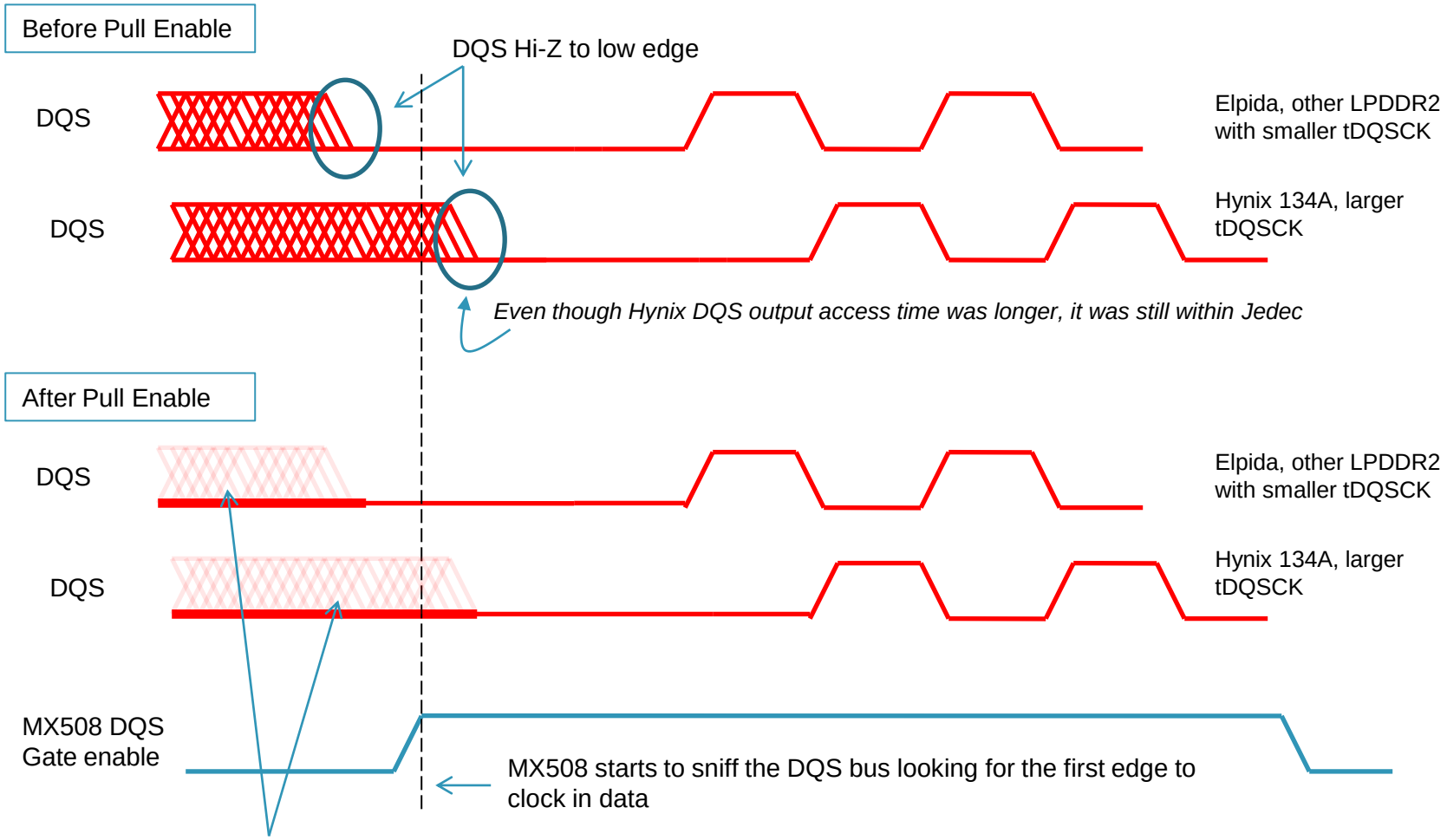
Agenda

- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

Case Study: MX508 and LPDDR2 Failure
Overview

- Major customer reported issues with MX508 and LPDDR2
  - Previous LPDDR2 vendor Elpida showed no problems
  - When switching to Hynix, customer reported Linux boot failures at 266MHz
- Failure could only be re-produced by booting Linux (it would crash)
  - Booting uboot showed no problems
  - When Hynix LPDDR2 placed on FSL MX508 EVK, could re-pro failure
- DRAM Stress Test did not originally catch this failure
  - Eventually new test was written that could catch this (row hop read)
- Problem further complicated due to lack of knowledge/experience with Denali DRAM controller
  - Denali support contact had ended, but FSL was able to extend
- Approx 6 weeks of debug before resolution
  - Tried multitude of internal Denali controller-to-phy timing variations
  - Ultimately, internal DQS gate timing parameters led to resolution – enable pull down on DQS

Case Study: MX508 and LPDDR2 Failure
Illustration of Fix



With pull enable, during Hi-Z when DQS bus is floating, pull down keeps DQS at known state, avoiding Hi-Z to low edges (same concept applies to DQS_b, but with pull up)

Agenda

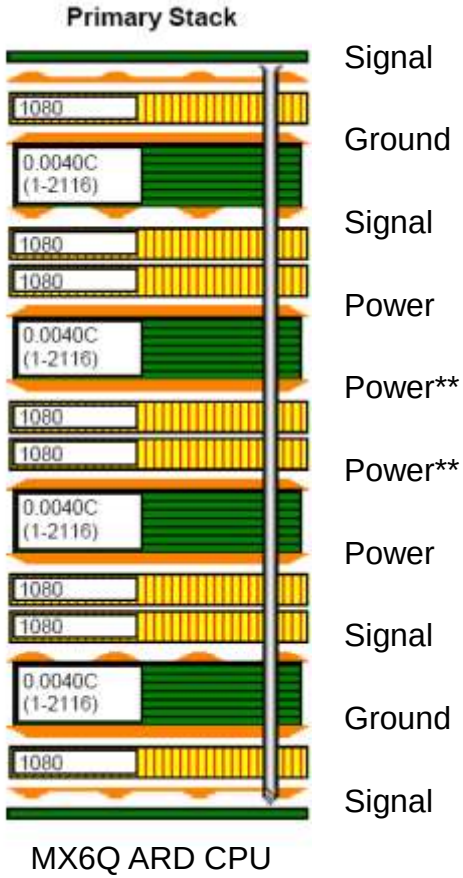
- Board bring-up: where DRAM bring-up fits in
  - Introduce the tools used for DRAM bring-up
- DRAM Register Programming Aid
  - Introduction/Overview
  - Walkthrough
- DRAM Stress Test
  - Introduction/Overview
  - How to build and run; deep dive into sub-tests
- DRAM Calibration Overview
- Case Study: MX508 and LPDDR2 Failure
- Board Design Considerations

8-Layer Board Stack-up

- Board stack-up critical for high-speed signal quality.
- Impedances must be pre-planned.
- High-speed signals must have a reference plane on an adjacent layer to minimize cross-talk.
- FSL Reference design = Isola 370HR

- Power** - additional power plane to support MX6Q to MX6Solo power options only.

	Impedance Type	Layer	Design	Actual	Pitch	Plane	Target	Tol (ohms)
1	Surface MS	L1	0.00470	0.0045	-	-	50	5.0
		-	-	-	-	L2		
2	EC Microstrip	L1	0.00370	0.0038	0.0090	-	90	9.0
		-	0.00370	0.0038	-	L2		
3	EC Microstrip	L1	0.00450	0.00325	0.0100	-	100	10.0
		-	0.00450	0.00325	-	L2		



MX6Q ARD CPU

DDR Routing 1

- Swapping DDR3 Data lines within bytes facilitates routing
  - **Write Leveling –** lowest order bit within byte lane must remain on lowest order bit of byte lane
  - For example D0, D8, D16, … fixed, other data lines free to swap within byte lane
    - JEDEC DDR3 memory restriction.
- No restrictions for complete byte lane swapping
  - DQS and DQM must follow lanes

DDR Routing 2

- Data re-assignment facilitates routing
  - **Data re-assigned within byte group**
  - **Byte Groups can be reassigned**

i.MX Contact	Memory Contact
DRAM_D0	DQ8
DRAM_D1	DQ15
DRAM_D2	DQ10
etc	etc
DRAM_D7	DQ9
SDQS0	DQS1
DQM0	DM1

← Lowest order bit

DDR Routing 3

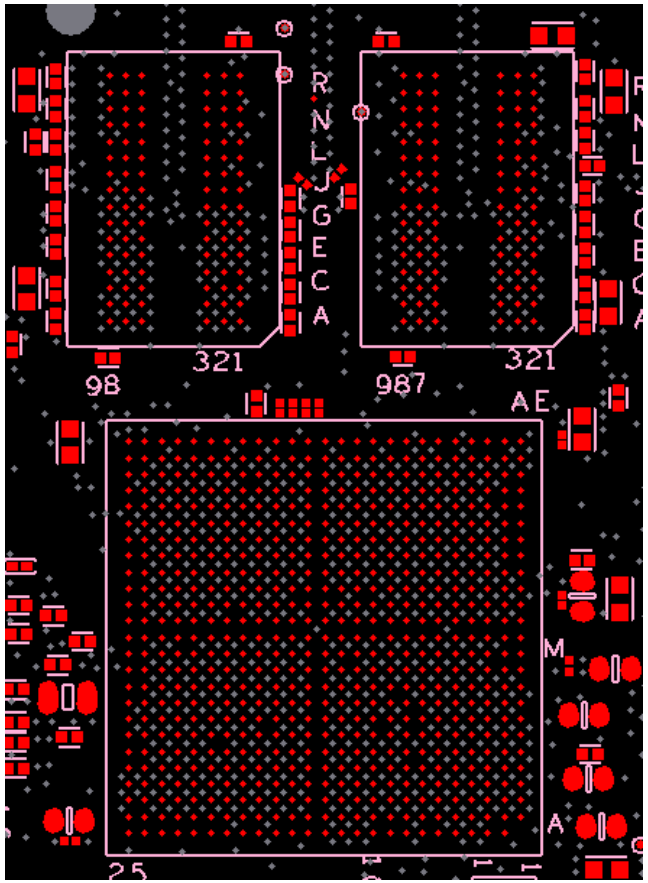
DDR (64-bit) routing configurations can be routed as:

- *"T" configurations*
  - Termination resistors not required
    - Accomplished with short routing lengths and on-chip drive strength control
  - Design limited to one chip select (4 x16 DDR's)
    - DDR3, 2 GBytes using latest memories (4 GBytes coming)
- *"Fly-by" configuration*
  - MX6 DDR controller provides Address mirroring when using 2 chip selects. Aids routing for memories on both sides of board.
  - Bus termination resistors required
  - Proven design method, easy to simulate

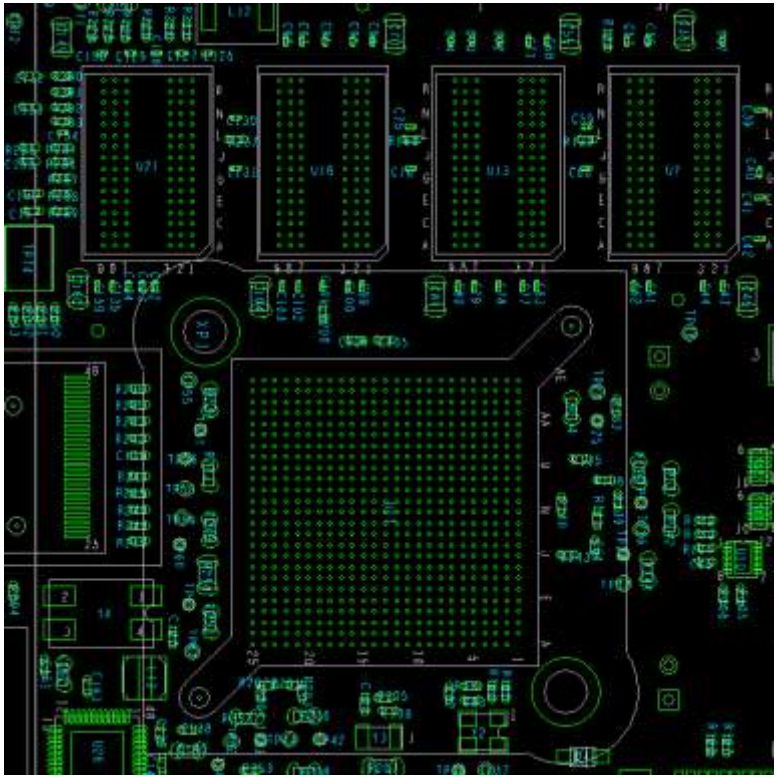
DDR Routing 4

- Prototype boards should plan for DDR signal breakout boards from Agilent or other.
  - Allows probing signal quality
  - Alternate – Remove one memory to probe bus

MX6Q DDR3 "T" Configuration



MX6Q DDR3 "Fly-by" Configuration



Fly-By Topology vs. T Route

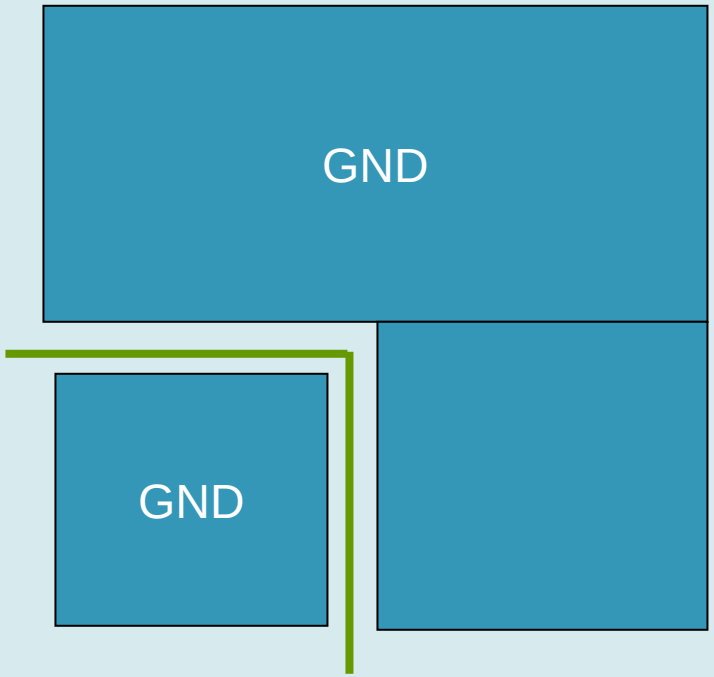
- Fly-By Topology Advantages:
  - Easier to route
  - Less chance for reflection in Address and Command traces.
  - Parallel termination resistors go at end of traces.
- T Route Advantages:
  - Less power consumption
    - Traces not pulled up to VREF.
  - Better performance.
    - Don’t lose extra clock cycle to reads and writes.

Layout considerations for high-speed signals

- High-speed signals must not cross reference plane gaps
- Avoid creating slots, voids, and splits in reference planes. Review via voids to ensure they do not create splits (space out vias).
- Clocks or Strobes on same layer need at least 2.5x spacing from adjacent trace (2.5x height from reference plane) to control cross-talk.
- All Synchronous modules should have bus length matching and relative clock length control.
  - CLK should be longer than the longest signal in the Data/Addr/Control group (+5 mils)
  - Many web resources

Power Grid Pontential Errors

- Chopping up planes reduces eff

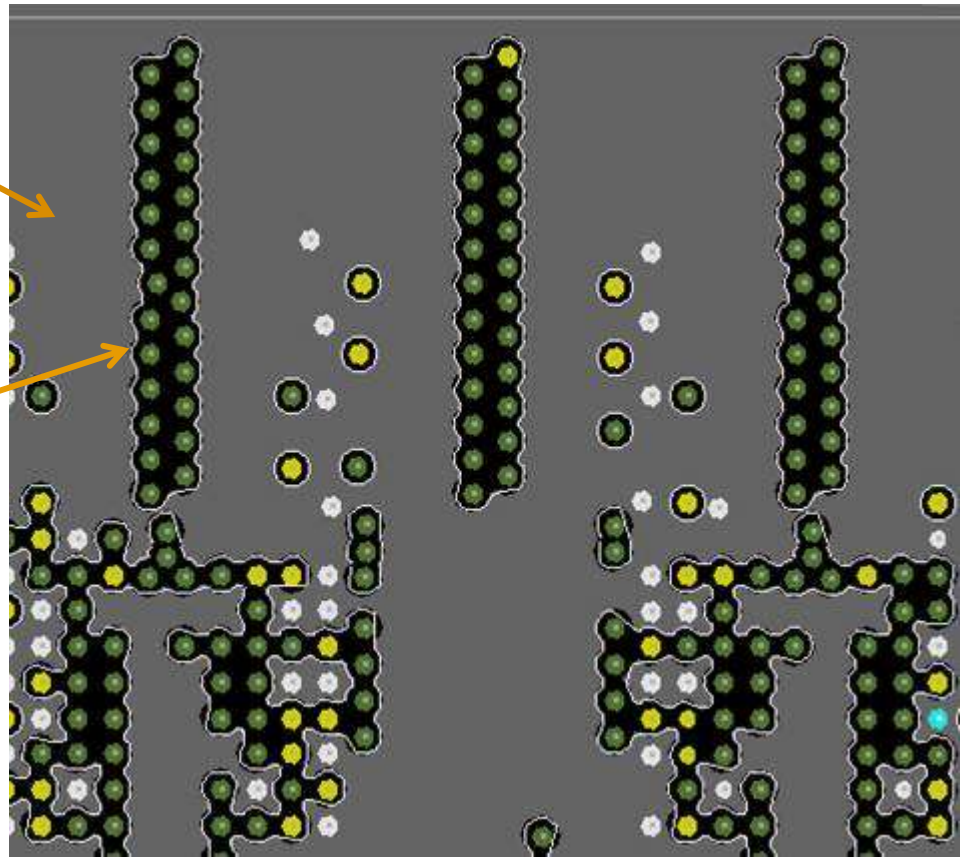


Poor GND plane –
chopped up to route green signal

DRAM GND Plane – Poor Layout

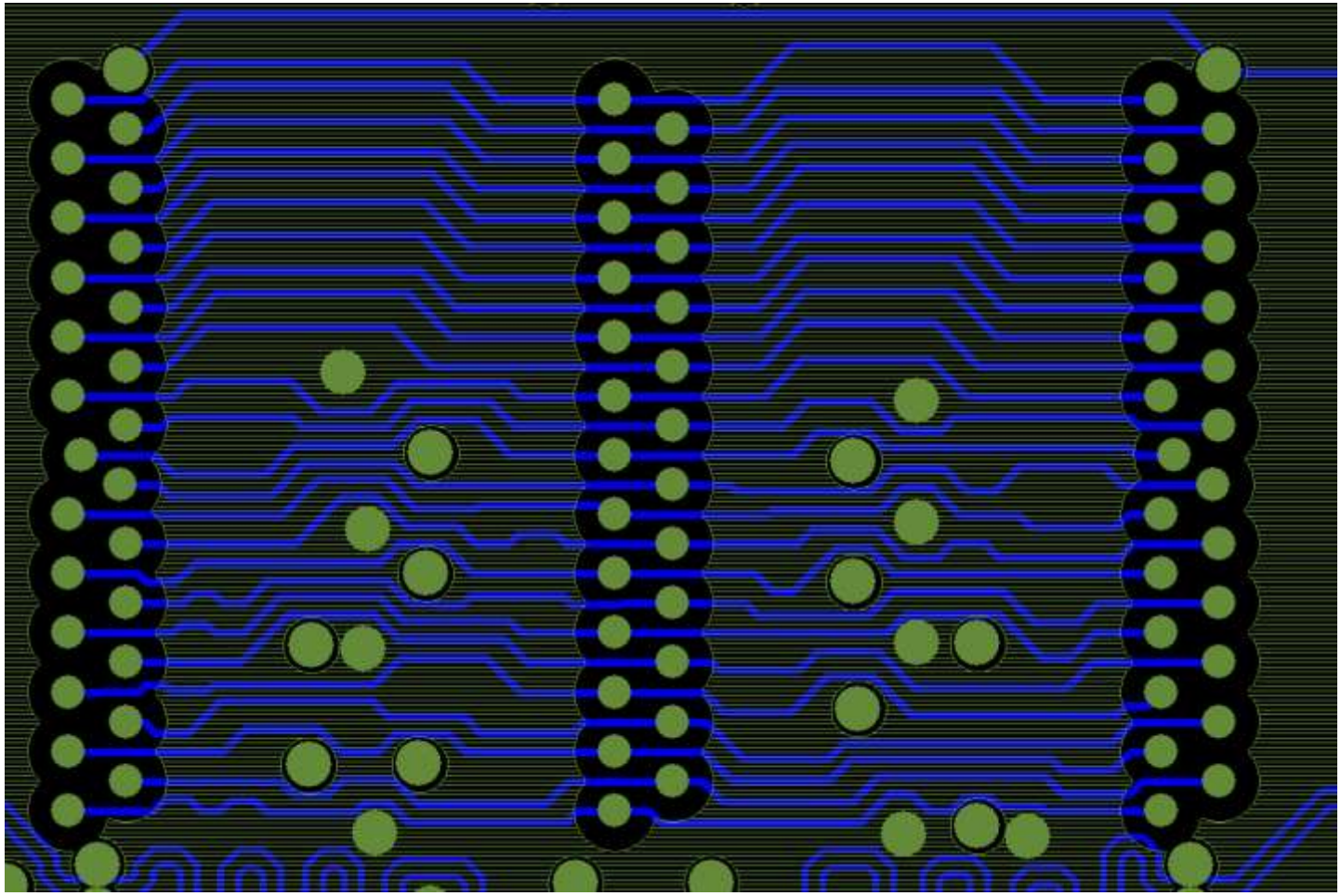
Gray = copper

Black = no copper



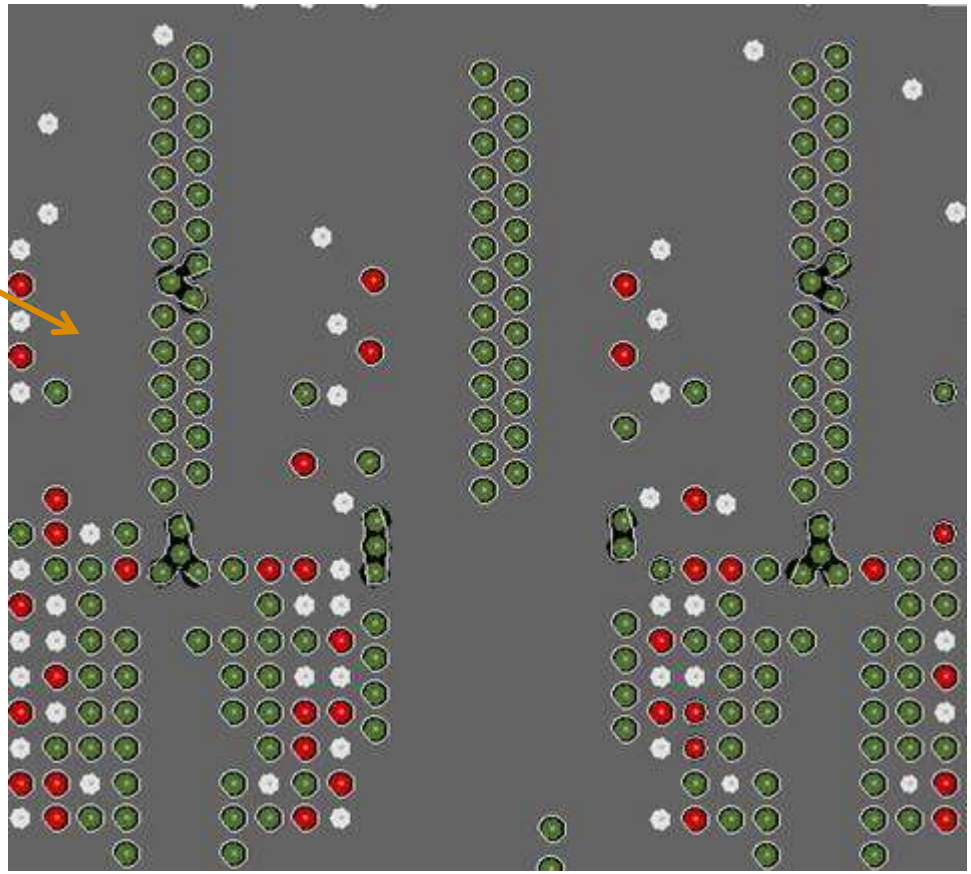
GND Plane of Previous Slide – Poor Layout Detail

- Vias too close together
- Horizontal current flow blockage



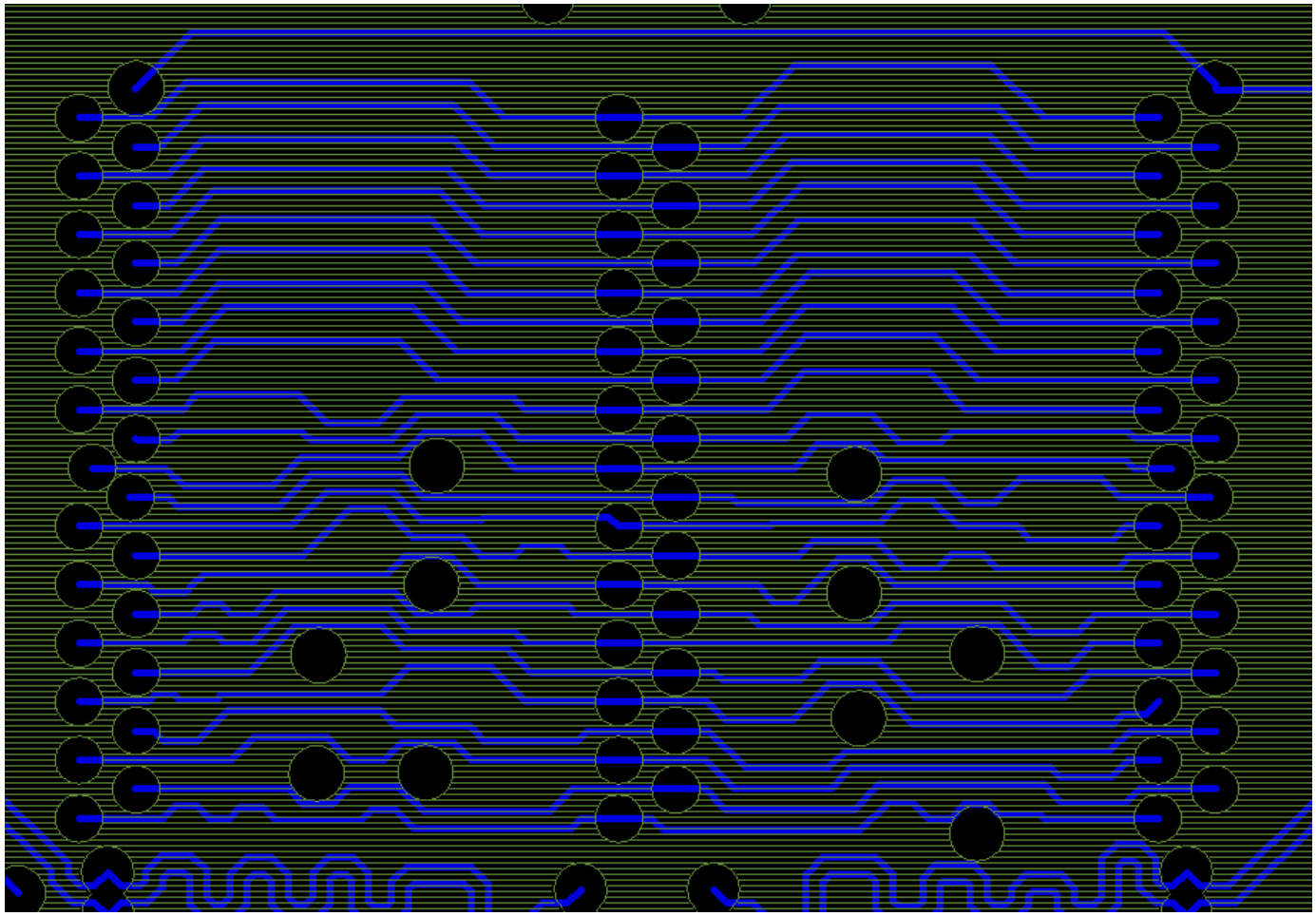
GND Plane - Good Layout

Gray = copper



GND Plane of Previous Slide – Good Layout Detail

- Vias spaced apart
- Facilitates horizontal current flow



High-Speed Signal Impedance

Signal Group	Impedance	Layout Tolerance (+/-)
All signals, unless specified	50 Ohm SE	2%
USB Diff signals	90 Ohm Diff	2%
Diff signals: LVDS, SATA, HDMI, DDR, PCIE, MIPI (CSI & DSI), MLB, Phy IC to Ethernet Connector	100 Ohm Diff (85 ohm PCI)	2%

Bring-Up: Power – Example Voltage Report

Board Name: _____ Serial #: _____ Data Collected by: _____ Date:

Source	Net Name	Expected (V)	Measured (V)	Measured Point	Comment
Main	5V0	5.0	5.103	C5.1	
3.3 V discrete reg	3V3_DELAYED	3.35	3.334	SH1	Requires LDO3 to enable
PMIC Switcher 1	VDDARM	1.375	1.377	SH2	
PMIC Switcher 2	VDDSOC	1.375	1.376	SH3	
PMIC Switcher 3	1V5_DDR	1.5	1.501	SH4	
PMIC LDO1	1V8	1.8	1.802	TP9	
PMIC LDO2	2V5	2.5	0.3	TP5	
VREFDDR	0V75_REFDDR	0.75	0.751	C8.1	50% of 1V5_DDR
Coin Cell	3V0_STBY	3.0	3.006	TP1	
MX6	VDDARM_CAP	1.1	1.114	C6.1	
MX6	VDDHIGH_CAP	2.5	2.515	SH5	
MX6	VDDSNVS_CAP	1.0	1.016	TP2	

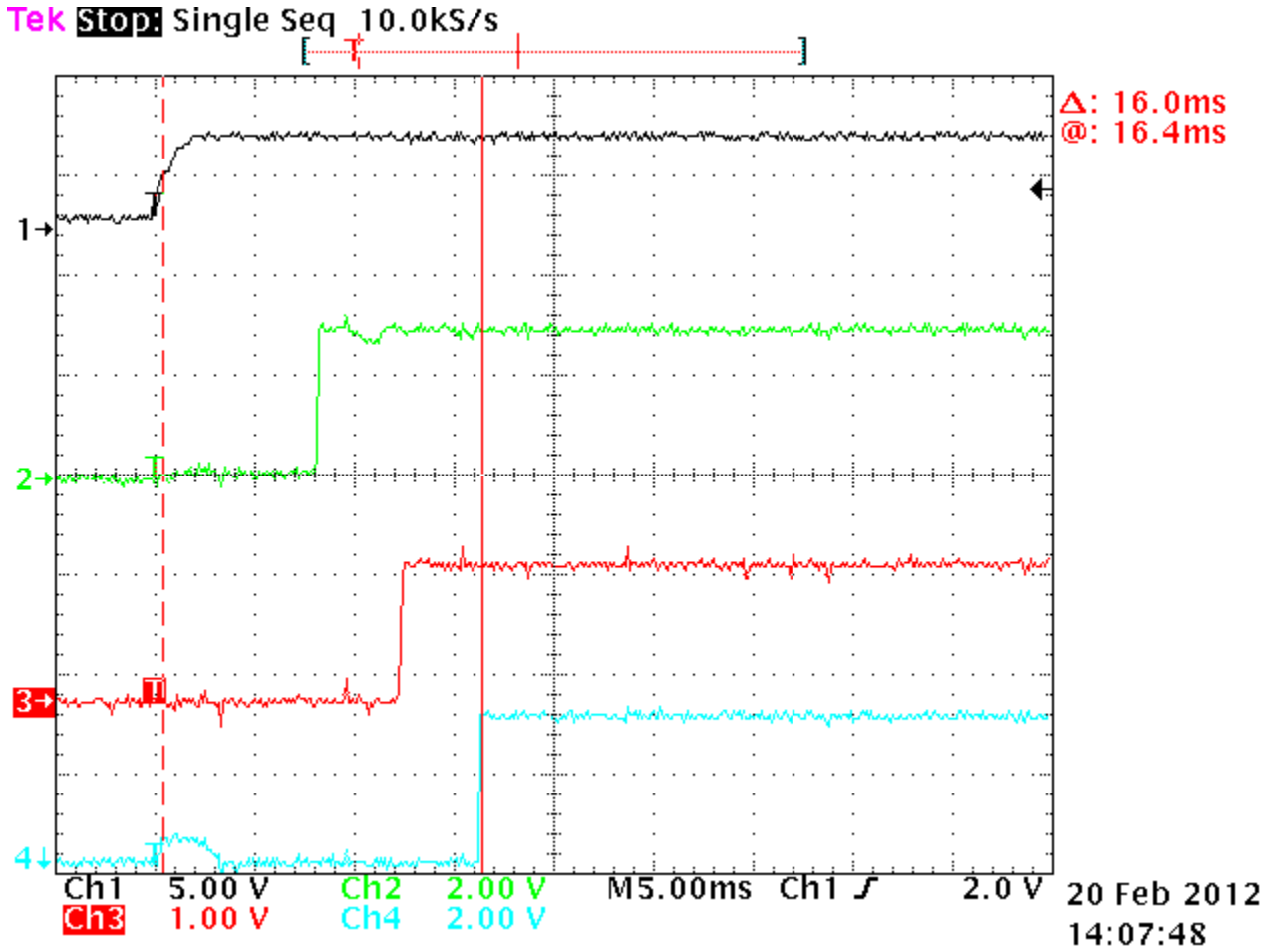
Bring-Up: Power-Up Sequence

5V_MAIN
Feeds 3.3 V Reg

SNVS

SW1A/B

POR_B
RESETBMCU



Summary

- Overview of tools used by the factory to optimize and debug DRAM interface
  - Excel spread sheet based register programming aid
  - DRAM stress test using open source compiler
- DRAM Calibration app note available from Freescale
  - Introduction of DRAM calibration concepts
- Case study of DRAM debug efforts

Links to Useful DRAM Documents

- JEDEC

DDR3 Specification:

http://www.jedec.org/sites/default/files/docs/JESD79-3E.pdf

DDR3L Amendment:

http://www.jedec.org/sites/default/files/docs/JESD79-3-1_1.pdf

LPDDR3 Specification:

http://www.jedec.org/sites/default/files/docs/JESD209-3.pdf

WideIO SDR Specification:

http://www.jedec.org/sites/default/files/docs/JESD229.pdf

LPDDR2 Specification:

http://www.jedec.org/sites/default/files/docs/JESD209-2E.pdf

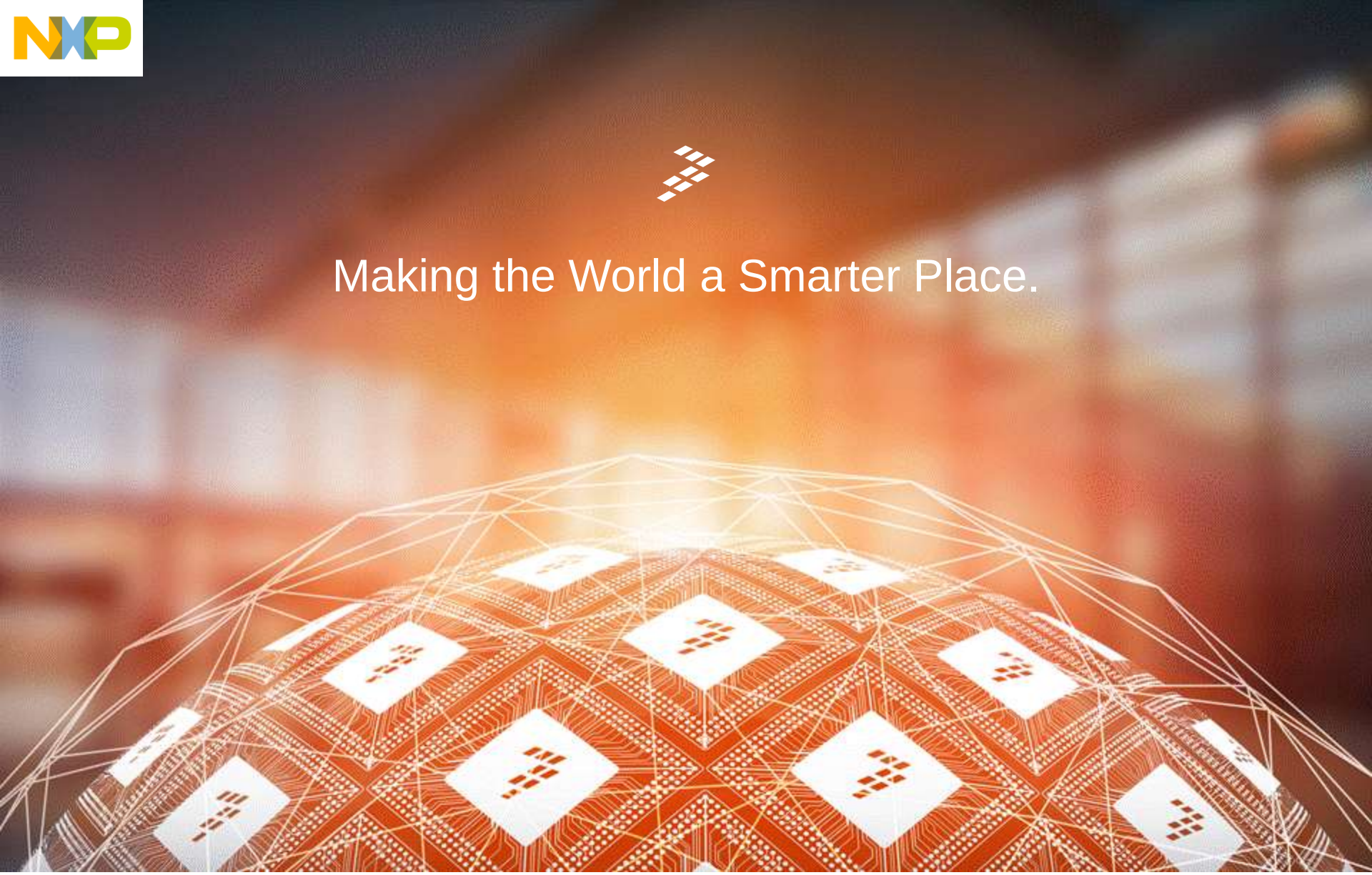
- Micron Documentation

DRAM Support Site:

http://www.micron.com/products/dram/ddr3-sdram#documentation_support

References

- FTF Session **FTF-ENT-F0039** *Designing Transmission Lines in High-Speed Printed Circuit Boards: Preventing Potential Problems*
  - Go to freescale.com → Freescale Technology Forum, Training, tools, … → FTF Americas → Technical Sessions Library → FTF-ENT-F0039
- *Books recommended from the session:*
  - *Right the First Time: A Practical Handbook on High Speed PCB and System Design, Volumes I & II, Lee W. Ritchey. Speeding Edge, ISBN* 0-9741936-0-7
  - *Signal and Power Integrity Simplified (2nd Edition),Eric Bogatin.* Prentice Hall, ISBN 0-13-703502-0
  - *High Speed Digital Design: A Handbook of Black Magic, Howard W.* Johnson & Martin Graham. Prentice Hall, ISBN 0-13-395724-1


Making the World a Smarter Place.

www.Freescale.com

www.Freescale.com