DDR subsystem: Enhancing System Reliability and Yield
Agenda

- Evolution of DDR SDRAM standards
- What is the variation problem?
- How DRAM standards tackle system variability
- What problems have been adequately addressed and what haven’t?
- Uniquify’s approach to tackling the variability problem.
- What does this mean for system reliability
Evolution of JEDEC Standards

2000
- DDR1
  - Source synchronous clocking (DQS) introduced
  - Data rate doubled compared to SDRAM by transferring data on both edges of DQS
  - 200MHz top speed
  - 256Mb – 512Mb
  - 2 or 4 bank devices

2003
- DDR2
  - Differential DQS (for signal integrity)
  - On-die termination
  - 533MHz top speed
  - 512Mb – 2Gb
  - 4 or 8 bank devices

2007
- DDR3
  - Write leveling
  - Bit leveling
  - Fly-by routing topology on the board
  - 1066MHz top speed
  - 1Gb – 4Gb
  - 8 bank standard

2012
- DDR4
  - More data training patterns
  - CRC
  - DBI
  - 1600MHz top speed

LPDDR1
- Similar to DDR1 without DLL
- Relaxed DQS vs clock timing on read; round-trip timing more difficult to predict
- Per-bank self-refresh
- 200MHz top speed

LPDDR2
- Many power-saving enhancements
- Low power modes
- Multiple refresh schemes including temperature controlled refresh
- 400MHz top speed
The Variation Problem

- DDRx Controller / PHY / I/O
- Output loading uncertainty
- Board trace uncertainty
- DDR memory timing uncertainty
- SoC Package
- Die
- PCB
- DDRx SDRAM

Uniquify, Inc.
How can we address variation?

OPTION 1

- Recover the clock from the incoming data stream

- But what are the disadvantages?
  - Need to encode data
  - Training requirements to detect data eye – IO becomes large and complex
  - Overhead for packetizing data and synchronizing data from parallel bit streams.
  - Added latency
  - IO size and complexity, overhead and latency makes it ill-suited for main memory application.
How can we address variation?

OPTION 2

- Use clock forwarding: Send the clock to be used to capture data with the data.

- Source synchronous clocking is the strategy used by all generations of DDR SDRAM.

- Very effective in addressing variation – all path delays on data are also seen on clock (called data strobe).

- If the system design is done carefully by routing data and associated clock together, we can ensure all variations on the data path are also seen on the strobe path.

- Both static (process variation, package and board configuration induced) and dynamic variations (caused by system operating voltage and temperature fluctuations) can be minimized by design in this way.
## Timing Margins are Shrinking

<table>
<thead>
<tr>
<th>DDR Type</th>
<th>Clock (MHz) / Data Speed (Mbps)</th>
<th>Timing Window (ps)</th>
<th>DRAM Margin (ps)</th>
<th>Package / Board Margin (ps)</th>
<th>On-Chip Margin (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR1</td>
<td>200 / 400</td>
<td>2,500</td>
<td>900</td>
<td>800</td>
<td>800</td>
</tr>
<tr>
<td>DDR2</td>
<td>533 / 1066</td>
<td>938</td>
<td>425</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>DDR3</td>
<td>1066 / 2133</td>
<td>469</td>
<td>188</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>DDR4</td>
<td>1600 / 3200</td>
<td>312</td>
<td>125</td>
<td>93</td>
<td>93</td>
</tr>
</tbody>
</table>
Uniquify’s approach to meet the variation challenge

- Ensure that all the data and associated strobe paths in the SoC are carefully **balanced** so they track each other effectively as voltage and temperature variations occur over time.

- Board and package design should also support the effort by routing data and strobe **as a group** to carefully match delays.

- For in-system training, DDR3 and DDR4 provide pre-defined registers that can be used to read out fixed training patterns.

- These can be used to further train the data eye in-system and obtain additional margin.
Uniquify’s approach to meet the variation challenge

- Read side training values can also be applied on the write side to avoid the need for separate write training.

- This is possible since data and strobe are bi-directional signals on the package and board and any skews seen on the read path will also be seen on the write path.

- But this does require that IC design is done such that it does not introduce different skews on write and read paths.
The Clock domain crossing problem

- Even though we capture the data successfully using the source forwarded clock, the data still needs to be transferred to the destination clock domain.

- This is far from simple, and has always been in our experience the most under-estimated, yet most deadly of all problems.

- Each generation of DDR standards tries to find new solutions, yet many people found that their problems seemed to stay one step ahead!
The write side is relatively simple.

All generation of DDR standards force the SoC to observe a timing relationship between strobe and DRAM clock.

This allows the DRAM to transfer data from its strobe clock domain to its system clock domain fairly easily.
For DDR2, SoC and board designers had no choice but to match clock path and strobe path delays inside the SoC and on the board.

On the SoC, this had to be done by matching insertion delay on the clock output with insertion delay on the DQ & DQS output of every byte lane – not trivial considering that the DDR interface might span a very wide area within the chip.

On the board, clock traces to DRAMs for all byte lanes were matched using the “star topology”. But this placed limitations on board signal quality.
Starting with DDR3, SoC and board designers received a big break with the introduction of write leveling.

Write leveling provides a mechanism to do automatic training in-system to satisfy strobe and clock relationship timing.

So clock trace delays did not need to match across byte lanes.

So SoC designers no longer had to match the clock insertion delay to match DQS output timing on every single byte lane in the SoC.
- And board designers could use a routing architecture for the clock that was more optimized for signal quality, such as the daisy chain topology.
Basic concept behind write leveling: Schmoo the write DQS and let the DRAM feedback information on when rising edge of DQS matches with rising edge of DRAM clock.

- Write leveling works without any problems if prior to write leveling we can guarantee these 2 conditions:
  - There is < 1 cycle of skew between write DQS and DRAM clock at every DRAM’s pins &
  - It can be guaranteed that write DQS delay is smaller than DRAM clock delay.

- This restriction is not a problem for embedded systems that usually have limited number of DRAM ranks and byte lanes.
But what happens if there is more than 1 cycle of skew between write DQS and DRAM clock?

In this case the write DQS could get aligned with the wrong DRAM clock edge.

This could potentially happen for high speed multi-rank DIMM configurations with wide data buses.

DRAM protocol does not specify any means to automatically correct this.
To validate if write is working correctly, we need to be able to read and verify the written data.

But this requires that reads should work correctly!

So this is the time to segue into the last and most difficult aspect of DDR interface timing – the read side clock domain crossing problem.

How do we transfer the data captured using the DRAM forwarded read clock to the internal system clock?
Unlike the write side where the DRAM has a pre-defined specification to solve the clock domain crossing problem, there is no such specification to help us on the SoC side.

First question we should ask is then – what is the expected relationship between these 2 clocks?

The DRAM memories do provide a timing specification on the relationship between the DRAM receive clock and the read strobe forwarded clock.

However these timing specifications vary between different flavors of DRAM.

This specification is particularly loose in the case of LPDDR1 and LPDDR2.
So the relationship between these 2 clocks on the SoC side is dependent on the total path delays in the system – from the SoC system clock tree to the DRAM clock pin, the DRAM itself and then back from the DRAM read strobe pin to the SoC.

So this is commonly called the “round trip timing” problem.
The most common solution to the round trip time problem is to use an asynchronous FIFO.

The write pointer is incremented by Read data strobe received from DRAM.

The read pointer is incremented by the system clock whenever it does not match the write pointer.
The strobe signal is bi-directional since it is used during both reads and writes.

So, before the read starts, it will be un-driven and perhaps noisy. So it needs to be gated (blocked) when it is not driving a valid signal and un-gated when it is driving a valid signal.

Without the correct gate timing, the FIFO might clock in invalid data (if too early) or NOT clock in valid data (if too late)

So, the round trip time problem has not really gone away. It’s come back in a different form – that of finding the correct “gate timing”
The preamble time is specified as 90% of the clock period in the JEDEC DRAM specifications. This translates to:

<table>
<thead>
<tr>
<th>Speed (Mbps)</th>
<th>Preamble Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 DDR1</td>
<td>4.5</td>
</tr>
<tr>
<td>1066 DDR2</td>
<td>1.7</td>
</tr>
<tr>
<td>2133 DDR3</td>
<td>844</td>
</tr>
<tr>
<td>3200 DDR4</td>
<td>560</td>
</tr>
</tbody>
</table>

**Pictorially:**

![Diagram showing preamble time](image-url)
But dynamic voltage and temperature dependent variations are getting worse as we go to smaller and smaller process nodes.

The combination of increased variation with smaller timing margins is a double whammy for system reliability!

<table>
<thead>
<tr>
<th>Data Rate</th>
<th>Shortest Round trip</th>
<th>Longest Round trip</th>
<th>DQS gating window</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 Mbps DDR1 @ 130nm node</td>
<td>3.5ns</td>
<td>6ns</td>
<td>4.5ns</td>
</tr>
<tr>
<td>1066 Mbps DDR2 @ 90nm node</td>
<td>2ns</td>
<td>4ns</td>
<td>1.7ns</td>
</tr>
<tr>
<td>1600 Mbps DDR3 @ 65nm node</td>
<td>1ns</td>
<td>2.5ns</td>
<td>1.12ns</td>
</tr>
<tr>
<td>2133 Mbps DDR3 @ 40nm node</td>
<td>0.8ns</td>
<td>2ns</td>
<td>844ps</td>
</tr>
<tr>
<td>3200 Mbps DDR4 @ 28nm node</td>
<td>0.6ns</td>
<td>1.5ns</td>
<td>560ps</td>
</tr>
</tbody>
</table>
For DDR1 and low speed DDR2, a simple fixed register setting could be used to fix the gate timing.

As DDR2 clock speeds reached 400Mhz and 533Mhz, gate timing caused lots of problems needing tweaking and adjustment of timing registers by system engineers.

With DDR3 800Mhz being mainstream now, many system engineers are struggling with system instability.
System instability and failure!

- At some point, because of the random voltage and temperature variations, the static parameters no longer hold good!
And even if the system engineers succeed in getting reliable operation on the bench, getting to production is an even bigger challenge.

A robust ATE test methodology to ensure DDR sub-system reliability is needed so that test escapes do not translate to system yield loss!
There have been various attempts by the DRAM community to address the round trip timing problem (such as DFI 2.1 gate training), but Uniquify came up with a different approach.

We found a way to solve the round trip time problem in its entirety! So it makes no difference if the round trip time is $\frac{1}{2}$ the cycle period or 4 times the cycle period!

Our solution: A mechanism to exactly measure the round trip time in-system on power up!

We call this patented technology SCL (for Self Calibrating Logic).
What does SCL do?

- SCL provides exact information regarding:
  
  (A) The phase relationship between incoming read strobe and system clock. 
  (B) Read strobe latency in system clock cycles.

- SCL programs the registers that control these settings automatically with no user intervention.

- As a result, the DDR subsystem is truly and completely “Self Calibrating”.

SCL is capable of measuring and compensating many cycles of round trip time variation with a fine precision. It has more than enough range for even the smallest process nodes and highest clock speeds.

This entire range of variation can be covered with a fine granularity – about 40ps in TSMC 40G node.
Further SCL guarantees the BEST possible system read latency.

It ensures that the read data is delivered to the system at the earliest possible cycle after the read strobe has been received on all byte lanes.

Faster than the FIFO based approach since it eliminates pointer synchronization latencies!
Every byte lane is tuned by SCL independent of others, so SCL accommodates a wide range of variation in round trip time across byte lanes.

No need for SoC designers to worry about matching data strobe output delay across multiple byte lanes.

No need for system designers to worry about delay matching across different byte lanes.

SCL is even more powerful in combination with write leveling, since reads can be used to check writes!

If SCL fails after write leveling, re-run write leveling to align the write strobe to the next DRAM clock edge and try again!
SCL sets up the timing parameters to the best values at power on:

- **Static Parameters**
- **SCL Calibration**

**System Failure**
Over the last 3 years, Uniquify has further enhanced SCL, so that SCL is now also optimized to be run dynamically in the middle of system operation.

Dynamic SCL (DSCL) needs to be very optimal in minimizing impact on system performance by reducing the time interval of interruption.

DSCL can be run in just 300ns at DDR3 2133Mbps. This is comparable to a refresh interval for higher density DDR3 and DDR4 memories.

Needs to be run once a second or at most a few times a second to optimize the read timing for temperature driven variations.

Long term voltage variations are usually also caused by temperature since temperature affects the leakage current. So DSCL compensates for this effect as well.
DSCL delivers the highest system reliability

Data Window

Static Parameters

SCL Calibration

DSCL Calibration

System Failure
We have seen by real world experience and customer feedback, that DSCL is a quantum leap forward in providing excellent system reliability.

- It’s a comprehensive solution that addresses all possible sources of DDR interface integrity problems on both writes and reads.

- It addresses the unknown static variations due to the SoC, package, board, manufacturer of the DRAM or the type of DRAM AND the dynamic variations in system operating conditions.
SCL / DSCL additional benefits

- Simplified ATE testing.

- More and more SoC manufacturers are finding that the best way to test DDR memory subsystem on the ATE is to add real DRAM memory on the load board.

- Relays can be used to isolate the DRAM during connectivity test or VOH/VOL test.

- With DRAM on the load board, DDR sub-system testing with DSCL is as simple as performing a sequence of registers writes and a single register read-back to verify test pass or fail.

- Since DSCL does comprehensive interface training, if there is a solution, DSCL will find it and make the test pass!

- Ensures both high quality test coverage AND high yield
$60B leading edge consumer electronics company

HDTV application
- Thermal issues caused field reliability problems
- Design based on standard (non-Uniquify) PHY
- No SCL/DSCL

Latest generation of application
- Adopted Uniquify’s DDR memory subsystem with SCL/DSCL

Problem Solved!
- “…we were surprised to find that the DDR interface is now much more robust than it was in the past. SCL/DSCL solved the problem.”
Thank you