

Determining DLLRCVEREN Pulse Optimum Location

Application Note

February 2007



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

IMPORTANT - PLEASE READ BEFORE INSTALLING OR USING INTEL® PRE-RELEASE PRODUCTS.

Please review the terms at http://www.intel.com/netcomms/prerelease_terms.htm carefully before using any Intel® pre-release product, including any evaluation, development or reference hardware and/or software product (collectively, "Pre-Release Product"). By using the Pre-Release Product, you indicate your acceptance of these terms, which constitute the agreement (the "Agreement") between you and Intel Corporation ("Intel"). In the event that you do not agree with any of these terms and conditions, do not use or install the Pre-Release Product and promptly return it unused to Intel.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

The DLLRCVEREN may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Code names presented in this document are only for use by Intel to identify products, platforms, programs, services, etc. ("products") in development by Intel that have not been made commercially available to the public, i.e., announced, launched or shipped. They are never to be used as "commercial" names for products. Also, they are not intended to function as trademarks.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel Leap ahead, Intel Leap ahead logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

The ARM* and ARM Powered logo marks (the ARM marks) are trademarks of ARM, Ltd., and Intel uses these marks under license from ARM, Ltd.

Copyright © 2007, Intel Corporation. All Rights Reserved.



Contents

1.0 Introduction	5
2.0 DLLRCVREN Algorithm	6

Figures

1 DLLRCVREN Optimum Pulse Location	5
--	---

Tables

No Tables Used At This Time



Revision History

Date	Revision	Description
February 2007	001	Initial release.



1.0 Introduction

The pseudo-code listed in [Section 2.0, “DLLRCVREN Algorithm” on page 6](#), provides an algorithm to determine the optimum location for the DLLRCVEREN pulse with respect to the DQS signals. The rising and falling edges of the DLLRCVEREN pulse must be placed in the pre-amble and post-amble windows driven by the DDR2 memory as shown in [Figure 1](#). The DLLRCVEREN pulse has a fixed width. The entire pulse is shifted by controlling the DLLRCVEREN delay value. The basic goal of the algorithm is to locate the first rising edge of the DQS signal and use the result to calculate the optimum DLLRCVEREN delay value.

Note: The SDRAM initialization sequence must be executed before executing this algorithm. Refer to "DDR SDRAM Initialization".

The algorithm starts by taking an initial sample of the DQS signal value using a given DLLRCVEREN delay value. The DQS sample indicates the value of the DQS signal at the leading edge of the DLLRCVEREN pulse as shown in [Figure 1](#). Using the initial DQS value, the algorithm proceeds to find the rising edge of DQS. The DQS signal rising edge is searched by shifting the DLLRCVEREN pulse either left or right depending on the first sampled DQS value. For example, when the first sampled DQS value is '1', the DLLRCVEREN delay is decreased one delay element at a time until the DQS rising edge is located. And when the first sampled DQS value is '0', the DLLRCVEREN delay is increased one delay element at a time until the rising edge of DQS is located.

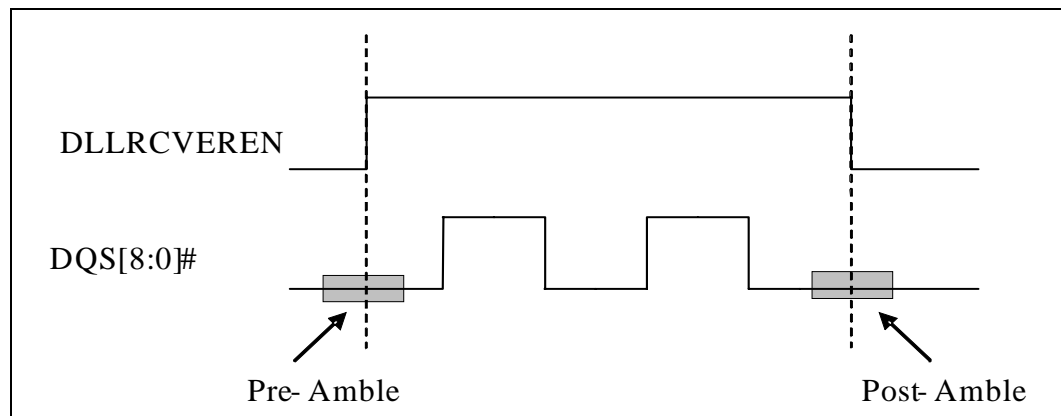
Once a DQS rising edge is located, the algorithm proceeds to verify that this is the first DQS rising edge. For example, the DQS rising edge found may be the second rising edge. When it is determined that it is indeed the second DQS rising edge, the DLLRCVEREN delay is adjusted by subtracting one full MCLK worth of delay from the current DLLRCVEREN delay value.

At this stage in the algorithm, with the current DLLRCVEREN delay value, the DLLRCVEREN leading edge must be aligned with the first rising edge of the DQS signal. To calculate the optimum DLLRCVEREN delay value, MCLK/4 worth of delay must be subtracted from the current DLLRCVEREN delay value.

After programming the correct DLLRCVEREN delay value, the FIFO must be reset, SDCR0[7].

Note: The DLLRCVER auto configuration logic does not work with non-ECC DIMMs. The logic samples DQS[8] which is the DQS for the ECC byte. On a non-ECC DIMM this DQS signal is not present, and therefore DQS will always be sampled as 0.

Figure 1. DLLRCVREN Optimum Pulse Location





2.0 DLLRCVREN Algorithm

```
// DLLRCVREN Algorithm Starts Here

compute_dllrcver()
{

/* Clear Auto Calibration Circuit Result bit */

// Read current DLLRCVER Register
DLLRCVER_REG = *dllrcver_reg_addr; // read register
    DLLRCVER_REG &= 0xFFFFBFFF; // Clear bit 18 to clear Result bit

// Write DLLRCVER Register
    *dllrcver_reg_addr = DLLRCVER_REG;

/* Set Initial Value of DLLRCVER Delay using a recommended initial value */
dllrcver_value = 0x50;

/* Take first sample of DQS with current DLLRCVER Delay */
    dllrcver_value |= 0x00060000; // Enable auto calibration circuit
dqs_value = sample_dqs(dllrcver_value);

/* Create an inverted current DQS value, and
    also set the direction to search for the DQS rising edge */
    if (dqs_value == 1)
    {
inverted_dqs_value = 0;
        direction_value = -1; // Go backward
    }
    else
    {
        inverted_dqs_value = 1;
direction_value = +1; // Go forward
    }
}
```



```

    }

/* Find DQS rising edge - infinite loop */
loop:
{
dllrcver_value = dllrcver_value + direction_value;

// Take a sample of DQS with current DLLRCVER Delay
    dllrcver_value |= 0x00060000; // Enable auto calibration circuit
dqs_value = sample_dqs(dllrcver_value);

if (dqs_value == inverted_dqs_value)
{
    // Found edge
    break;
}

// Exit if edge is not found - Error condition algorithm ends
if (dllrcver_value == 0 | dllrcver_value == 255)
{
    exit();
}
} // End infinite loop

/* Verify if the DQS edge found above is the second DQS edge, and
adjust DLLRCVER Value if it was indeed the second edge */

if (dllrcver_value >= 48)
{
    dllrcver_value |= 0x00060000; // Enable auto calibration circuit
dqs_value = sample_dqs(dllrcver_value - 48)
    if (dqs_value == 1) // second edge
    {
        // Shift back to first edge

```



```
        dllrcver_value = dllrcver_value - 64;
    }
}

/* Compute Optimum DLLRCVER Value by moving
   DLLRCVEREN delay back by MCLK/4 to be in the pre-ample window */
dllrcver_value = dllrcver_value - 16;

/* Write the optimum DLLRCVEREN delay and clear the Auto Calibration
   Circuit */

dllrcver_value |= 0x00040000; // Disable auto calibration circuit
sample_dqs(dllrcver_value);

/* Reset Read FIFO by Clearing bit 07 of SDCR0 */

} // End Algorithm

/* Sample DQS Procedure */

sample_dqs(dllrcver_value)
{
    /* Read current DLLRCVER Register */
    DLLRCVER_REG = *dllrcver_reg_addr; // read register
    DLLRCVER_REG &= 0x00010000; // Preserve bit 16

    /* Assemble bits to write DLLRCVER Register */
    TEMP_DLLRCVER_REG = dllrcver_value & 0xE0; // Bit[10:08] value
    TEMP_DLLRCVER_REG = TEMP_DLLRCVER_REG << 3;
    DLLRCVER_REG |= TEMP_DLLRCVER_REG; // Bit[10:08] set
    TEMP_DLLRCVER_REG = dllrcver_value & 0x1F; // Bit[04:00] value
    DLLRCVER_REG |= TEMP_DLLRCVER_REG; // Bit[04:00] set
    TEMP_DLLRCVER_REG = dllrcver_value & 0x00060000; // Bit[18:17] value
```



```
DLLRCVER_REG |= TEMP_DLLRCVER_REG; // Bit[18:17] set

/* Write DLLRCVER Register */
*dllrcver_reg_addr = DLLRCVER_REG;

/* Read register again to guarantee previous write operation */
TEMP_DLLRCVER_REG = *dllrcver_reg_addr;

// Need a fence instruction here to guarantee read occurred

/* Read a DDR SDRAM Memory Location,
   this memory read causes the auto calibration circuit to
   sample the DQS signal */
TEMP = *ddr_mem_addr;

/* Read DLLRCVER register bit 24 to get DQS sample */
DLLRCVER_REG = *dllrcver_reg_addr;
DLLRCVER_REG &= 0x01000000;
if (DLLRCVER_REG == 0)
{
    return(0); // DQS sampled as 0
}

return(1); DQS sampled as 1
} // End sample_dqs() procedure
```



§ End Of Chapter §