

**Please note that Cypress is an Infineon Technologies Company.**

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**

Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec No: 001-14796

Spec Title: USING THE COLLISION DETECTION WITH  
FULLFLEX(TM) SYNCHRONOUS DUAL  
PORT SRAMS - AN5025

Sunset Owner: Sheetal Chennapoor (SMCH)

Replaced By: 001-92810.

# Using the Collision Detection with FullFlex™ Synchronous Dual Port SRAMs

**AN5025**

**Associated Project:** No

**Associated Part Family:** FullFlex™ Synchronous Dual port SRAMs

**Associated Application Notes:** None

## Abstract

Collision detection in Cypress's FullFlex™ Synchronous dual port SRAMs is a technique to guarantee data while performing simultaneous accesses to memory in a synchronous dual port SRAM. The functionality of this collision detection circuitry and its advantages are discussed in this application note.

## Introduction

In synchronous dual ports, when both ports are accessing the same memory location, and at least one port is doing a write, there is a risk of data corruption. For instance, if one port reads from an address, the data could be corrupted if the other port writes to the same memory location simultaneously. Similarly, if both ports attempt to write to the same location, the data actually written cannot be guaranteed. Unlike asynchronous dual ports, which have arbitration circuitries, synchronous dual ports usually necessitate external logic to ensure that the data in the dual port isn't corrupted. To address this problem, Cypress's FullFlex™ Family of Synchronous dual ports contain collision detection circuitry. The collision detection circuitry is a method to guarantee data during a collision.

## Collision Detection Benefits

Historically, dual ports have not included the collision detection circuitry feature. Cypress FullFlex dual ports provide the collision detection circuitry feature, along with other new practical features, such as the Variable Impedance Matching (VIM) circuitry and the Echo Clock circuitry. With the addition of the collision detection circuitry, it is not necessary to implement external logic to perform the same functions as is the case with traditional dual ports. Thus, its implementation eliminates the need for external logic, hence simplifying the board design.

The collision detection circuitry asserts a busy signal if the data is corrupted as seen in Table 1. This results in better system reliability.

## Collision Detection Circuitry

Both ports can read from the same memory location at any time. However, if one or both of the ports write to the same location in the memory array, the circuitry will detect a collision.

It will send a flag to the port whose data could be corrupted. The flag is an active low signal sent to the BUSY pin of the port.

On read and write cycles, the device compares the address being driven on one port with the address being used by the other port. The circuitry will detect a collision if three conditions are met:

- The addresses are the same.
- The chip enables ( $\overline{CE0}$  and  $\overline{CE1}$ ) for both ports are asserted
- The rising edge of both clocks violate the clock-to-clock set-up time ( $t_{CCS}$ ).

Table 1 lists the collision detection circuitry outputs and descriptions.

**Table 1. Collision Detection Output**

| Port-L | Port-R | $\overline{BUSY-L}$ | $\overline{BUSY-R}$ | Description                  |
|--------|--------|---------------------|---------------------|------------------------------|
| R      | R      | H                   | H                   | No collision                 |
| W      | R      | H                   | H                   | Read Old Or New data         |
|        |        | H                   | L                   | Data not guaranteed          |
| R      | W      | H                   | H                   | Read Old Or New data         |
|        |        | L                   | H                   | Data not guaranteed          |
| W      | W      | L                   | L                   | Array data corrupted         |
|        |        | H                   | L                   | Array stores left port data  |
|        |        | L                   | H                   | Array stores right port data |

Unlike the arbitration circuitry for asynchronous dual ports, the collision detection circuitry does not block the write. It asserts an external BUSY signal on the losing port synchronously on the fifth (5<sup>th</sup>) rising edge of the clock following the collision, including the rising edge of the clock that triggered the collision. The next rising edge of the clock deasserts it. The collision detection logic can also detect consecutive collisions. It also saves the busy address to a readable register named

## Collision Detection Timing

One timing parameter to be considered when doing a collision detection analysis is the clock-to-clock set-up time ( $t_{CCS}$ ). If  $t_{CCS}$  is violated in a write-write collision, both ports receive a busy signal. In the same situation, if  $t_{CCS}$  is met, then only the losing port receives a busy flag, and this is the port that first wrote into the memory. If  $t_{CCS}$  is met in a read-write scenario, then no busy flag is asserted, since the writing port has enough time to write to the location. If  $t_{CCS}$  is violated, then the reading port receives a flag, since the writing port may not have the time to write to the location. See the figures at the end of this application note for more details. Table 2 shows the actual timing parameters for  $t_{CCS}$ .

## Timing Example

There are two scenarios that can occur when using the collision detection circuitry in a system: both ports write to the same memory location, or one port writes to a memory location while the second reads from the same memory location.

**Table 2.  $t_{CCS}$  Timing Parameters**

| $t_{CCS}$                    |             |                             |             |  |
|------------------------------|-------------|-----------------------------|-------------|--|
| Port A (Early Arriving Port) |             | Port B (Late Arriving Port) |             | C/C rise to opposite C/C rise set-up time for non-corrupt data |
| Mode                         | Active Edge | Mode                        | Active Edge |  |
| SDR                          | C-R         | SDR                         | C-R         | (Clock Period)[min] - 1  |

The first scenario is comprised of two relevant cases: the rising edges of the port clocks violate the  $t_{CCS}$  timing parameter, or they meet it. In the first case, the busy flag of both ports will be asserted on the fifth clock edge following the collision. In the second case, when the  $t_{CCS}$  timing parameter is not violated, the port that writes first will receive a busy signal, since the port that arrives last will write over the previous data. The address that is under contention is then saved into the BUSY register. This value of the register can be read back to the address line. Figure 1 shows both cases mentioned above. Note that the third clock cycle for Port A contains two diagonal strips signifying that two cycles have been completed, not just one. The busy flag is deasserted on the rising edge of the clock immediately following its assertion.

The second scenario is when both the processors are accessing the same location at the same time and only one processor is doing a write. For instance, if the first microcontroller is writing to a memory location, and the second microcontroller attempts to read from the same location within  $t_{CCS}$ , then it will receive a busy signal on the fifth rising edge of the clock. The address that is under contention is then saved into the BUSY register. This value of the register can be read back to the address line. However, if the second microcontroller meets  $t_{CCS}$ , it reads the new data and no busy flag is asserted. In the same situation, if the reading microcontroller reads from an address, and the writing microcontroller meets  $t_{CCS}$ , then the reading microcontroller reads old data. Figure 2 illustrates both of these cases. The busy flag is deasserted on the rising edge of the clock immediately following its assertion.

Figure 1. BUSY Timing, WRITE-WRITE Collision with  $t_{CCS}$  Violation Followed by Non-violation.  
Third cycle of Port A clock represents two clock cycles.

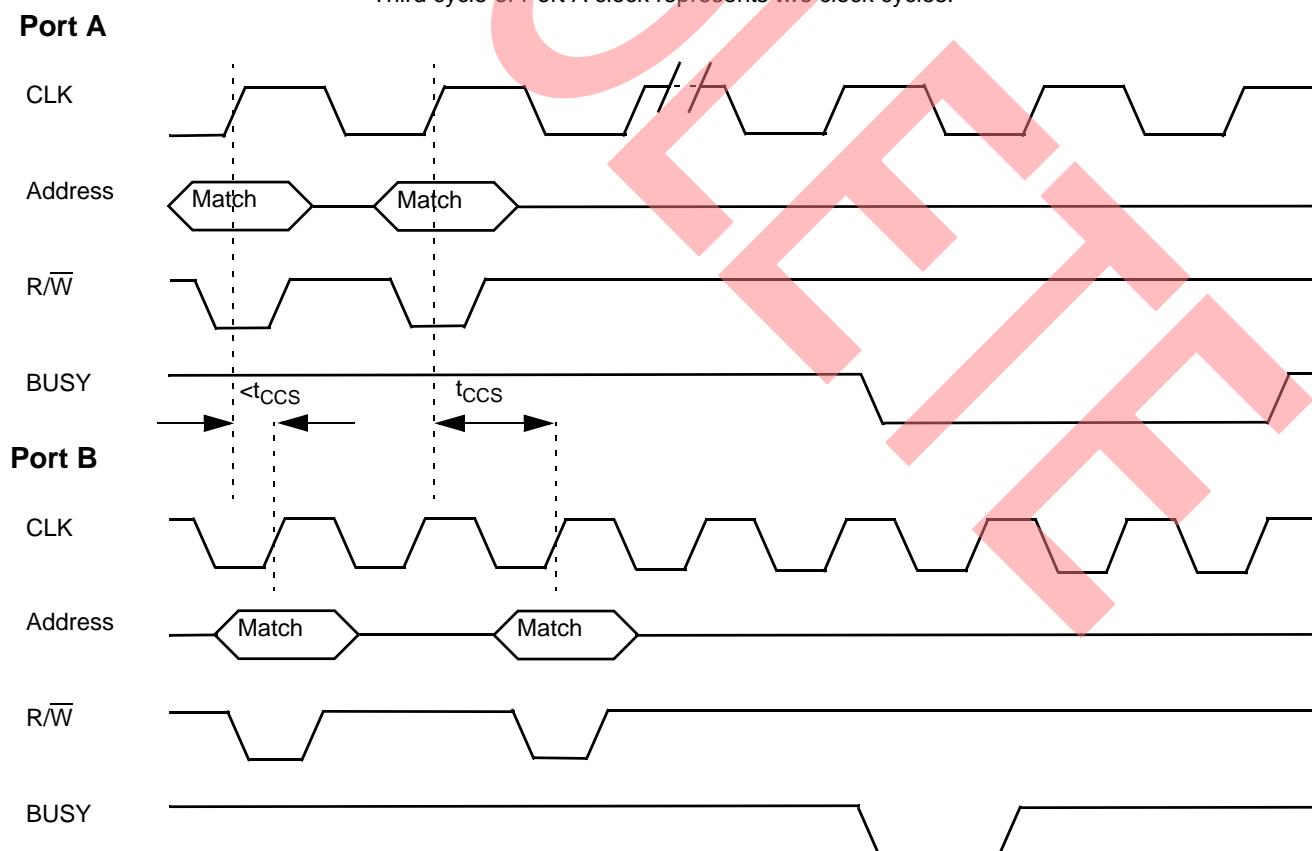
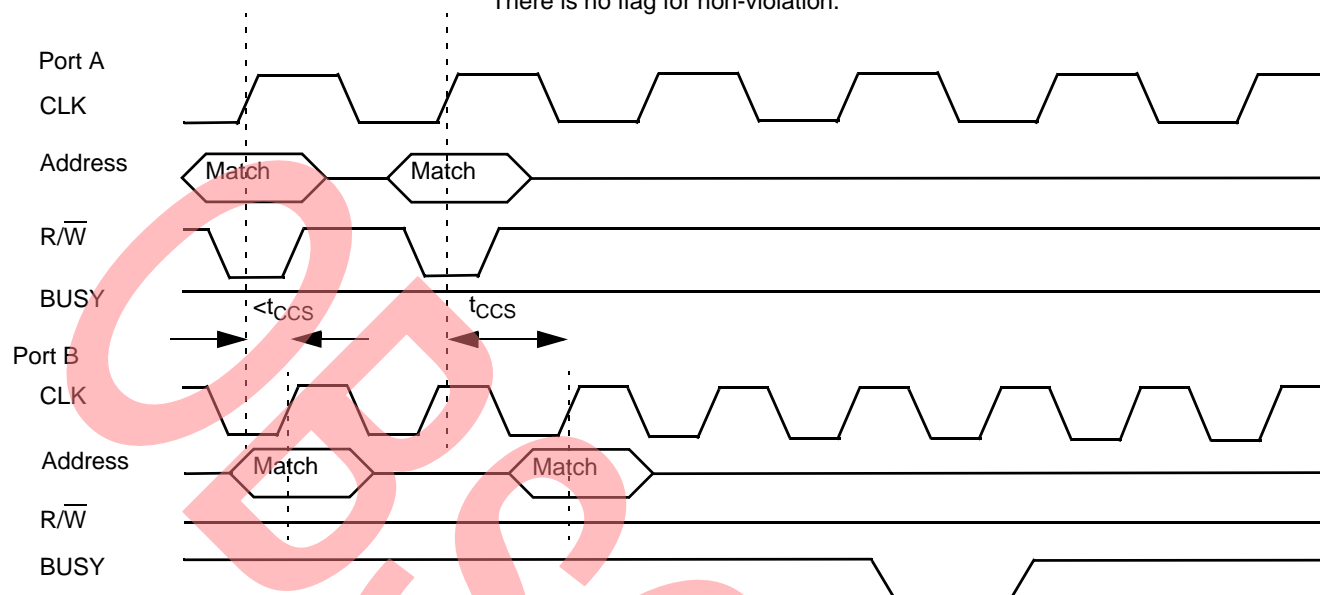


Figure 2. BUSY Timing, WRITE-READ Collision with  $t_{CCS}$  Violation followed by Non-violation.  
There is no flag for non-violation.



## Conclusion

Most systems can easily benefit from the added flexibility provided by the collision detection circuitry. As this application note has shown, this feature helps to eliminate the need for external logic. This simplifies the system design and can contribute to lowering total design time. Also, the data stored in the dual port can now be guaranteed.

For further information, visit the [Cypress](http://www.cypress.com) website. The website also provides the latest datasheets, models, and any related documentation.

## Document History

**Document Title:** Using the Collision Detection with FullFlex™ Synchronous Dual Port SRAMs - AN5025

**Document Number:** 001-14796

| Revision | ECN     | Orig. of Change | Submission Date | Description of Change   |
|----------|---------|-----------------|-----------------|---|
| **       | 988660  | PSR             | 04/25/2007      | Existing Application Note in the web - Added Spec No. and new disclaimer and also updated the copyright date. |
| *A       | 3127873 | ADMU            | 01/05/2011      | Updated Template and removed DDR information.   |
| *B       | 3250457 | ADMU            | 05/17/2011      | Modified abstract.  |
| *C       | 4401045 | ADMU            | 06/06/2014      | Obsolete document. Content moved into a white paper, 001-92810.   |

FullFlex is a trademark of Cypress Semiconductor Corporation. All other product and company names mentioned in this document are the trademarks of their respective holders.

Cypress Semiconductor  
198 Champion Court  
San Jose, CA 95134-1709  
Phone: 408-943-2600  
Fax: 408-943-4730  
<http://www.cypress.com>

© Cypress Semiconductor Corporation, 2005-2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

OBsolete