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.

AN1044

Understanding Cypress Asynchronous Fifos

Author: Adithi Perepu

Associated Part Family: CY7C421

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN1044.

AN1044 gives an overview of architecture, features, and expansion logic of the asynchronous FIFO CY7C421, and discusses the common FIFO problems and their solutions.

Contents

1 Introduction

This application note describes the internal architecture of Cypress' asynchronous FIFO CY7C421. A summary of key device features, applications, failure modes, typical problem symptoms and solutions is also included. Timing parameters specified in this application note are reproduced from the device datasheet - CY7C421, 512 x 9 Asynchronous FIFO.

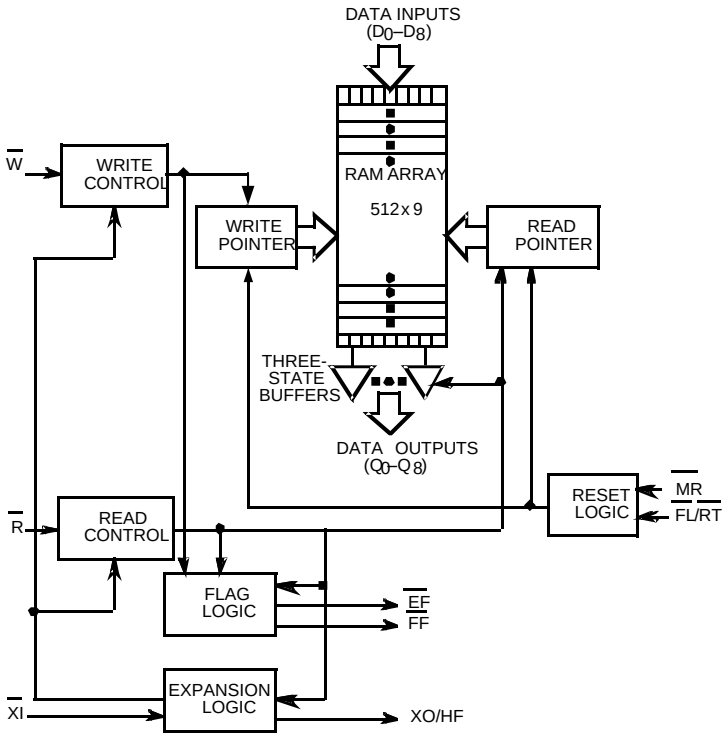
2 Asynchronous FIFO Overview

The Cypress asynchronous FIFO (CY7C421) is 512 words deep and 9 bits wide. This monolithic device offers access times as fast as 15 nanoseconds and cycle times as fast as 25 nanoseconds. Refer to device datasheet for information on valid speed and package combinations.

Cypress Asynchronous FIFOs employ an SRAM type of interface with dedicated read/write ports, which allow independent read/write operations. The FIFO uses specially designed dual-port SRAM cells which have separate read and write transistors to support simultaneous access from both ports.

The CY7C421 Asynchronous FIFO is organized such that data is read out in the same sequential order in which it was written. Full, half-full, and empty flags facilitate write and read operations. Additional pins are provided to facilitate expansion in width and depth. Refer to Figure 1 for the FIFO logic block diagram.

Figure 1. FIFO Block Diagram



3 FIFO Read / Write Operations

Asynchronous FIFO read/write timings are illustrated in Figure 2 and Figure 3. Read operation is initiated at the falling edge of read enable ($\overline{R}$). The output data bus, Q_0–Q_8, provides valid data t_A after the falling edge of $\overline{R}$. This t_A period is referred to as the FIFO's read access time. The output data bus transitions out of the high-impedance state t_{LZR} after $\overline{R}$ is asserted. Care should be taken to ensure that a read is performed only after valid data is available on the bus (t_A after the falling edge of $\overline{R}$). De-assertion of $\overline{R}$ ends the read operation.

The data on the Q_0–Q_8 bus remains valid for t_{DVR} after the rising edge of $\overline{R}$. This is the output data hold time at the end of the read cycle. The internal circuitry then prepares itself for the next read operation. This period is referred to as the t_{RR}, or read recovery time. Subsequent read operations should not be initiated within this period. The minimum pulse width denoted by t_{PR}, is required for read access and is equivalent to the read access time, tA.

Read cycle time (t_{RC}) is calculated as:

t_{RC} = Access time (t_A) + Read Recovery time (t_{RR})

The maximum read frequency is the reciprocal of t_{RC}, i.e.,

Read frequency (max) = $1/(t_A + t_{RR})$.

For example, a Cypress FIFO with a 20 ns access time and 10 ns read recovery time results in a 30 ns read cycle time,or 33.3 MHz maximum read cycle frequency.

The write operation is similar to the read operation. A write operation is initiated on the assertion of the write signal, $\overline{W}$, and terminates with the de-assertion of $\overline{W}$ (rising edge). For a valid write to occur, the input data bus, D_0–D_8, must be stable for t_{SD} (setup time) prior to the rising edge of $\overline{W}$ and should remain valid for t_{HD} (hold time) after this edge. A minimum write enable pulse width of t_{PW} is required for a valid write operation. A write recovery time, t_{WR}, is required between consecutive write cycles.

Maximum write frequency is $1/(t_{PW} + t_{WR})$. For example, a device with a 15 ns write strobe width and a 10 ns write recovery time yields a 25 ns write cycle time, or a 40 MHz maximum write frequency.

FIFOs include separate internal write and read counters (pointers). The write pointer always points to the next word to be written to and the read pointer always points to the current FIFO word to be read. Each write or read operation increments the appropriate counter by one position. The relative position of these counters determines device status, which is indicated externally via empty, half-full, and full flags.

Figure 2. Asynchronous Read Timing

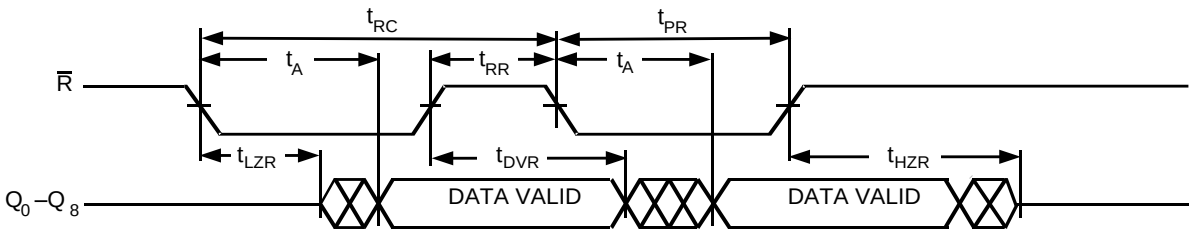
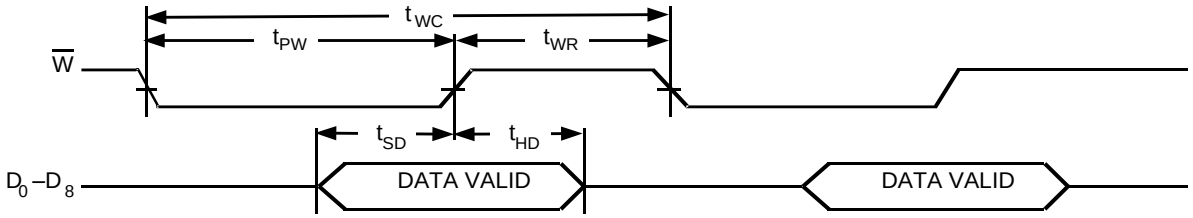


Figure 3. Asynchronous Write Timing



4 Common FIFO Configurations

Multiple asynchronous FIFOs can be cascaded to create wider and/or deeper FIFOs with minimal external logic. The external logic implements an OR gate is required to generate composite flag. The following section describes standalone operation, width expansion, and depth expansion.

Cypress FIFOs provide pins $\overline{XI}$ and $\overline{XO}$ to implement an expansion logic (both width and depth expansion). $\overline{XI}$ and $\overline{XO}$ pins are used token passing from one FIFO and the next one. $\overline{FL}$ indicates the first FIFO to be loaded with the data.

4.1 Standalone and Width Expansion Configurations

Figure 4 illustrates stand alone configurations. In this configuration, the $\overline{XI}$ (Expansion in) pin is tied LOW and the $\overline{FL}$ (first load) pin is tied HIGH.

Figure 4. Standalone Operation

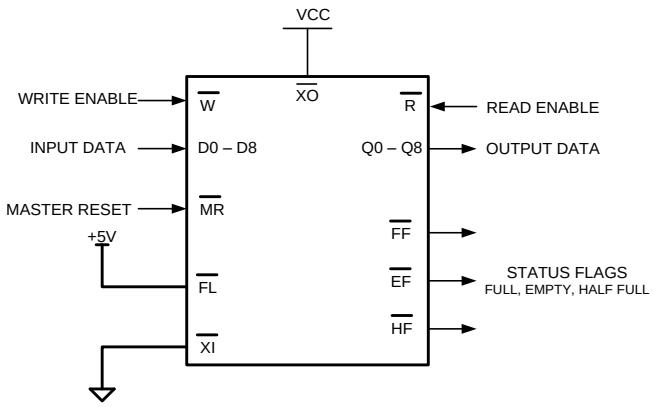
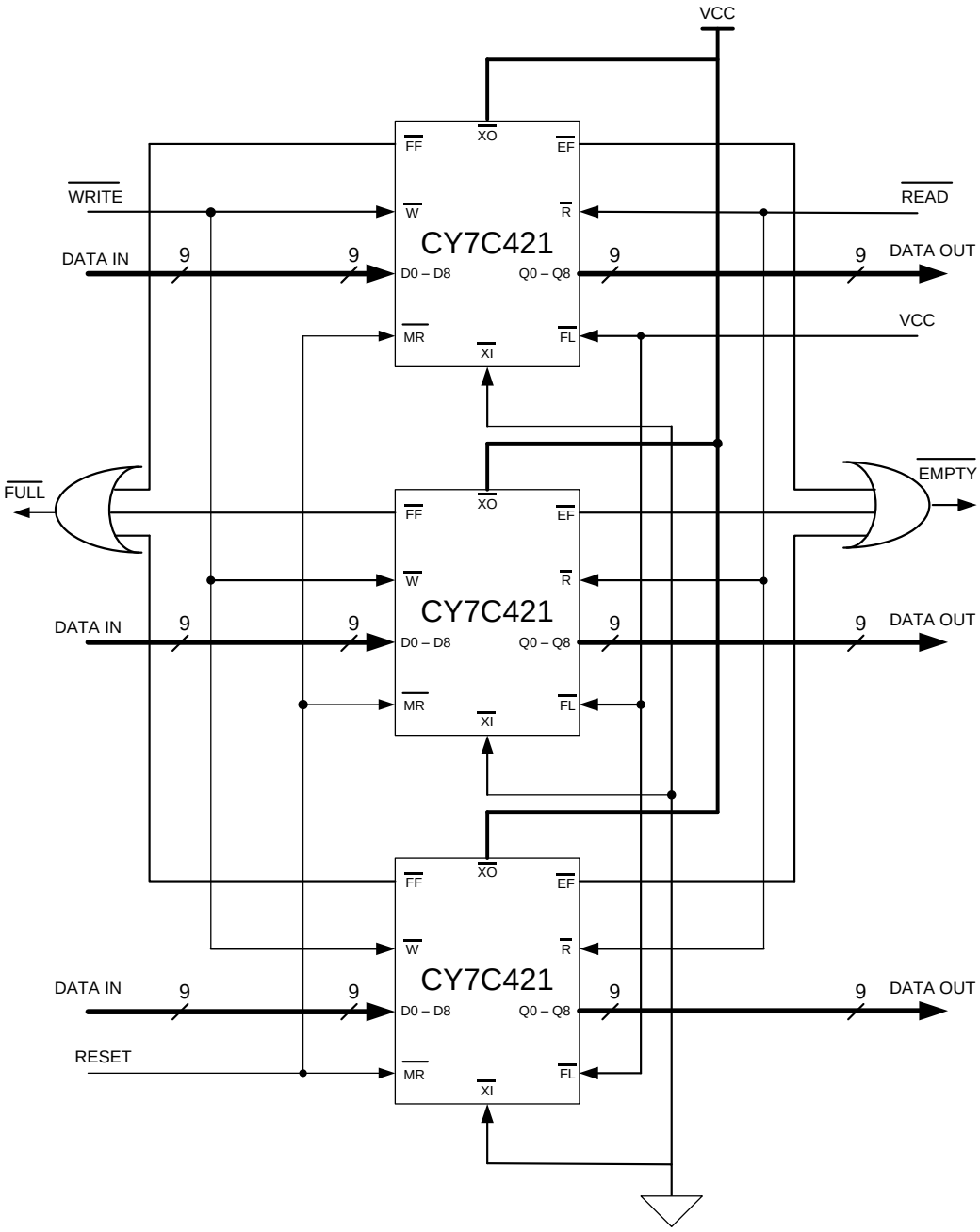


Figure 5 illustrates the width expansion configuration. Similar to standalone configuration, the FIFO pins XI is tied LOW and FL is tied HIGH.

In width expansion, propagation delays might prevent individual FIFOs in the design from entering full, half full or empty conditions simultaneously. Hence, composite flags must be generated externally to properly reflect the instantaneous state of the width expanded FIFO. This can be implemented with an OR gate.

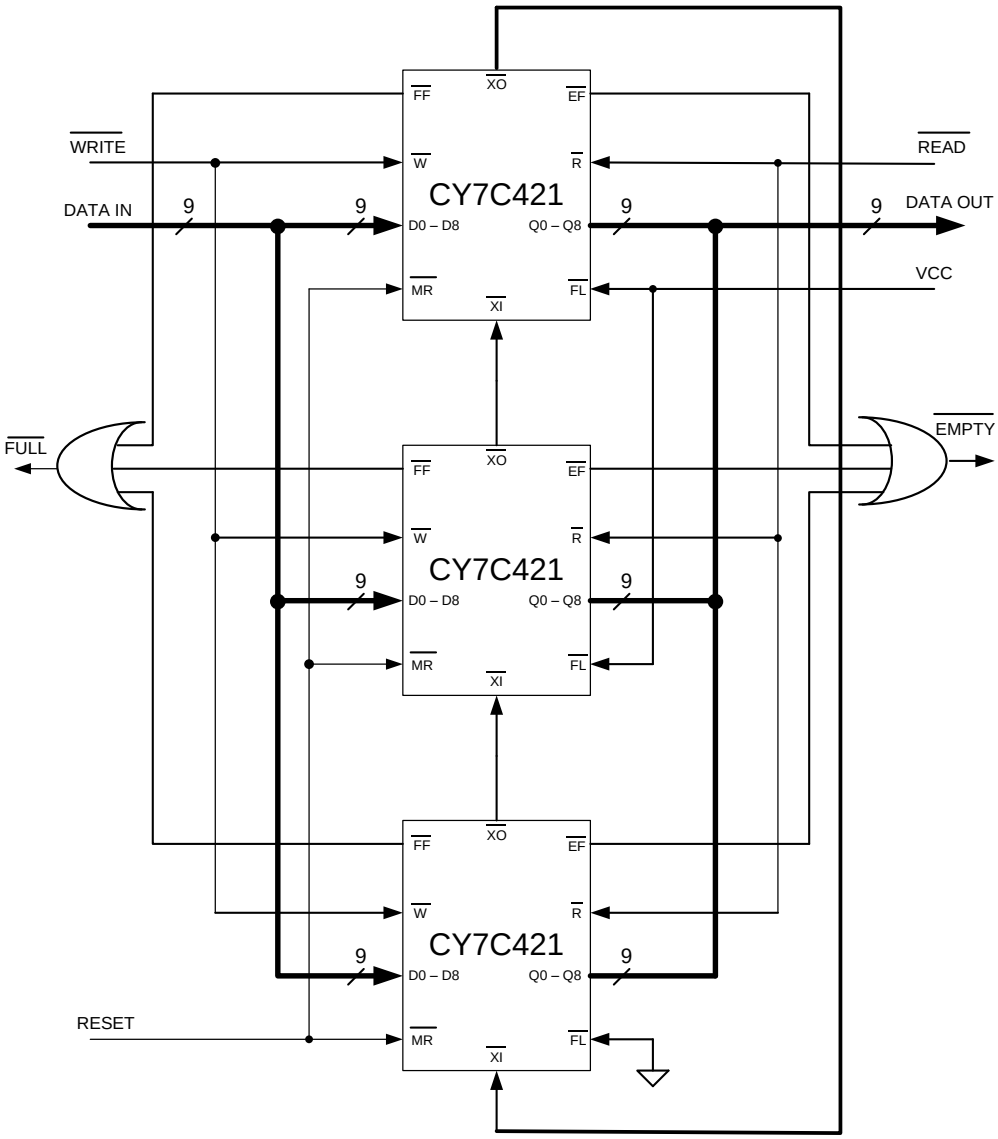
Figure 5. Width Expansion



4.2 Depth Expansion Configuration (Token Passing mechanism)

Figure 6 illustrates depth expansion. In this configuration, the $\overline{FL}$ (first load) pin on one device must be tied LOW to designate that device as the first FIFO to be written to. The FIFOs are then daisy-chained together by connecting $\overline{XO}$ (expansion out) of one device to the $\overline{XI}$ (expansion in) pin of the next. The $\overline{XO}$ signal of the last device in the chain is connected to the $\overline{XI}$ pin of the first device, thus forming a token-passing ring.

Figure 6. Depth Expansion (Token Passing)



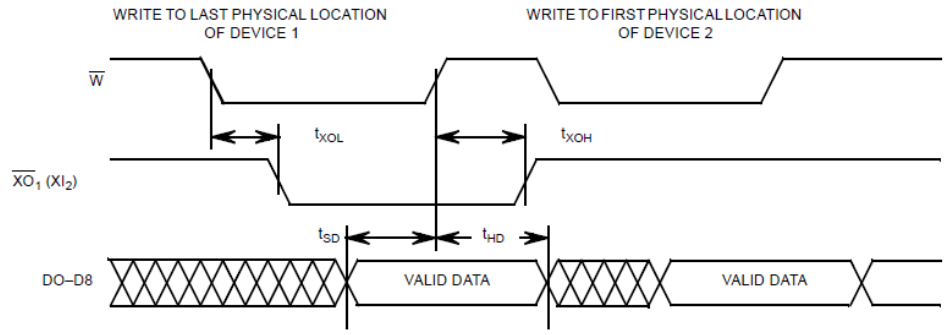
As in the case of width expanded FIFOs, an OR gate can be used to generate composite empty and full flags. Token passing ensures that write and read processes remain consistent. The read/write token determines the device that is accessed for a read/write operation. In the token-passing procedure for write operations, the first FIFO is written to until it is filled. An internal write pointer determines the location being written to and after every write, the pointer is incremented. When the write pointer reaches the last physical location in the first FIFO, no more writes can occur to that device. At that point, the first FIFO passes the write token to the next FIFO in the chain via the $\overline{XO}$ to $\overline{XI}$ interface. The second device, now in possession of the write token, receives all future written data until this device also fills up and passes the write token onto the next device in the chain.

If enough writes occur to fill up the FIFO chain, the last device fails in its attempt to pass the write token back to the first device. This is because a full FIFO cannot accept a write token. No further writes to the FIFO chain are allowed until a read operation occurs, which frees up an internal location. The relative position of the internal write and read counters determine device status and whether it can accept data though a write operation. Figure 7 shows the timing for write operations.

Similar to the write process, the first FIFO in the chain holds the read token. When the FIFO chain is read from, the device holding the read token supplies the data from the address specified by the device's read pointer. The read pointer is then incremented. The incrementing continues until the FIFO is empty, and the read token is passed to the next device in the chain. The passing of the read token is done via the $\overline{XO}$ to $\overline{XI}$ interface. Figure 8 shows the timing for read operations.

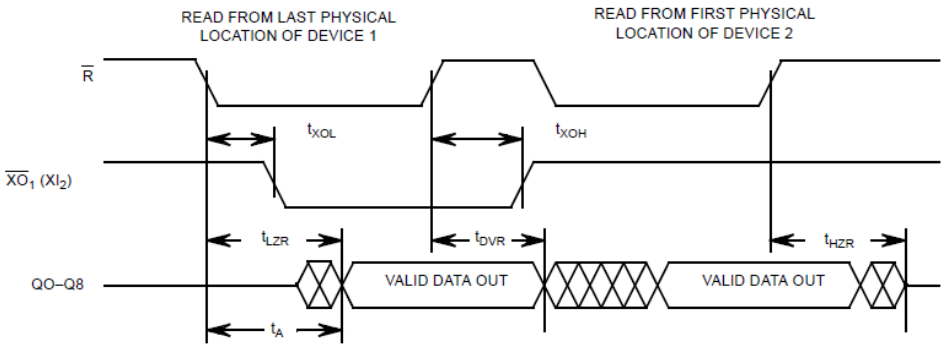
A depth-expansion design must generate composite status flags to reflect the instantaneous state of the entire FIFO chain, as is done for width expansion.

Figure 7. Write Expansion Timing



Expansion out of device 1 ($\overline{XO_1}$) is connected to expansion in of device 2 (XI_2).

Figure 8. Read Expansion Timing



Expansion out of device 1 ($\overline{XO_1}$) is connected to expansion in of device 2 (XI_2).

5 Retransmit Feature

The retransmit feature is useful in tele-communication applications, for retransmitting packets of data; in disk drives for rewriting sectors. It is especially useful in applications where a single block of data in the FIFO must be sent out multiple times, as in a word or pattern generator.

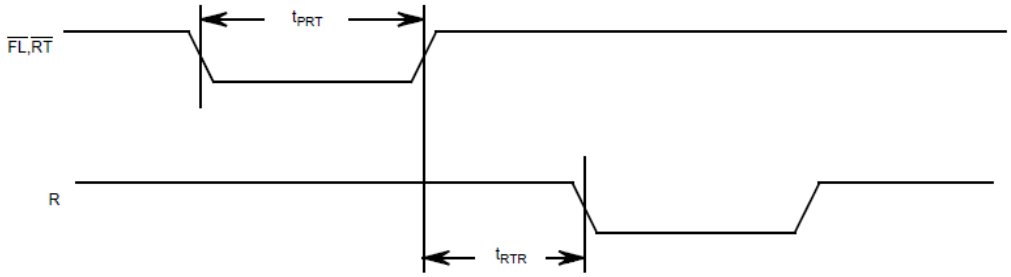
Data can be retransmitted any number of times, and with Cypress FIFOs, the retransmit feature can be used at any time, no matter how much data the FIFO contains. This is an advantage compared to some competing FIFOs which do not allow use of the retransmit function when the FIFO is full or if it has less than the required number of words.

In the retransmit operation, the read pointer is reset to its initial location and the $\overline{R}$ pin is pulsed until the read pointer advances to the same memory location addressed by the write pointer. The retransmit ($\overline{RT}$) pin is available in the standalone and width-expansion modes. Depth expansion mode does not support the retransmit feature and this pin designates the FIFO to be loaded first.

The retransmit function is initiated by asserting an active-LOW pulse to the retransmit input, which resets the internal read counter to zero. $\overline{R}$ input is held inactive during this time; otherwise, conflicting requirements may corrupt the read counter. The retransmit process does not affect the state of the write counter or the write process and no design or usage rules are violated if retransmit and write cycles overlap or occur simultaneously. The device does not lock up, and data is neither lost nor corrupted if the retransmit timing constraints shown in Figure 9 are met.

Keeping track of what data is currently in the FIFO and what data is being read out can become complicated when writes and retransmit are performed simultaneously. For example, consider a scenario where the FIFO is being written to and the retransmit function is activated when the FIFO is half full. The FIFO begins to retransmit/read data starting from its initial location and the reads continue until the empty condition is reached. The FIFO may be filled to three quarters full before the read pointer catches up with the write pointer and the new data (written into the FIFO after retransmit was activated) is also read out. Hence keeping track of the data can be challenging in such scenarios.

Figure 9. Retransmit Timing



t_{PRT} is the minimum retransmit pulse width.

t_{RTR} is the retransmit recovery time. It is a timing window that must not be violated.

6 Applications

A FIFO allows two systems running at different data rates to communicate by providing a temporary data or control buffer.

Typical FIFO applications include:

- Interprocessor communications
- Communications systems, including local area networks
- Digital-signal-processing-based systems for buffering real-time data
- Electronic data processing, CPU, and peripheral equipment, including high-performance disk controllers

7 Design Considerations and Solutions

The following section discusses some of the common design consideration for Cypress Asynchronous FIFOs and their solutions respectively.

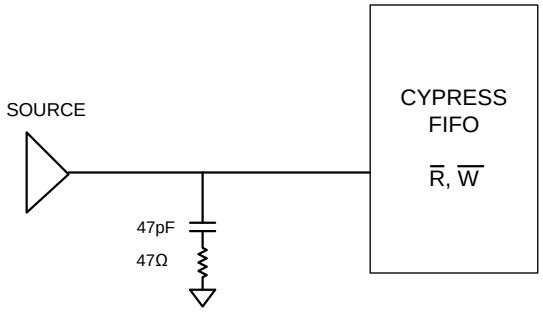
7.1 Corrupted or Repetitive Data

The most common cause for corrupted and repetitive data to be present in a FIFO is a spurious active signal (glitch) on the FIFO's $\overline{W}$ input. Write glitches cause the logic levels present at the data inputs to be written into the FIFO. This causes false data to be written into the device. A write glitch causes this data to be duplicated if a valid data is present at the data inputs.

Write glitches are often the result of voltage reflections due to impedance mismatches, which can be eliminated using impedance-matching termination networks. Termination networks are recommended on the $\overline{W}$ and $\overline{R}$ traces on printed circuit boards (PCBs) when the transmission lines exceed approximately 4 inches from source to destination, assuming a 2 ns rise/fall time for the read and write signals. For $\overline{R}$ and $\overline{W}$ signals with less than 2 ns rise/fall times, line lengths as short as 1 inch will require termination.

A termination network matches the load impedance with the PCB trace characteristic impedance, which is usually 50 Ω or less (for microstrip or stripline construction on G-10 glass epoxy material). To minimize voltage reflections, a slightly overdamped termination is preferred. Cypress recommends a 47 pF (max) capacitor, in series with a 47 ohm resistor to be connected from the read/write pin to ground (Figure 10). This termination network acts as a high-pass filter for high-frequency pulses and does not dissipate DC power. Read or write lines that drive more than one FIFO device require only one termination network. Connect the network at the input that is electrically farthest from the source. For multiple loads, see the "SRAM System Design Guidelines" white paper for help in determining the maximum line length.

Figure 10. Recommended Termination Network

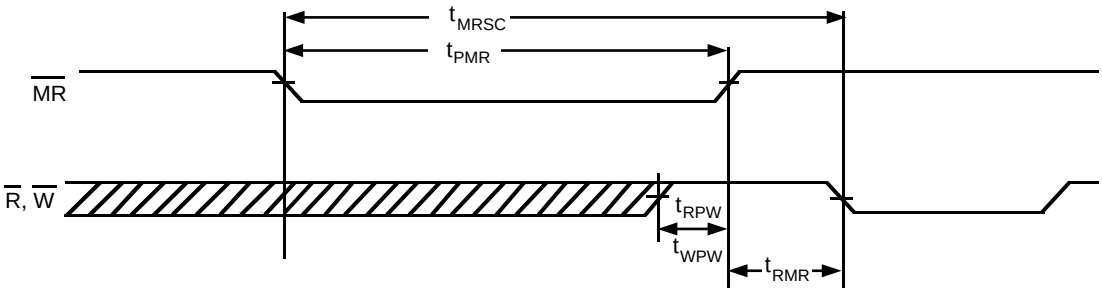


FIFO data corruption can also be caused by violation of master-reset timing constraints. As shown in Figure 11, read and write signals must not be asserted around the rising edge of MR (master reset) to satisfy the t_{RMR} (master-reset recovery-time) specification. This constraint is necessary because the FIFO goes through an internal initialization process and requires a settling period after the reset terminates.

7.2 FIFO Locks Up

Short noise pulses on the MR pin can cause the FIFO to be "partially reset" because of which the FIFO may not respond. An appropriate termination scheme must be incorporated on line to prevent this issue.

Figure 11. Master Reset Timing



7.3 Missing or Disappearing Data

Glitches on the R input can cause data to disappear because of an unintended read operation. This increment the internal read counter, resulting in data loss. An appropriate termination scheme must be incorporated on R line to eliminate these unwanted glitches.

7.4 Repetitive or Out-of-Sequence Data, False Full or Empty

A misaligned internal read or write pointer can cause a variety of symptoms, including repetitive or out-of-sequence data and false full and/or empty conditions. The two most common causes of misaligned pointers are master-reset violations and boundary-condition violations.

Boundary conditions are defined as the FIFO being either full or empty. When FIFOs are connected in parallel to make a wider word, certain conditions may cause the individual to either ignore or act upon a read or write request. The system-level symptom of individual FIFOs making different decisions is word misalignment. The problem occurs in the empty condition when a read immediately follows a write operation and in the full condition when a write immediately follows a read operation.

7.4.1 Operation at the Empty Boundary

Consider a FIFO that has been reset and is empty. The empty flag is active (LOW), and internal logic inhibits read operations. Generally, the read and write signals are asynchronous. Upon completion of a write operation the internal state of the FIFO goes from empty to (empty + 1). During this interval, a read operation may or may not be recognized. A read operation preceding the write is ignored; a read following the write is not. In between these conditions, the FIFO decides whether to recognize the read. During this window of uncertainty, it cannot be determined whether the read will be ignored or not. With one FIFO, this uncertainty is acceptable. However, if two or more FIFOs are connected in parallel to make a wider word, some might ignore the read, and others might not.

7.4.2 Waiting at the Empty Boundary

Figure 13 shows the timing that prevents problems with reads at the empty boundary. Any device reading from the FIFO must wait for a period of time, t_{RAE}, after the completion of the write operation before initiating a HIGH-to-LOW transition of the $\overline{R}$ signal. The $\overline{W}$ signal's rising edge indicates the completion of the write operation.

7.4.3 Operation at the Full Boundary

A similar condition occurs when a single FIFO becomes full. The full flag is active (LOW), and internal logic inhibits write operations. A read operation immediately followed by a write operation causes the FIFO to go from full to full – 1 and back to full. During the time the FIFO is going from full to full – 1, a write operation might or might not be recognized. The aperture of uncertainty applies here because the FIFO takes a finite amount of time to change states, and a write command arriving at this instant might be ignored.

One way to satisfy this timing is to gate read operations with the composite empty flag ($\overline{EF}$) such that the read operation is prevented when the empty flag is active. Note, however, that the $\overline{R}$ signal can be LOW either before or during the first write to the empty FIFO and the data still propagates to the outputs correctly.

7.4.4 Waiting at the Full Boundary

Figure 13 shows the timing that prevents problems with writes at the full boundary. Any device writing to the FIFO must wait an amount of time, t_{WAF}, after the termination of the read operation before causing a HIGH-to-LOW transition of the $\overline{W}$ signal. The $\overline{R}$ signal's rising edge indicates the end of the read operation.

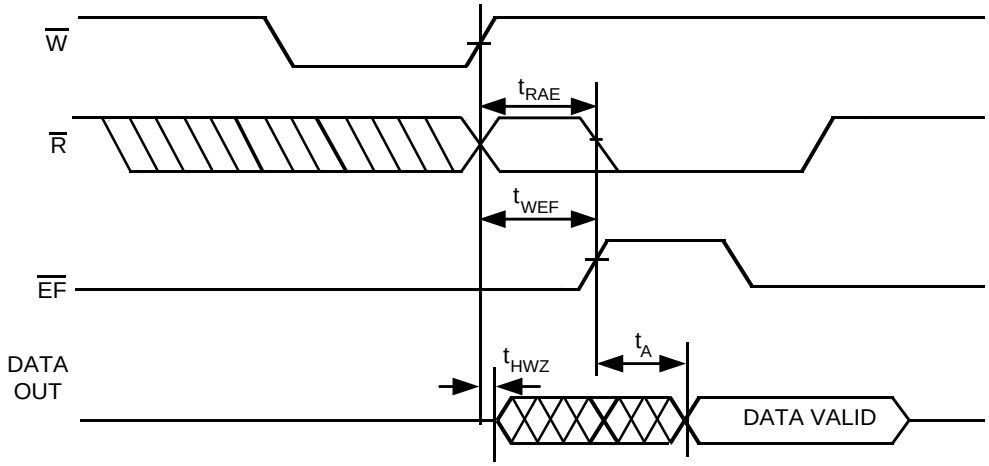
This timing is met by gating write operations with the composite full flag ($\overline{FF}$) such that the write operation is prevented when the full flag is active. However, the $\overline{W}$ signal can be LOW either before or during the first read from a full FIFO and the data is still properly written.

7.5 Empty Reads and Full Writes

When Cypress FIFOs are empty, their data outputs go to the high-impedance state. Therefore, attempting to read from an empty FIFO yields unpredictable data. Internal logic inhibits the read, and the read pointer is not incremented.

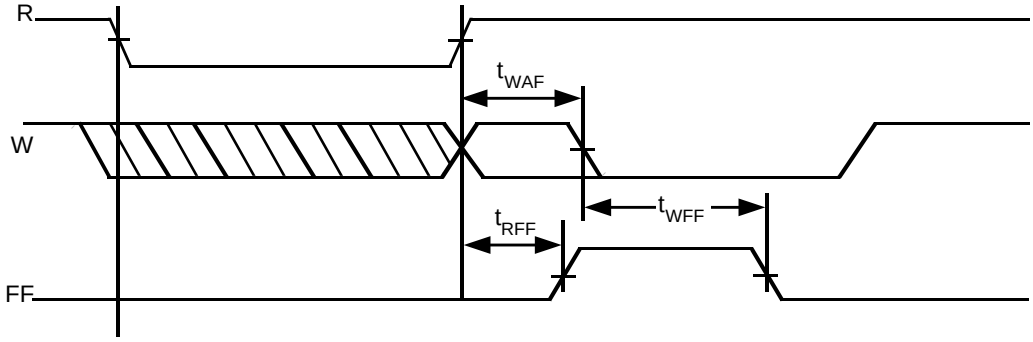
Internal logic also inhibits attempts to write to a full FIFO, and the write pointer is not incremented.

Figure 12. Read Fall-Through Timing Violation



t_{RAE} is an invalid read window. A read operation should never be initiated inside this window.

Figure 13. Write Bubble-Through Timing Violation



t_{WAF} is an invalid write window. A write operation should never be initiated inside this window.

7.6 Effective Pulse Width Violation

This phenomenon can occur at either the empty or the full boundary if the flags are not properly used. The empty flag must be used to prevent reading from an empty FIFO and the full flag must be used to prevent writing into a full FIFO. Otherwise, the effective pulse width of the read or the write strobe will be violated, even though the actual signals meet the data sheet specifications.

Consider a case where the FIFO is empty and is receiving the read pulses. This operation will be ignored since the FIFO is empty. In the next operation a single word is written into the FIFO, moving the FIFO to (empty + 1). In the meanwhile if the read signal continues to assert, and if the write signals occurs slightly before the rising edge of the read signal an effective minimum LOW read pulse width violation will occur.

Similarly, the minimum write pulse width may be violated by attempting to write into a full FIFO and asynchronously performing a read. The empty and full flags must be used to avoid these effective pulse width violations.

8 Summary

Cypress's Asynchronous FIFOs are dual port devices which simplify the task of data synchronization across clock domains. A simple interface combined with features like status flags, retransmit feature and support for width / depth expansion make these devices ideally suited for inter – processor communication systems.

About the Author

Name: Adithi Perepu.

Title: Applications Engineer Staff

Document History

Document Title: AN1044 - Understanding Cypress Asynchronous FIFOs

Document Number: 001-25919

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1420883	ADMU	09/12/2007	New Spec.
*A	3082092	ADMU	11/09/2010	Removed information about parts which are obsolete. Updated to new template.
*B	3181540	ADMU	02/24/2011	Updated Document Title. Updated Abstract.
*C	3406989	ADMU	11/04/2011	Modified content and sequencing of topics. Added Figure 1 (FIFO Block Diagram).
*D	3686347	ADMU	07/20/2012	Updated Asynchronous FIFO Overview. Updated FIFO Read / Write Operations. Added Summary. Updated to new template.
*E	3898216	ADMU	01/29/2013	Updated Figures. Moved the figures after relevant explanation of the content. Replaced the Heading “Common Problems and Solutions” with “Design Considerations and Solutions” “SRAM System Design Guidelines” is a white paper.
*F	3957925	ADMU	04/08/2013	Updated Document Title in Document History Page (To match with spec title on first page).
*G	4584927	ADMU	12/05/2014	Updated Asynchronous FIFO Overview: Updated Figure 1. Updated FIFO Read / Write Operations: Updated Figure 2, Figure 3. Updated Common FIFO Configurations: Updated Standalone and Width Expansion Configurations: Updated Figure 4, Figure 5. Updated Depth Expansion Configuration (Token Passing mechanism): Updated Figure 6. Updated Design Considerations and Solutions: Updated Corrupted or Repetitive Data: Updated Figure 10. Updated FIFO Locks Up: Updated Figure 11. Updated Empty Reads and Full Writes: Updated Figure 12, Figure 13. Updated to new template.
*H	5826306	AESATMP9	07/21/2017	Updated logo and copyright.
*I	5966402	NILE	11/14/2017	Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709