AN1044

Understanding Asynchronous FIFOs

Author: Adithi Perepu
Associated Project: No
Associated Part Family: CY7C421
Software Version: NA
Related Application Notes: None

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

Contents
Introduction ............................................................. 1
Asynchronous FIFO Overview .................................. 1
Writing to and Reading from the FIFO .................... 3
Common FIFO Configurations ................................. 3
Retransmit ............................................................ 6
Applications ......................................................... 6
Common Problems and Solutions ............................... 6
  Corrupted or Repetitive Data ............................... 6
  FIFO Locks Up ................................................ 7
  Missing or Disappearing Data .............................. 7
  Repetitive or Out-of-Sequence Data, False Full or Empty .................................................. 7
  Operation at the Empty Boundary ...................... 7
  Operation at the Full Boundary ......................... 8
  Waiting at the Empty Boundary ......................... 8
  Waiting at the Full Boundary ............................ 8
  Empty Reads and Full Writes ............................. 8
  Effective Pulse Width Violation ......................... 9
  Intermittent Malfunctions ............................... 9

Introduction
This application note explains the internal architecture of the asynchronous FIFO made by Cypress (CY7C421) and its functionality - the writing and reading process. It also discusses FIFO configurations - depth and width expansion and the retransmit feature available in these devices. Other topics briefly covered in this application note include FIFO applications, failure modes, typical problem symptoms and solutions. This information applies to the Cypress Asynchronous FIFO CY7C421.

Timing parameters given in this application note are from CY7C421 512 x 9 Asynchronous FIFO datasheet.

Asynchronous FIFO Overview
The Cypress asynchronous FIFO (CY7C421) is 512 words deep with a 9-bit word width. This monolithic device is available in a wide variety of packages with the industry standard pinout and with access times as fast as fifteen nanoseconds and cycle times as fast as twenty five nanoseconds. Not all speed grades are available in all packages; see CY7C421 512 x 9 Asynchronous FIFO datasheet to determine valid speed, package combinations. The smallest package available is the 32-pin 7 mm x 7 mm TQFP, which occupies less than one-third the area of a 300-mil-wide 28-pin DIP.

Although the first FIFOs utilized a shift-register type of architecture, today’s asynchronous FIFOs employ an SRAM type of interface. Data is written into and read out of the device, as with SRAM write and read operations. These operations can occur independently of one another and are made possible by a specially designed six-transistor, dual-ported SRAM cell. The SRAM cell makes use of separate read and write transistors to allow independent R/W operation.

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 writing and reading. Additional pins are provided to facilitate unlimited expansion in width and depth, with no performance penalty.
Figure 1. FIFO Block Diagram

Figure 2. Asynchronous Read and Write Timing
Writing to and Reading from the FIFO

Figure 2 shows the asynchronous FIFOs’ read and write timing. Reads and writes are asynchronous to each other. The read process begins with \( R \)'s falling edge. The output data bus, Q0–Q8, leaves the high-impedance state \( t_LZR \) ns after \( R \)'s falling edge. The output data becomes valid \( t_A \) ns after that same falling edge. This \( t_A \) period is referred to as the FIFO’s read access time. \( R \)'s rising edge ends the read process.

The data on the Q0–Q8 bus remains valid for \( t_{DVR} \) ns following the \( R \) rising edge. This is the output data hold time at the end of the read cycle. The internal circuitry then reads itself for the next read operation. This period is referred to as the \( t_{RR} \), or read recovery time, and must be observed between consecutive read operations. The read signal’s minimum pulse width is denoted by \( t_{PR} \) and is identical to the read access time, \( t_A \).

You can determine the read cycle time (\( t_{RC} \)) by adding the access time (\( t_A \)) and the read recovery time (\( t_{RR} \)), which you can find in the FIFO data sheet. The maximum read frequency is the reciprocal of \( t_{A} + t_{RR} \). For example, a Cypress FIFO with a 20 ns access time and a 10 ns read recovery time results in a 30 ns read cycle time, or 33.3 MHz maximum read cycle frequency.

The write process is similar to the read process. A write begins with the falling edge of the write line, \( W \), and terminates with \( W \)'s rising edge. For a valid write to occur, the input data bus, D0–D8, must be stable for \( t_{ISD} \) ns prior to \( W \)'s rising edge and for \( t_{HD} \) ns after this edge. These specifications are referred to as the data set-up and hold times, respectively. The write strobe also has a minimum negative pulse width, denoted as \( t_{PW} \). A minimum recovery time, \( t_{WR} \), is required between write cycles.

The maximum write frequency is the reciprocal of \( t_{PW} + t_{WR} \). As an 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 cycle frequency.

The FIFOs include separate internal write and read counters (pointers). The write pointer always points to the next word to be written 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. When the FIFO is empty or full, both counters point to the same location. The relative position of these counters determines the device’s status, which is indicated externally via empty, half-full, and full flags.

Common FIFO Configurations

All asynchronous FIFOs can be interconnected, without external logic, to create either wider FIFOs, deeper FIFOs, or both. Standalone operation, width expansion and depth expansion are described next.

Figure 3 shows standalone mode, and Figure 4 shows width expansion mode. In both these modes, the \( XI \) (expansion in) pin is grounded and the FL (first load) pin is tied HIGH.

Composite flags must be generated to properly reflect the instantaneous state of the width or depth expanded FIFO. Composite flags are necessary because variations in propagation delays might prevent the individual FIFOs in the design from entering the F, HF, or E states simultaneously. The OR gates in the width-expansion design are used to generate the composite full and empty flags (F, E). Similarly, the composite half-full flag (HF) is generated.

Figure 5 shows depth expansion. The \( FL \) (first load) pin on one device must be grounded to define that FIFO as the first FIFO to be written to. The FIFOs are then daisy-chained together by connecting one device’s \( XO \) (expansion out) output pin to the next device’s \( XI \) (expansion in) input. The \( XO \) of the last device in the chain is connected to the \( XI \) of the first device, thus forming a token-passing ring.
Figure 4. Width Expansion

Figure 5. Depth Expansion
Token passing allows the writing and reading processes to stay consistent. That is, the passing and holding of a read or write token tells an individual FIFO whether it is actively being read from or written to. 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 written to, and after every write, the pointer is incremented. When the pointer reaches the last physical location, 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 $XO$ – $XI$ 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 the 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 positions of the internal write and read counters determine a device’s status and whether it can accept data through a write operation. Figure 6 shows the timing for write operations.

As with the procedure for writes, 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 $XO$ – $XI$ interface. Figure 7 shows the timing for read operations.

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

The retransmit feature is useful in communications for retransmitting packets of data and 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 in contrast 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 \( R \) pin is pulsed until the read pointer advances to the same memory location addressed by the write pointer. The retransmit \( (RT) \) pin is available in the standalone and width-expansion modes, but not in depth expansion because 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. Keep the \( R \) input inactive during this time; otherwise, the conflicting requirements on the read counter might cause it to become corrupted. 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 8 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.

Applications

FIFOs are asynchronous devices that are ideal for interfacing between two asynchronous processes. 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, in which bidirectional devices are especially useful
- 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

Common Problems and Solutions

To help prevent problems and correct them when they occur, this section describes the causes and solutions to some common FIFO problems. The first problem to consider is corrupted or repetitive data in a FIFO.

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 \( W \) input. Because Cypress devices are extremely fast, a write pulse as short as 3 ns initiates a write. Write glitches cause the logic levels present at the data inputs to be written into the FIFO, which can put false data into the device. If valid data is present at the data inputs, a write glitch causes this data to be written a second time, resulting in duplicated data.

Write glitches are often the result of voltage reflections due to impedance mismatches, which you can eliminate using impedance-matching termination networks.

Termination networks are recommended on the \( W \) and \( R \) traces on printed circuit boards (PCBs) when the lines exceed approximately 4 inches from source to a single load. This line length assumes a 2 ns rise/fall time for the read and write strobes. For \( R \) and \( W \) signals with sub 2 ns rise/fall times, line lengths as short as 1 inch might require termination.

A termination network matches the load impedance to the PCB trace’s characteristic impedance, which is typically 50 \( \Omega \) 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) series capacitor and a 47 ohm
A resistor be connected from the read or write pin to ground (Figure 9). This termination network acts as a high-pass filter to short, high-frequency pulses and dissipates no DC power. Read or write lines that drive more than one FIFO require only one termination network. Put the network at the input that is electrically farthest from the source. For multiple loads, see the “SRAM System Design Guidelines” application note for help in determining the maximum line length.

**Figure 8. 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.

**Figure 9. Recommended Termination Network**

FIFO data corruption can also be caused by violation of master-reset timing constraints. As shown in the timing diagram in Figure 10, the read and write signals must be inactive around the rising edge of $MR$ (master reset) to satisfy the $t_{RMR}$ or master-reset recovery-time specification. This constraint is necessary because the FIFO goes through an internal initialization process during reset and requires a settling period after the reset terminates.

**FIFO Locks Up**

Short noise pulses on the FIFO’s master reset pin can cause the FIFO to be “partially reset” because of which the FIFO may not respond. If this problem occurs, you need to terminate the master reset line.

**Missing or Disappearing Data**

Glitches on the $R$ input can cause data to disappear because of an unintended read operation. The read increments the internal read counter, resulting in the loss of the current data word. Here again, a termination network eliminates the unwanted glitches.

**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 high-density FIFOs are connected in parallel to make a wider word, certain conditions can cause the FIFOs to choose individually 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 and in the full condition when a write immediately follows a read.

**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. In the general case, the read and write signals are asynchronous. Upon completion of the write operation the internal state of the FIFO goes from empty to empty + 1. During this interval, a read operation might or
might not be recognized. A read 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 aperture 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.

**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.

**Figure 10. Master Reset Timing**

![Master Reset Timing Diagram]

**Waiting at the Empty Boundary**

Figure 11 shows the timing that prevents problems with reads at the empty boundary. Any device reading from the FIFO must wait an amount of time, \( t_{RAE} \), after the termination of the write operation before causing a HIGH-to-LOW transition of the \( R \) signal. The \( W \) signal's rising edge indicates the termination of the write operation.

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

**Waiting at the Full Boundary**

Figure 12 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 \( W \) signal. The \( R \) signal's rising edge indicates the end of the read operation.

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

**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.
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 an empty FIFO that is receiving read pulses. Because the FIFO is empty, the read pulses are ignored, and nothing happens. Next, a single word is written into the FIFO, with a signal that is asynchronous to the read pulses, while the read pulses continue. The internal state machine in the FIFO goes from empty to empty + 1 shortly after the rising edge of the write pulse. However, it does this asynchronously with respect to the read pulse, and it does not look at the read signal until it enters the empty + 1 state. If the rising edge of the write signal 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.

Intermittent Malfunctions
If all the timing requirements appear to be met and data in the FIFO is still corrupted, the cause is likely to be noise on the power supply. Non-repeatable failures may be caused due to random spikes on either the VCC or ground pins of the FIFO.

The solution for this problem is to add a high-pass filter capacitor between the device’s power and ground pins. This practice is recommended whenever the read or write frequency exceeds 5 MHz. Use a very small (100–500 pF)
ceramic or mica capacitor. Surface-mounted capacitors are recommended because they have at least an order of magnitude less lead inductance than radial or axial leaded capacitors.

The filter capacitor is in addition to the 0.1 or 0.01 µF decoupling capacitor that should always be present with any high-speed digital chip. Although decoupling capacitors are often referred to as bypass capacitors—implying filtering properties—their true function is to supply the instantaneous current required when many or all device outputs simultaneously switch from LOW to HIGH. This larger capacitor thus decouples or isolates the IC from the power distribution system.
## Document History

Document Title: Understanding Asynchronous FIFOs - AN1044  
Document Number: 001-25919

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Orig. of Change</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>1420883</td>
<td>ADMU</td>
<td>09/12/2007</td>
<td>New Spec.</td>
</tr>
<tr>
<td>*A</td>
<td>3082092</td>
<td>ADMU</td>
<td>11/09/2010</td>
<td>Removed information about parts which are obsolete. Updated in new template.</td>
</tr>
<tr>
<td>*B</td>
<td>3181540</td>
<td>ADMU</td>
<td>02/24/2011</td>
<td>Modified the title and the abstract.</td>
</tr>
<tr>
<td>*C</td>
<td>3406989</td>
<td>ADMU</td>
<td>11/04/2011</td>
<td>Modified content and sequencing of topics. Added Figure 1 (FIFO Block Diagram).</td>
</tr>
</tbody>
</table>
# Understanding Asynchronous FIFOs

## 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

<table>
<thead>
<tr>
<th>Category</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automotive</td>
<td>cypress.com/go/automotive</td>
</tr>
<tr>
<td>Clocks &amp; Buffers</td>
<td>cypress.com/go/clocks</td>
</tr>
<tr>
<td>Interface</td>
<td>cypress.com/go/interface</td>
</tr>
<tr>
<td>Lighting &amp; Power Control</td>
<td>cypress.com/go/powerpsoc</td>
</tr>
<tr>
<td></td>
<td>cypress.com/go/plc</td>
</tr>
<tr>
<td></td>
<td>cypress.com/go/memory</td>
</tr>
</tbody>
</table>

### PSoC® Solutions

- PSoC 1
- PSoC 3
- PSoC 5

- Cypress Developer Community
  - Community
  - Forums
  - Blogs
  - Video
  - Training

## Memory

<table>
<thead>
<tr>
<th>Category</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical Navigation Sensors</td>
<td>cypress.com/go/ons</td>
</tr>
<tr>
<td>PSoC</td>
<td>cypress.com/go/psoc</td>
</tr>
<tr>
<td>Touch Sensing</td>
<td>cypress.com/go/touch</td>
</tr>
<tr>
<td>USB Controllers</td>
<td>cypress.com/go/usb</td>
</tr>
<tr>
<td>Wireless/RF</td>
<td>cypress.com/go/wireless</td>
</tr>
</tbody>
</table>

Xx and xx are registered trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

---

© Cypress Semiconductor Corporation, 2007-2011. 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 licensees 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.