# Understanding Large FIFOs

## Introduction

This application note explains the internal operation of the large FIFOs manufactured by Cypress and shows how to use the devices to accomplish depth and width expansion. Other topics covered here include FIFO interfacing, the writing and reading process, failure modes, and typical problem symptoms and solutions. This information applies to the following Cypress FIFOs: CY7C419, CY7C420, CY7C421, CY7C424, CY7C425, CY7C428, CY7C429, CY7C432, CY7C433, CY7C439, CY7C460, CY7C462, CY7C464, CY7C470, CY7C472, and CY7C474.

Timing parameters given in this application note are taken from Cypress Semiconductor's *High Performance Data Book*.

## Large FIFO Overview

The Cypress product line of large FIFOs include densities from 256 x 9 up to 32, 768 (32K) x 9,with the depth doubling (256, 512, 1K, 2K, 4K, 8K, 16K, 32K) between densities. These monolithic devices are available in a wide variety of packages with the industry standard pinout and with access times as fast as ten nanoseconds and cycle times as fast as twenty nanoseconds. Not all speed grades are available in all densities or all packages, so consult the Cypress databook to determine valid speed, density, package combinations. The smallest package available is the 32-lead 7mm x 7mm 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 large FIFOs employ an SRAM type of interface. Data is written into and read out of the devices, 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. This cell makes use of separate read and write transistors to allow independent R/W operation.

Operating these FIFOs at their maximum throughput rates demands the generation of narrow write and read pulses. To facilitate significantly higher throughput rates, Cypress has developed the CY7C440 and CY7C450 families of clocked, or self-timed FIFOs.

These FIFOs feature 70-MHz operation and are characterized by self-timed interfaces. You generate the read and write enables, which are combined internally with the appropriate clocks. Thus, you do not need to generate narrow read and write pulses. These FIFOs also feature totally independent, asynchronous, read and write operations.

Each 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.

## Writing to and Reading from the FIFO

*Figure 1* shows the large FIFOs' read and write timing. Reads and writes are asynchronous to each other. The read process begins with $\overline{R}$'s falling edge. The output data bus, Q0 − Q8, leaves the high-impedance state $t_{LZR}$ ns after $\overline{R}$'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. $\overline{R}$'s rising edge ends the read process.

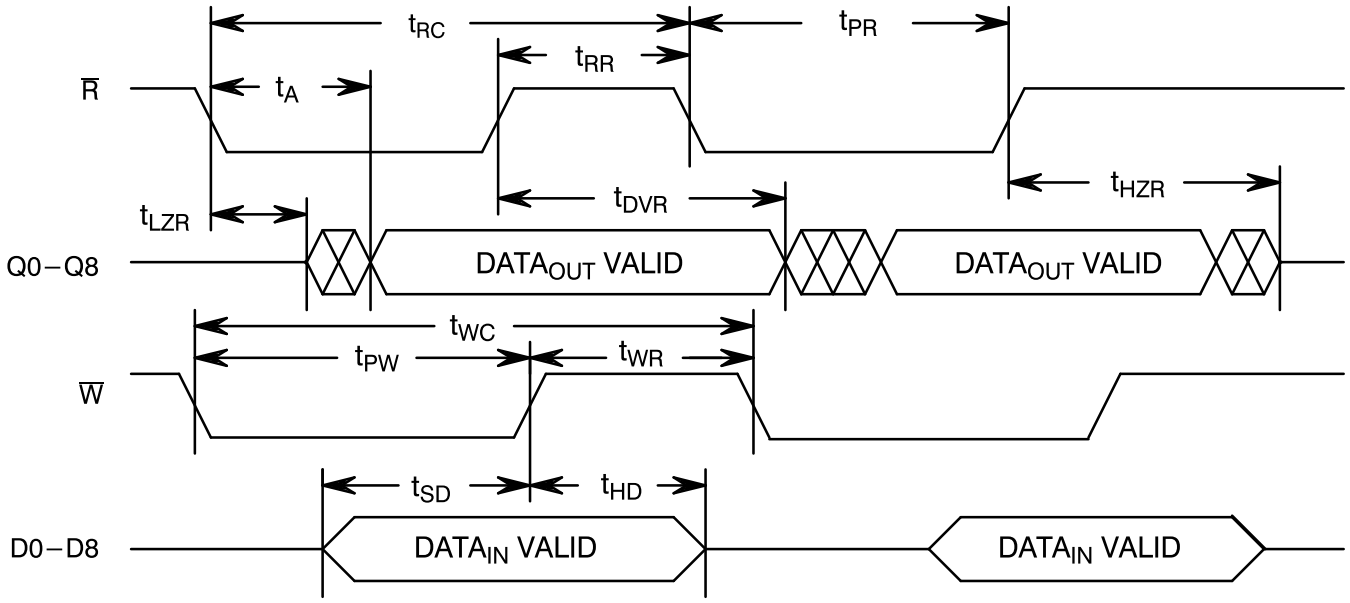


**Figure 1. Asynchronous Read and Write Timing**

The data on the Q0 − Q8 bus remains valid for $t_{DVR}$ ns following the $\overline{R}$ rising edge. This is the output data hold time at the end of the read cycle. The internal circuitry then readies 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, $\overline{W}$, and terminates with $\overline{W}$'s rising edge. For a valid write to occur, the input data bus, D0 − D8, must be stable for $t_{SD}$ ns prior to $\overline{W}$'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 20-ns write strobe width and a 10-ns write recovery time yields a 30-ns write cycle time, or a 33.3-MHz maximum write cycle frequency.

The FIFOs include separate write and read counters (pointers). Each write or read operation increments the appropriate counter one position. When the FIFO is empty, 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.

## 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

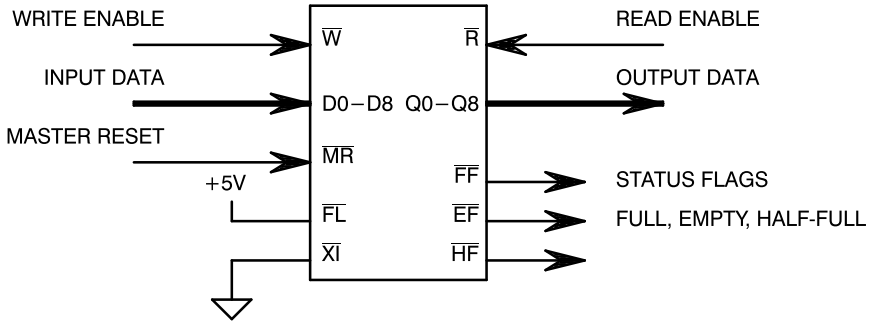


**Figure 2. Standalone Operation**

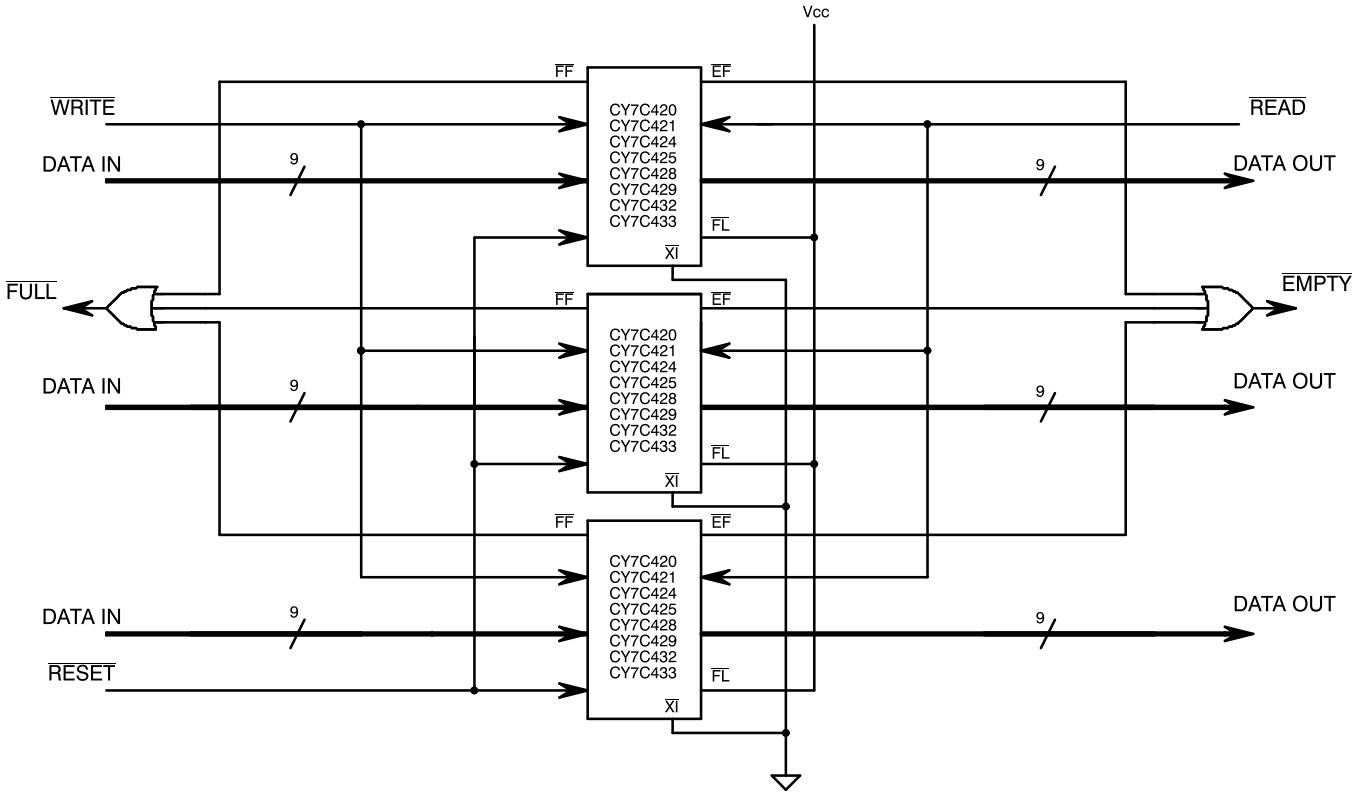


**Figure 3. Width Expansion**

- 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 FIFO Configurations

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

*Figure 2* illustrates standalone mode, and *Figure 3* shows width expansion mode. In both these modes,

the $\overline{XI}$ (expansion in) pin is grounded and the FL (first load) pin is tied HIGH.

The OR gates in the width-expansion design generate composite full, half-full, and empty flags (F, HF, E). 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. A composite flag properly reflects the instantaneous status of the entire word.

*Figure 4* illustrates depth expansion. The $\overline{FL}$ (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 $\overline{XO}$ (expansion out) output pin to the next device's $\overline{XI}$ (expansion in) input. The $\overline{XO}$ of the last device in the chain is connected to the $\overline{XI}$ of the first device, thus forming a token-passing ring.

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 $\overline{XO}-\overline{XI}$ 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 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 count-

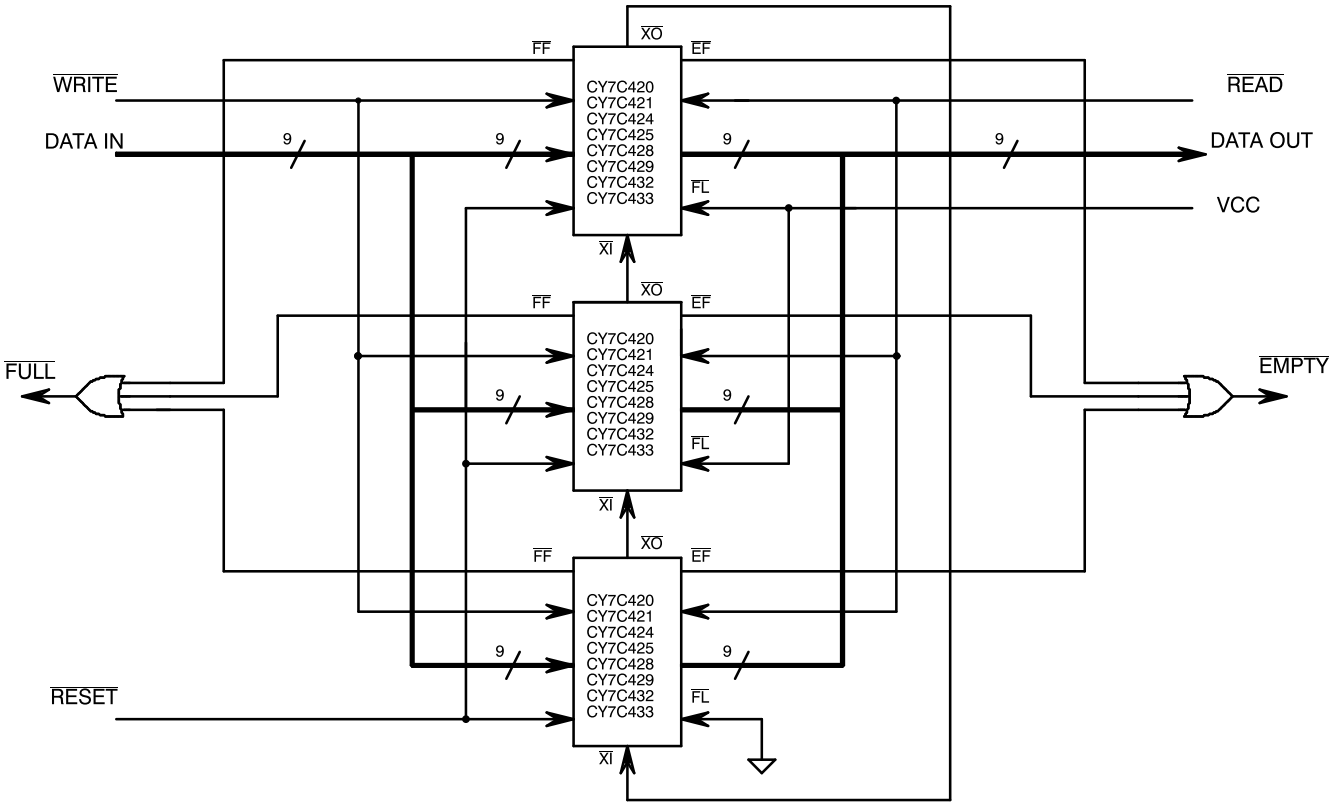


**Figure 4. Depth Expansion**

ers determine a device's status and whether it can accept data though a write operation. *Figure 5* 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 $\overline{XO}-\overline{XI}$ interface. *Figure 6* 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 ap-

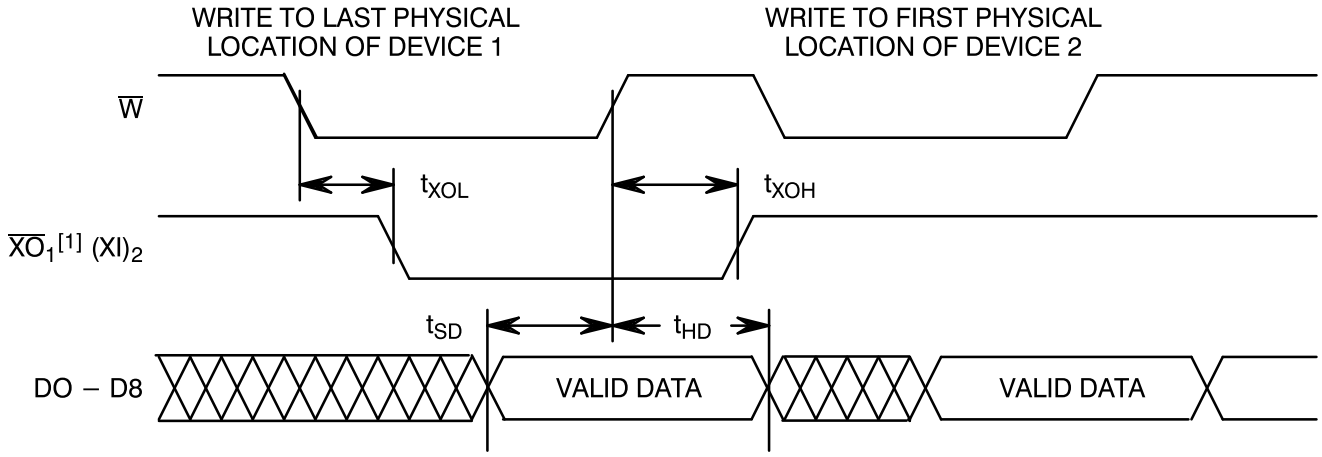


**Figure 5. Write Expansion Timing**

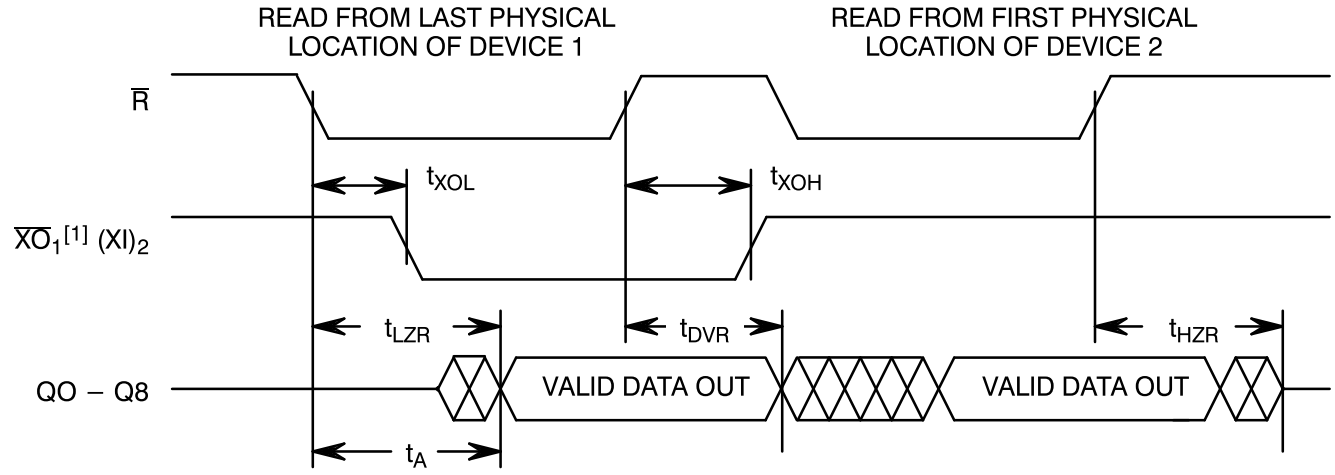


**Figure 6. Read Expansion Timing**

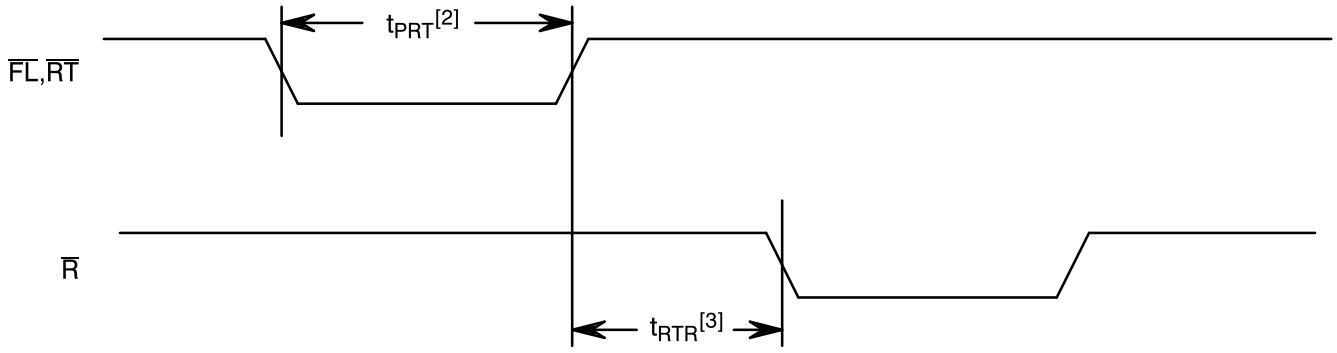


**Figure 7. Retransmit Timing**

plications 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, such those from IDT, which do not allow use of the retransmit function when the FIFO is full.

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 single-device 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 $\overline{R}$ 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, though the retransmit timing constraints shown in *Figure 7* must not be violated.

Note that the architectural description in the 1990 and previous Cypress data books incorrectly stated that the $\overline{W}$ input must be inactive during a retransmit cycle. 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.

The reasons for the data book's retransmit/write restriction are more historical and application-oriented than functional. Specifically, the first large FIFOs did not permit writes during a retransmit cycle. This set a documentation precedent that all future devices had to match.

Additionally, keeping track of what data is currently in the FIFO and what data is being read out can become complicated. For example, if a FIFO is half full and the retransmit function is activated and writes continue, filling the FIFO to three quarters full before the read pointer catches up with the write pointer, the FIFO outputs all of the data.

## 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 of corrupted and repetitive data being present in a FIFO is a spurious active signal (glitch) on the FIFO's $\overline{W}$ input. Because Cypress devices are extremely fast, a write pulse as short as 3 ns initiates a write. Write glitches cause whatever logic levels are 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 $\overline{W}$ and $\overline{R}$ 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 $\overline{R}$ and $\overline{W}$ 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 50Ω 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 resistor be connected from the read or write pin to ground (*Figure 8*). 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 "Systems Design Considerations When Using Cypress CMOS Circuits" application note for help in determining the maximum line length.

FIFO data corruption can also be caused by violation of master-reset timing constraints. As shown in the timing diagram in *Figure 9*, the read and write signals must be inactive around the rising edge of $\overline{MR}$ (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.

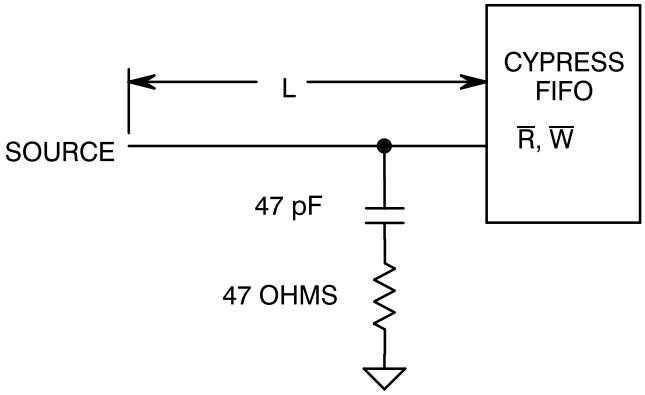


**Figure 8. Recommended Termination Network**

### FIFO Locks Up

Short noise pulses on the FIFO's master reset pin can cause the FIFO to not respond because it is "partially reset." If this problem occurs, you need to terminate the master reset line.

### Missing or Disappearing Data

Glitches on the $\overline{R}$ input can cause data to disappear because of an unintended read operation. The read increments the internal read counter, resulting in

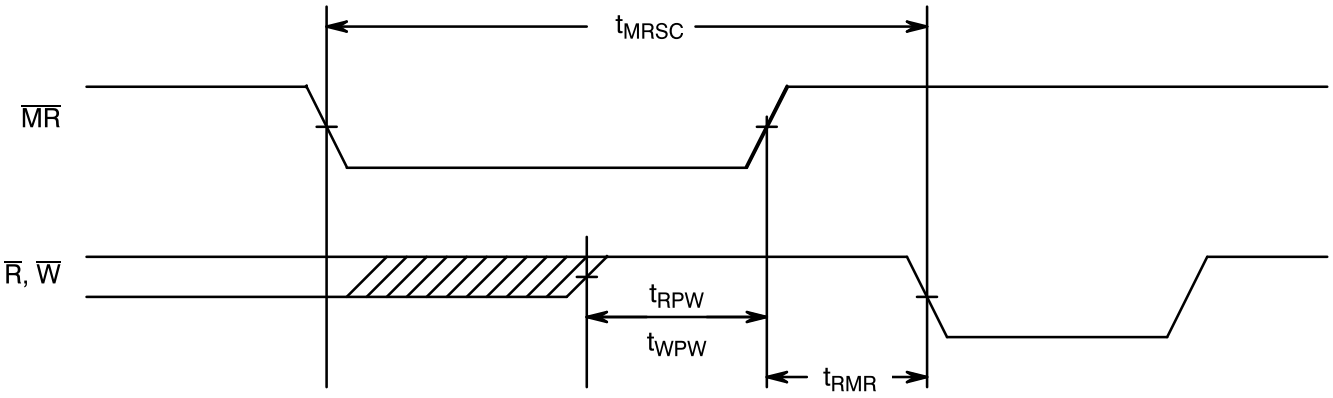


**Figure 9. Master Reset Timing**

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.

**Waiting at the Empty Boundary**

*Figure 10* 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 $\overline{R}$ signal. The $\overline{W}$ 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 ($\overline{EF}$) 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.

**Waiting at the Full Boundary**

*Figure 11* 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.

You can meet this timing 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.

**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.

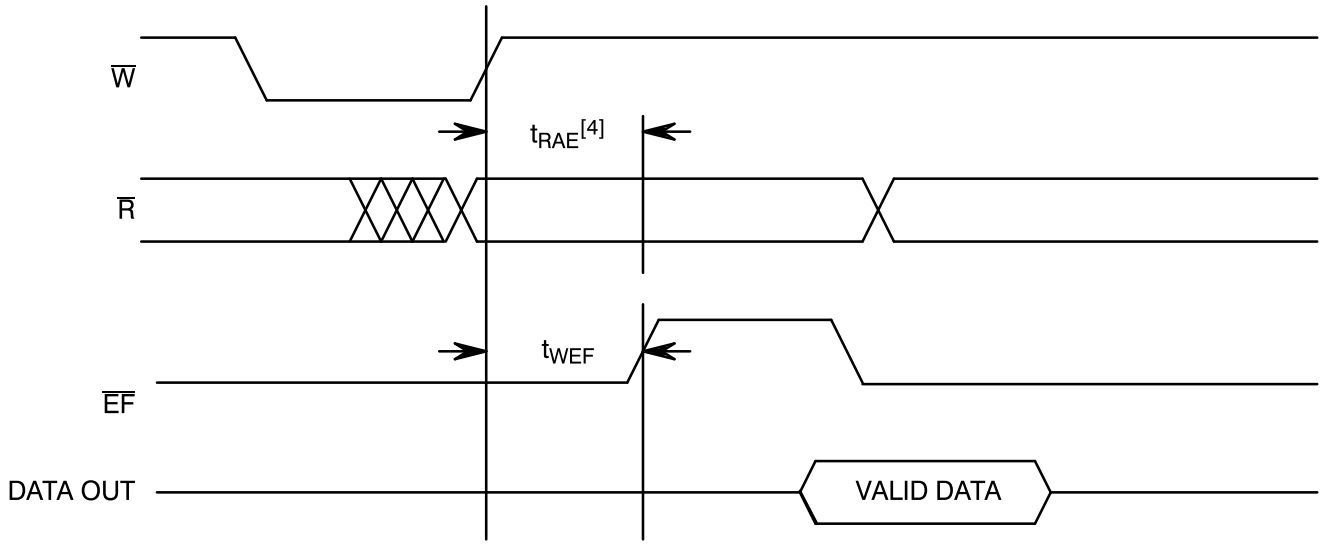


**Figure 10. Read Fall-Through Timing Violation**

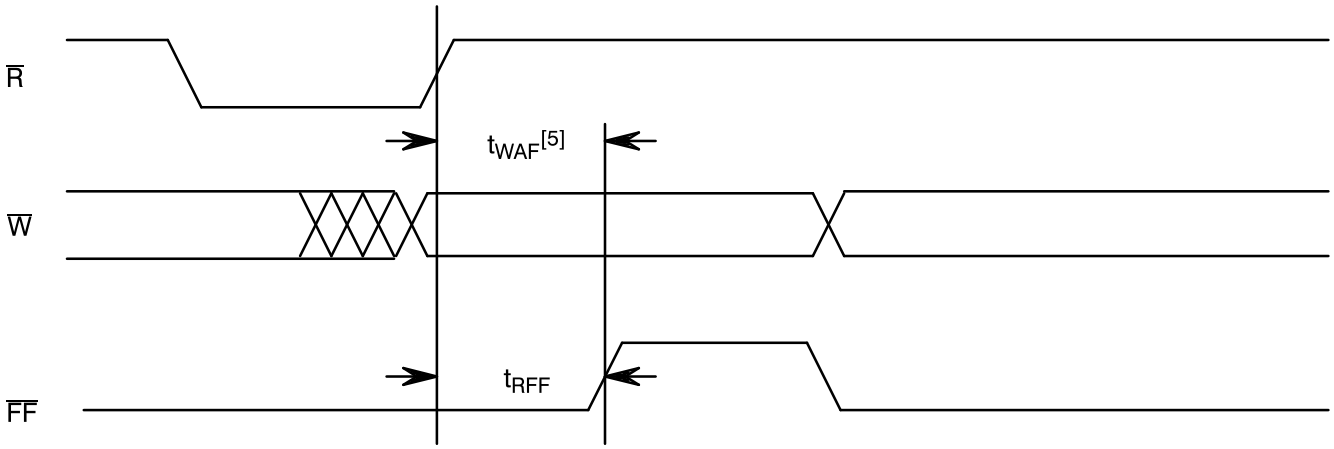


**Figure 11. Write Bubble-Through Timing Violation**

**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.

In a similar manner, the minimum write pulse width 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. Random spikes on either the $V_{CC}$ or ground pins of the FIFO are likely culprits when non-repeatable failures occur.

The cure 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.

## Notes

1. Expansion out of device 1 ($XO_1$) is connected to expansion in of device 2 ($XI_2$).
2. $t_{PRT}$ is the minimum retransmit pulse width.
3. $t_{RTR}$ is the retransmit recovery time. It is a timing window that must not be violated.
4. $t_{RAE}$ is an invalid read window. A read operation should never be initiated inside this window.
5. $t_{WAF}$ is an invalid write window. A write operation should never be initiated inside this window.