Understanding Clocked FIFOs

Introduction

This application note explains the basic operations and features of Cypress clocked FIFO memories. Cypress clocked FIFOs are ideally suited for applications requiring high data throughput and asynchronous data buffering. The clocked FIFO interface simplifies high-speed design and provides greater noise immunity over industry-standard asynchronous FIFOs.

Design considerations of the clocked FIFO architecture are examined, including proper flag operation and decoding, FIFO boundary operation, and resetting and programming the FIFO. FIFO depth and width expansion are also covered.

The Cypress family of clocked FIFOs are available in several densities with a variety of features. *Table 1* outlines the features of Cypress's clocked FIFOs. The entire clocked FIFO family feature fully asynchronous operation at clock rates of up to 70 MHz in non-depth expansion mode. Clocked FIFOs cascaded for depth expansion can operate at frequencies of up to 50 MHz.

The CY7C441 and CY7C443 feature 512 and 2K word by 9 bit memory arrays, respectively. These FIFOs feature high-speed operation and Empty, AlmostEmpty, and AlmostFull flags, center power and ground pins, and width expandability. Both FIFOs are available in either a 32-pin PLCC/LCC package or a 28-pin DIP package.

The CY7C451 and CY7C453 clocked FIFOs have all of the features of the 7C44X FIFOs plus Full and HalfFull flags, programmable AlmostEmpty and AlmostFull flags, parity generation and parity checking, output enable (OE), and depth expandability. The 7C451 features a 512 word by 9 bit memory array and the 7C453 features a 2K word by 9 bit memory array. Both FIFOs are available in either a 32-pin PLCC/LCC package or a 32-pin DIP package.

Table 1. Features of Cypress Clocked FIFOs

FIFO	Density	Speed	Flag Architecture	Parity	Output Enable	Depth Expandable	Width Expandable
7C441	512 x 9	71.4 MHz	Synchronous	No	No	No	Yes
7C443	2048 x 9	71.4 MHz	Synchronous	No	No	No	Yes
7C451	512 x 9	71.4 MHz	Synchronous, Programmable	Programmable	Yes	Yes*	Yes
7C453	2048 x 9	71.4 MHz	Synchronous, Programmable	Programmable	Yes	Yes*	Yes
7C455	512 x 18	71.4 MHz	Synchronous, Programmable	Programmable	Yes	Yes*	Yes
7C456	1024 x 18	71.4 MHz	Synchronous, Programmable	Programmable	Yes	Yes*	Yes
7C457	2048 x 18	71.4 MHz	Synchronous, Programmable	Programmable	Yes	Yes*	Yes

* 50 MHz in this mode

Clocked Architecture

The clocked FIFO architecture is designed to achieve maximum performance from FIFO memories while simplifying their use in a system. Timing pulses for the memory array are generated internally from the read and write clocks thus eliminating the need for generating very narrow external read and write pulses.

The read and write ports have separate clock inputs (CKR, CKW), and read and write operations are enabled through separate clock-enable pins ($\overline{ENR}$, $\overline{ENW}$). The read and write clocks can be fully asynchronous. *Figure 1* demonstrates asynchronous reading and writing to a clocked FIFO.

The clocked FIFO interface is ideally suited for state machine control. A state machine can perform reads or writes by simply asserting the respective enable lines LOW. It is not necessary to toggle the enable lines to perform consecutive operations.

FIFO Writes

Figure 2 shows a simplified block diagram of the clocked FIFO data path. The internal write control logic circuitry controls the input register, the write pointer, and the write port of the dual-ported memory array.

This write operation is similar to writing to a standard 377 register. The FIFO input register is clocked by CKW and enabled by $\overline{ENW}$. Data is clocked into the FIFO on the enabled rising edge of CKW. The data is then written into the memory location pointed to by the write pointer, provided the FIFO is not full ($\overline{Full}$ = 1). The write pointer is then incremented. A full FIFO will ignore any attempted write without upsetting the memory array or the flags. The 70-MHz clocked FIFOs have a data and enable set-up time (t_{SD} and t_{SEN}) of 7 ns.

FIFO Reads

The internal read control logic circuitry controls the output register, the read pointer, and the read port of the dual-ported memory array. The output register holds the word that was last read from the FIFO memory array. This register is loaded from the memory array in a manner similar to loading a standard 377 register. The output register is clocked by CKR and enabled by $\overline{ENR}$. Note that the CY7C45X family of clocked FIFOs feature a three-state output register controlled by $\overline{OE}$.

The read pointer points to a word in the memory array. That word is loaded into the output register on the enabled rising edge of CKR, provided the FIFO is not empty ($\overline{Empty}$ = 1), and the read pointer is then incremented. The word is available at the output pins t_A after the clock edge. An empty FIFO will ignore the attempted read and continue to hold the last word in its output register. The set-up time for $\overline{ENR}$ (t_{SEN}) is 7 ns and the data access time (t_A) is 10 ns for a 70-MHz clocked FIFO.

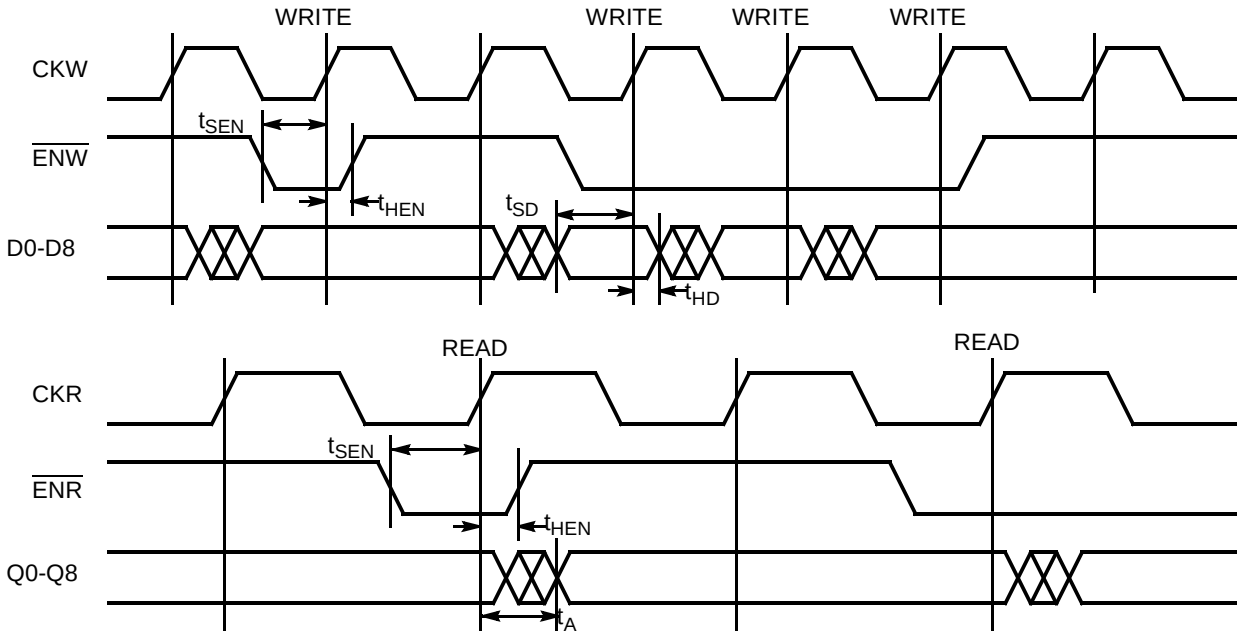

Figure 1. Asynchrous Writing and Reading to a Clocked FIFO

Flag Architecture

Cypress clocked FIFOs feature a synchronous encoded flag architecture that simplifies FIFO integration into a synchronous system. Synchronous flags guarantee that a flag update is only triggered by a rising clock edge. The state of a flag is guaranteed to be valid t_{FD} after the rising clock edge.

Unclocked asynchronous FIFOs can generate narrow flag pulses with indeterminate timing based on the timing relationship of read and write pulses. External flag synchronization

logic is required in synchronous designs using unclocked FIFOs. The Clocked FIFO architecture eliminates these short flag pulses and avoids the need for external flag synchronization logic.

A small package footprint is maintained by encoding the state of the flags. Pin count and package size are reduced and fewer PCB board signals require routing. Only two signals are needed to encode four states of the 7C44X FIFOs and three signals encode six states of the 7C45X FIFOs.

The FIFO flags are easily decoded inside a programmable control unit or a state machine controller. Decoding the signals properly produces flags synchronized to a single clock. *Figures 3* and *4* show a block diagram of the flag architecture for both the 7C44X and 7C45X FIFOs. The diagrams also show the external logic needed to decode and synchronize the flags.

The decoded Empty-type flags are synchronized to the read clock (CKR) and decoded Full-type flags are synchronized to the write clock (CKW). The CY7C45X family of Clocked FIFOs features a Programmable Almost Full/Empty flag ($\overline{PAFE}$) that is synchronized to the read and write clocks.

Reads and Writes with Boundary Flags

The Empty and Full flags are considered Boundary flags because they indicate that the FIFO has reached its boundary of operation. Attention must be paid to the status of these flags when operating the FIFO at or near the boundaries. The internal FIFO write and read control logic uses the Boundary flags to determine if an access to the memory array is possible. The internal write control logic will not attempt to write to the memory array or increment the write pointer if the FIFO is Full, as indicated by the registered $\overline{Full}$ flag. Similarly, the read control logic will not load the output register or increment the Read Pointer if the FIFO is empty, as indicated by the registered $\overline{Empty}$ flag (see *Figures 2*, *3*, and *4*).

The boundary flags determine the state of the read and write logic control circuits inside the Clocked FIFO. Design considerations with boundary flags are explored in the next two sections.

Boundary Latency Cycles

A write or a read can cause the FIFO memory array to exit from an empty- or full-boundary condition. At the empty boundary, the FIFO write control logic will allow an enabled write clock to store a word in the memory array. However, the $\overline{Empty}$ flag synchronization register will not reflect the current state of the FIFO memory array until it is clocked by the read clock. Similarly, at the full boundary, the FIFO read control logic will allow an enabled read clock to remove a word from the memory array, but the $\overline{Full}$ flag synchronization register will not reflect the current state of the FIFO until it is clocked by write clock.

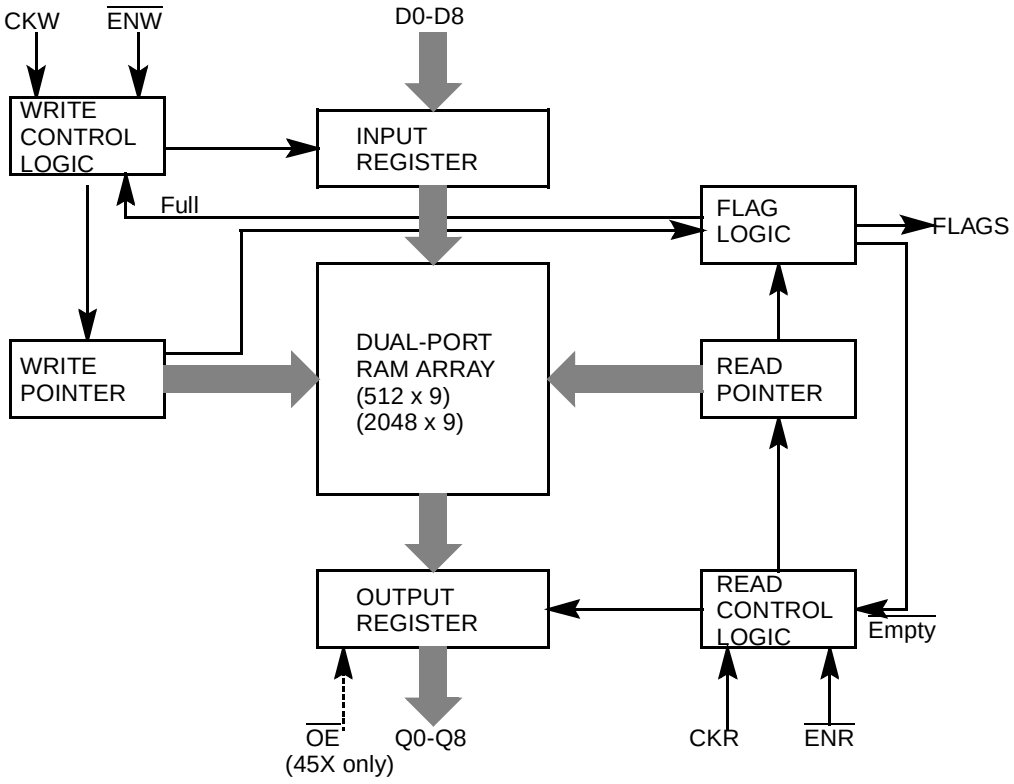

Figure 2. Clocked FIFO Data Path

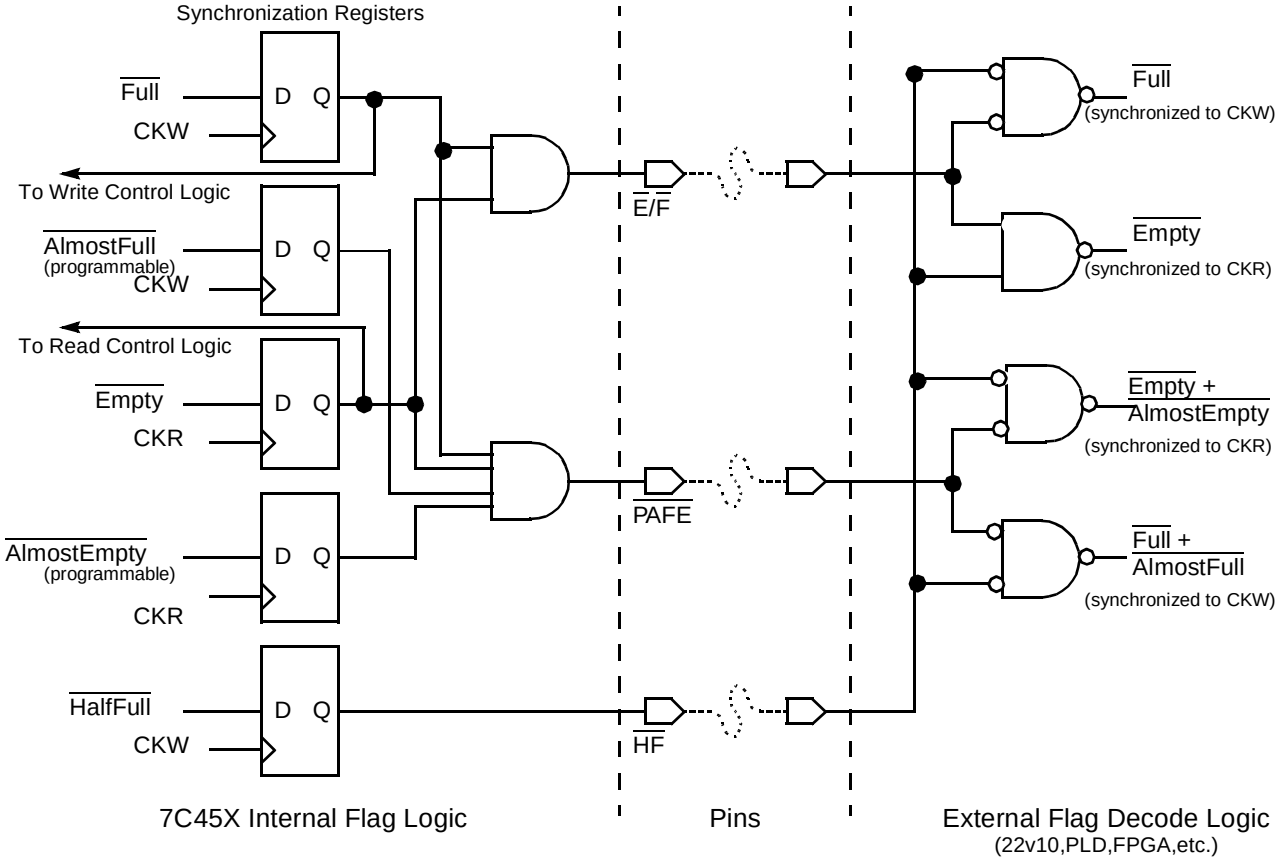

Figure 3. 7C45X Flag Architecture

A FIFO latency cycle (update cycle) refers to the clock cycle that causes a boundary flag register to be updated with the current status of the memory array. During this cycle, only a boundary flag register is updated regardless of the state of ENR or ENW. A read-clock latency cycle updates the Empty flag register from LOW to HIGH regardless of the state of ENR. When the Empty flag register is in the HIGH state, an enabled read clock can retrieve data from the memory array. The overall effect is that after the FIFO memory becomes non-empty, it takes two read cycles to get the first word from the FIFO-one to update the flag and one to read the data.

Similarly, a write clock latency cycle updates the Full flag register from LOW to HIGH regardless of the state of ENW. When the Full flag register is in the HIGH state, an enabled write clock can store data in the memory array. The overall result is that after the FIFO memory becomes non-full, it takes two write cycles to put the first word in the FIFO.

This type of flag operation is desirable because it guarantees that flags in the inactive (HIGH) state will be valid and usable for at least one clock cycle. This architecture eliminates indeterminate short flag pulses characteristic of asynchronous flag architectures.

Free-Running CKR and CKW Clocks

Boundary-operation timing and latency cycles should pose no problem in designs that employ free-running read and write clocks. Free-running clocks insure that flag update cycles will be performed automatically. The flag registers will be constantly updated with the current FIFO status.

Designs that do not use free-running clocks must explicitly issue a clock cycle near the FIFO boundaries in order to update the flag registers. Absence of free-running clocks may decrease system performance by causing the external control circuitry to wait for one clock cycle during the flag update cycle before performing an operation.

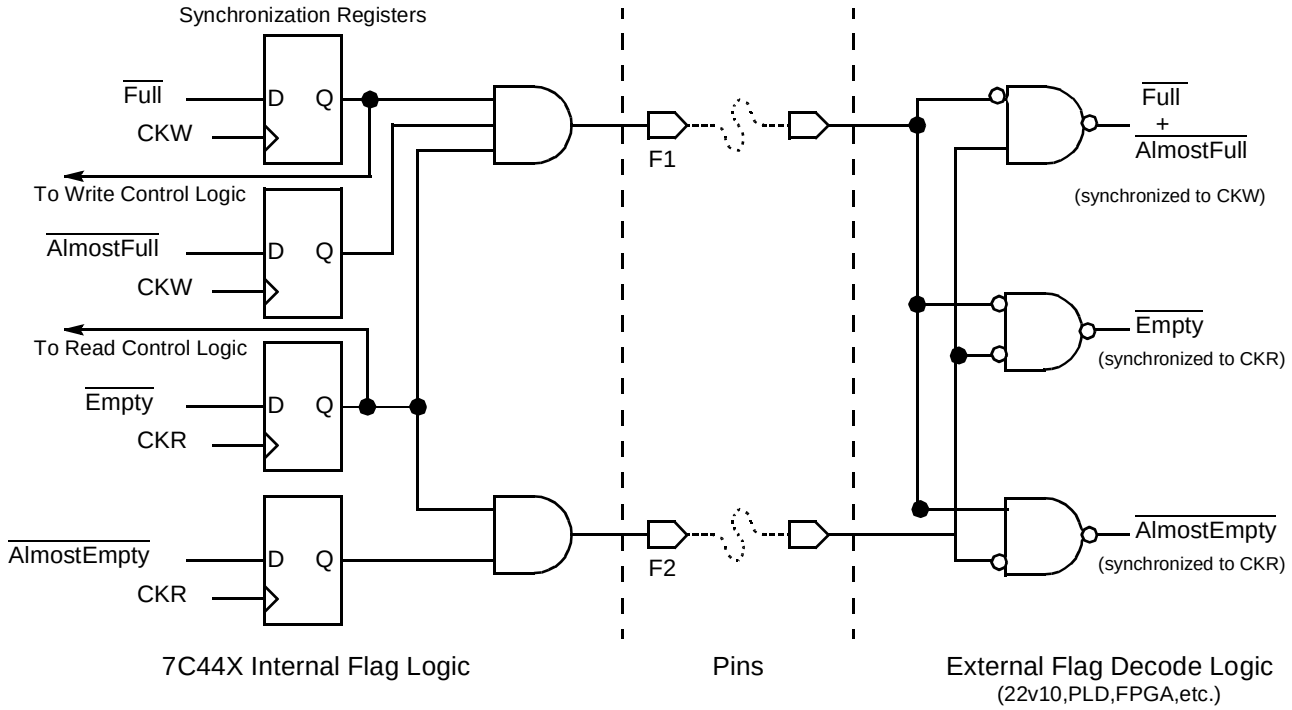

Figure 4. 7C44X Flag Architecture

Resetting and Programming Clocked FIFOs

Master Reset

Clocked FIFOs are reset by pulsing the $\overline{MR}$ (Master Reset) pin LOW. Resetting the FIFO clears the read and write pointers so that they both point to location zero of the memory array, causing the FIFO to be Empty. The data output register will contain all 0s after the reset pulse occurs. Master Reset also resets the internal read and write control logic circuits. The 7C45X family of clocked FIFOs can also be programmed during Master Reset. Programming the FIFO causes the program word to be stored in the FIFO program register.

Clocked FIFOs generate internal timing pulses off of the falling edge of $\overline{MR}$ in order to reset and program the internal FIFO control logic. For this reason, it is very important that the assertion of $\overline{MR}$ be glitch free. A narrow glitch of only a few nanoseconds while $\overline{MR}$ is LOW can be interpreted as a false edge and interrupt the reset timing sequence. As a result, the FIFO will not be fully reset or programmed.

To insure that Master Reset is glitch free, it is recommended that $\overline{MR}$ be driven by a flip-flop. In applications requiring a single Master Reset signal to reset or program multiple FIFOs, the FIFO pin farthest way from the flip-flop may need to be terminated in order to reduce glitches caused by voltage reflections. The need for terminations is a function of trace length, rise time, and PCB characteristics (see “System Design Considerations When Using Cypress CMOS Circuits,” in the *Cypress Semiconductor Applications Handbook*).

The probability of improperly resetting a clocked FIFO due to glitches induced by ground bounce or other sources of noise can be reduced by using a Master Reset pulse that is as short as possible but is greater than t_{PMR}. Long reset pulses increase the chance that noise from somewhere in the system will be coupled to the $\overline{MR}$ pin through the ground plane. *Figure 5* shows a circuit for creating a short $\overline{MR}$ pulse from a long reset pulse. The duration of the $\overline{MR}$ pulse can be increased by adding more delay registers before the AND gate.

The proper reset sequence requires that enabled read and write cycles not be performed during or near the Master Reset pulse. Clock cycles that are not enabled by $\overline{ENR}$ or $\overline{ENW}$ are allowed during Master Reset. To insure that the clocks are disabled, $\overline{ENR}$ and $\overline{ENW}$ should not glitch LOW. Exact timing parameters are given in the data sheet. An easy way to insure that timing restrictions are met with a state machine is to insert pad states (clock enables HIGH) between the last read and write before Master Reset and between the first read and write after Master Reset.

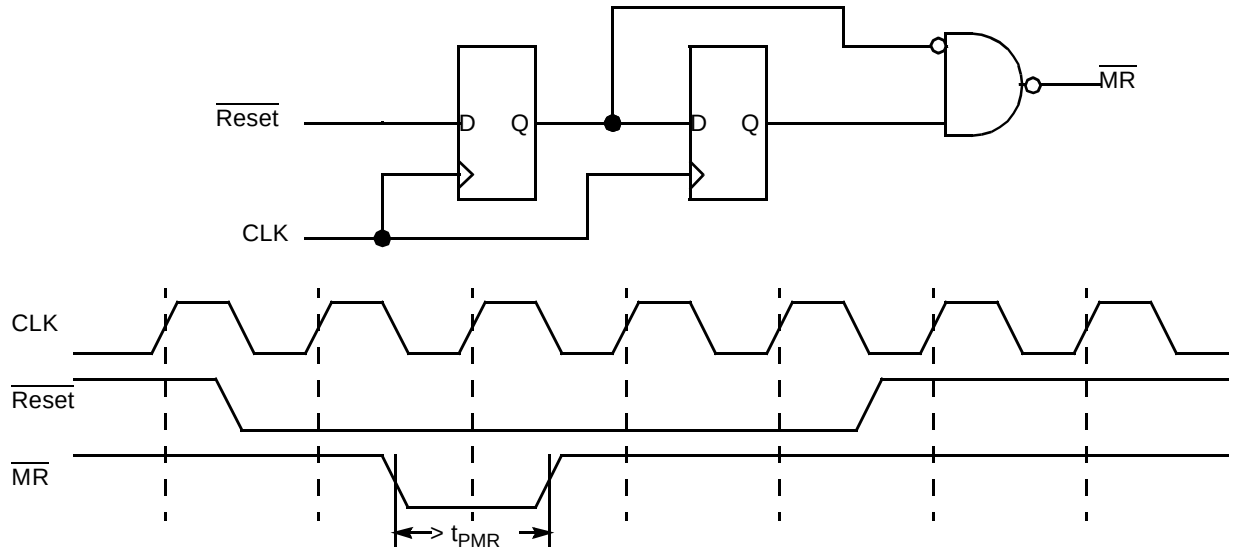

Figure 5. $\overline{MR}$ Pulse Generation

Programming the CY7C45X

The CY7C45X family of clocked FIFOs can be programmed during the Master Reset cycle. Programming affects the AlmostEmpty and AlmostFull flags and sets the Parity. Programming is accomplished by writing data to the FIFO while asserting MR LOW. The program word is stored in the program register. The programming information may be verified my reading the FIFO while MR is still asserted LOW. The FIFO program register is programmed to its default value if no write is performed during a Master Reset.

Data lines D0–D5 are used to program the AlmostEmpty and AlmostFull flags. The value of D0–D5, which is written into the program register, determines the distance from the FIFO boundary flags (Empty and Full) that these flags become active. The distance is programmable in 16-word increments and is determined by 16•P where P is the value of D0–D5. The PAFE pin encodes the programmable flag states.

Data lines D6-D8 program the FIFO parity option. D8 enables the Parity feature when set HIGH. D7 selects between Parity Generation and Parity Checking. Parity Generation is selected when D7 is LOW. D6 selects even parity when set LOW and odd parity when set HIGH.

Parity generation provides a simple means for systems to detect data bit errors. When enabled, the FIFO parity checker will examine bits D0-D7 being written into the FIFO before writing them into the memory array. The ninth bit (D8) will be set according to the parity mode set in the program register. Even-parity mode will set D8 such that the sum off all the bits including D8 is even. Odd-parity mode will set D8 such that the sum is odd. D8 is available on output line Q8/PG/PE during a read from the FIFO. Parity checkers down stream in the system can use D8 to determine when data has been corrupted.

The CY7C45X can be configured as a parity error checker. During a write, data bits D0–D8 are examined before being stored in the memory array. D8 is set LOW if a parity error is detected. When set for even parity checking, a parity error occurs if bits D0–D7 add to an odd number. Odd-parity checking will detect an error if D0–D8 add to an even number. D8 is written into the memory array with the rest of bits D0–D8. The parity-error bit (D8) is then available on Q8/PG/PE during a read from the FIFO.

Depth Expansion

The CY7C45X Family of Clocked FIFOs feature depth expandability. Two or more CY7C45Xs may be cascaded to achieve a single, large FIFO memory array. Depth expansion may be used in applications requiring buffering of large data packets, using extremely disparate read and write rates, or having long read latencies.

Depth expansion is achieved by cascading several FIFOs using the expansion pins. Data is automatically multiplexed from the FIFOs onto a single output bus using the FIFO's three-state output drivers. The flags must be combined to form composite flags. *Figure 6* shows two FIFOs cascaded for depth expansion.

The cascaded devices act as a single FIFO memory array. Read and write control is passed from one FIFO to another using the expansion pins. When a single FIFO has had all of its memory locations written to, it asserts the Expansion Out pin ($\overline{XO}$) signaling the next FIFO to begin writing to its array. Similarly, when the FIFO has had all of its memory locations read from, it deasserts the Expansion Out pin to signal the next FIFO to read data from its array. The FIFOs' expansion pins form a simple token ring.

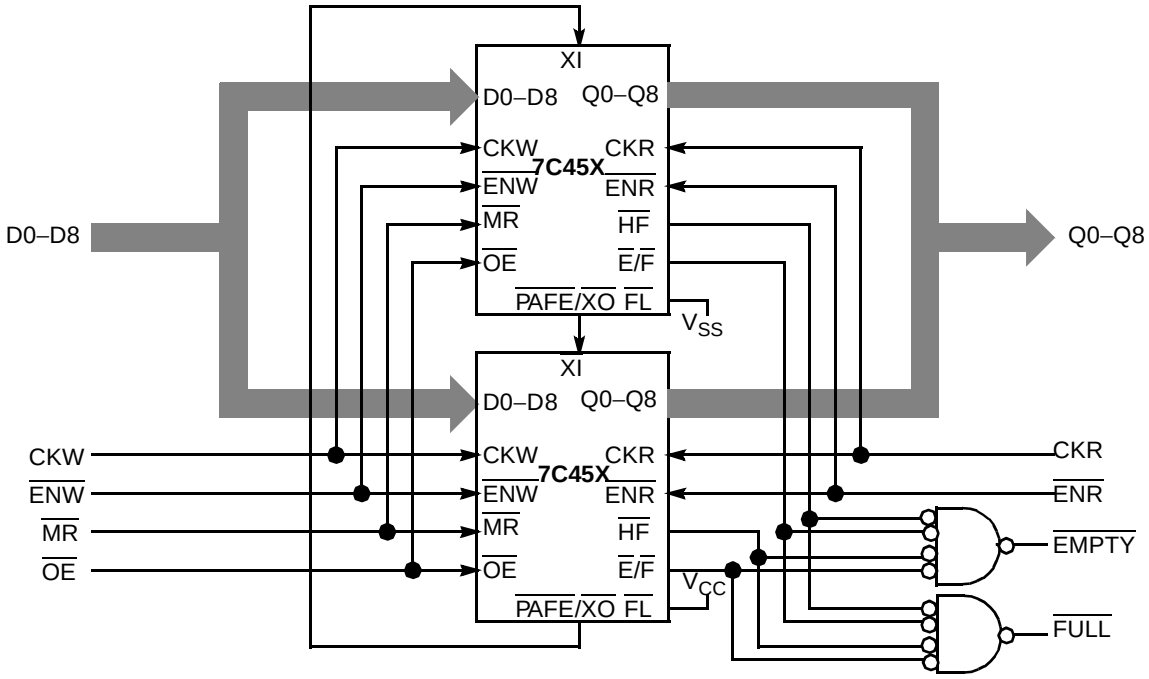

Figure 6. Depth Expansion with CY7C45X

The token-passing architecture necessitates the use of composite flags in order to detect when composite FIFO is in a boundary state (Full or Empty). In a long series of reads and writes, it is difficult to track which of the individual FIFOs possess the read and write tokens. The state of the composite FIFO could be determined by looking at the flags of the FIFOs in possession of these tokens, but this is difficult and unnecessary. Composite flags, shown in *Figure 6*, bypass this problem by looking at all the flags in parallel.

The First Load pin ($\overline{FL}$) indicates which device possesses the read and write tokens following a Master Reset. Only one device should have its $\overline{FL}$ pin tied to V_{SS}. All other devices should tie $\overline{FL}$ to V_{CC}.

The Almost Empty and Almost Full flags are not usable in depth expansion. The cascaded devices, however, can be programmed for parity. All cascaded devices will be programmed the same since all control and data pins are common. Program read occurs automatically on the First Load device only to avoid bus contention.

Width Expansion

Both the CY7C44X and CY7C45X family of Clocked FIFOs can be width expanded for applications requiring data wider than 9 bits.

Width expansion is achieved by wiring the FIFOs in parallel. *Figure 7* shows two FIFOs wired for width expansion. Composite flags should be used to provide proper read and write signaling near the FIFO Empty and Full boundaries. Process variations between FIFOs can result in differences in t_{SKEW1} and t_{SKEW2}. This can cause the update cycles to occur on staggered clock cycles in different FIFOs. Data misalignment can occur at the boundary condition if an operation is performed before all FIFO flag registers are in the same state. Composite flag signaling insures that all FIFOs are in the same state so that an operation at the boundary is performed concurrently by all FIFOs.

The $\overline{PAFE}$ flag from either FIFO may be monitored and will give the correct status, or each FIFO may be programmed differently to give different $\overline{PAFE}$ flags. Parity generation/checking is performed in each device independently according to how they are individually programmed.

Using a Clocked FIFO Like a Standard FIFO

Applications that require high-speed unclocked asynchronous FIFOs memory may use clocked FIFOs. Unclocked asynchronous FIFOs operate at much lower frequencies than clocked FIFOs but feature read and write interfaces driven by single read and write strobes.

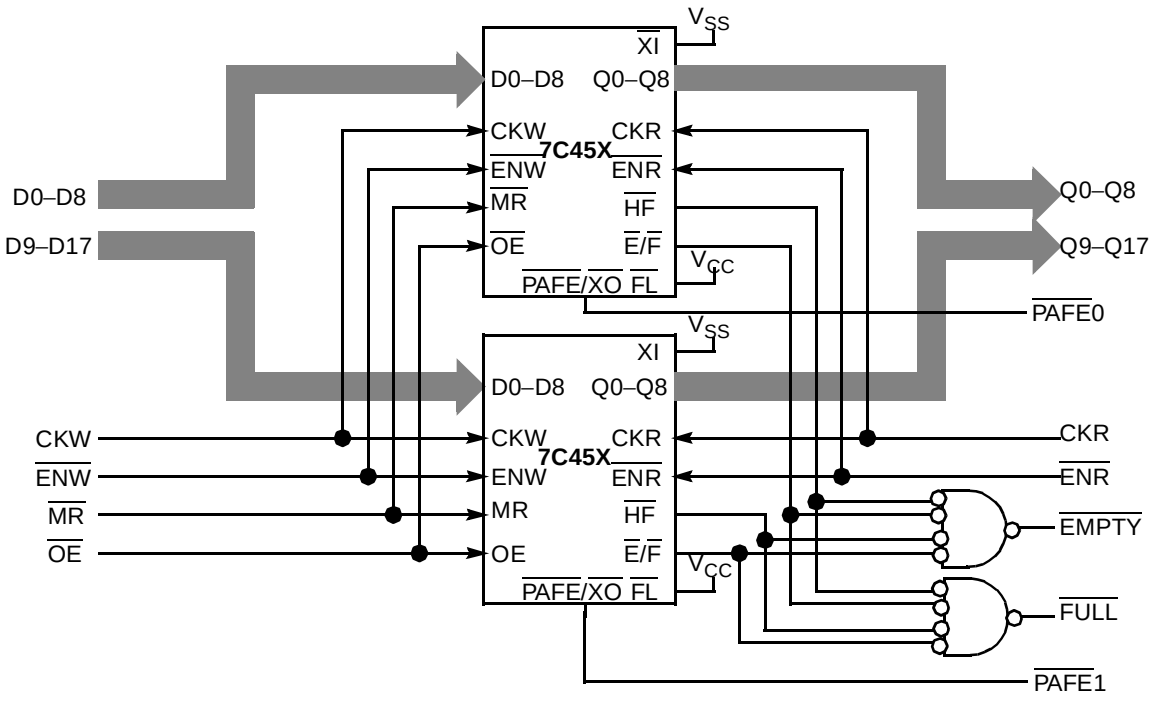

Figure 7. Width Expansion with CY7C45X

Applications can use clocked FIFOs to emulate this operation at high speeds by tying the clock to the appropriate enable line. The enable lines should not be tied straight to ground. Grounding the enable lines directly increases the probability of violating enable set-up times in a noisy environment. Tying the enables to the clocks closes the timing window (when noise can affect the enable pins) and filters out unwanted ground noise. The zero hold-time feature of the enable makes this configuration possible. *Figure 8* shows a CY7C45X configured as a standard FIFO.

A caveat occurs at the boundary condition flag timing. Absence of a free running clock will prevent the flags from being updated. As a consequence, the internal FIFO control logic will inhibit read or write operations if the respective flag is not updated. To avoid this problem, the FIFO in a boundary state must be strobed in order to force a flag update cycle. Data is not destroyed during the update cycle. The desired operation may proceed once the flag is updated correctly.

For example, an empty FIFO with its empty flag asserted is written to by strobing the write port. The empty flag, however, is only updated by the rising edge of CKR. Consequently, the read port must be strobed in order to force the flag to be updated. While the empty flag is asserted, the attempted reads are ignored (data remains in the FIFO) and only serve to update the empty flag. Once the empty flag is deasserted, the data can be read from the FIFO in the normal manner.

It also possible to build a controller that forces an update cycle at the FIFO boundary without checking the state of the flags. When a read or a write strobe occurs affecting the state of the memory array, the controller forces an update automatically by toggle the other strobe line.

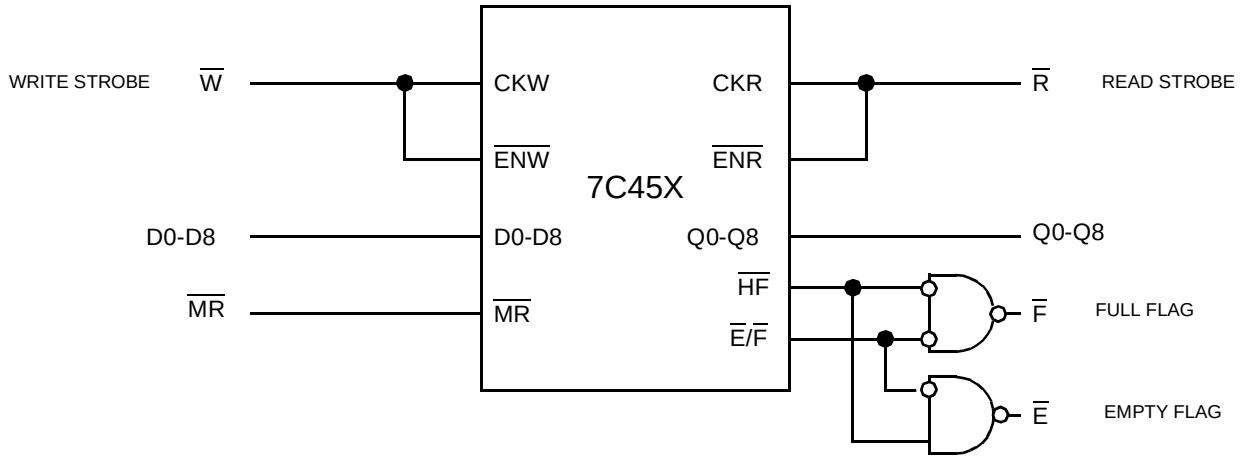

Figure 8. Using a Clocked FIFO Like a Standard FIFO

Conclusion

Cypress CY7C44X and CY7C45X Clocked FIFOs solve a wide variety of data buffering and storage needs for telecommunications, interprocessor, and data gathering applications. The clocked FIFO architecture offers 70-MHz performance and avoids the timing and noise problems inherent in unclocked asynchronous FIFOs.