EMPTY and FULL Flag Behaviors of the Axcelerator FIFO Controller

Introduction

The purpose of this application note is to specifically illustrate the following two behaviors of the FULL and EMPTY flags:

- The Axcelerator and RTAX-S FIFO controller deasserts the EMPTY flag for two read clock cycles immediately after the deassertion of the CLR signal while the read_clock (RCLK) pulse is high.
- The Axcelerator and RTAX-S FIFO controller could falsely assert the FULL and EMPTY flags if separate clock frequencies are used for RCLK and WCLK

For a complete description of the FIFO and FIFO controller functionality, reference the *Axcelerator Family FPGAs* datasheet, the *RTAX-S RadTolerant FPGAs* datasheet and the *Axcelerator Family Memory Blocks* application note.

FIFO Controller

The circuit in Figure 1 depicts the FIFO Controller architecture. Table 1 on page 2 lists and describes the FIFO Controller input and output signals.

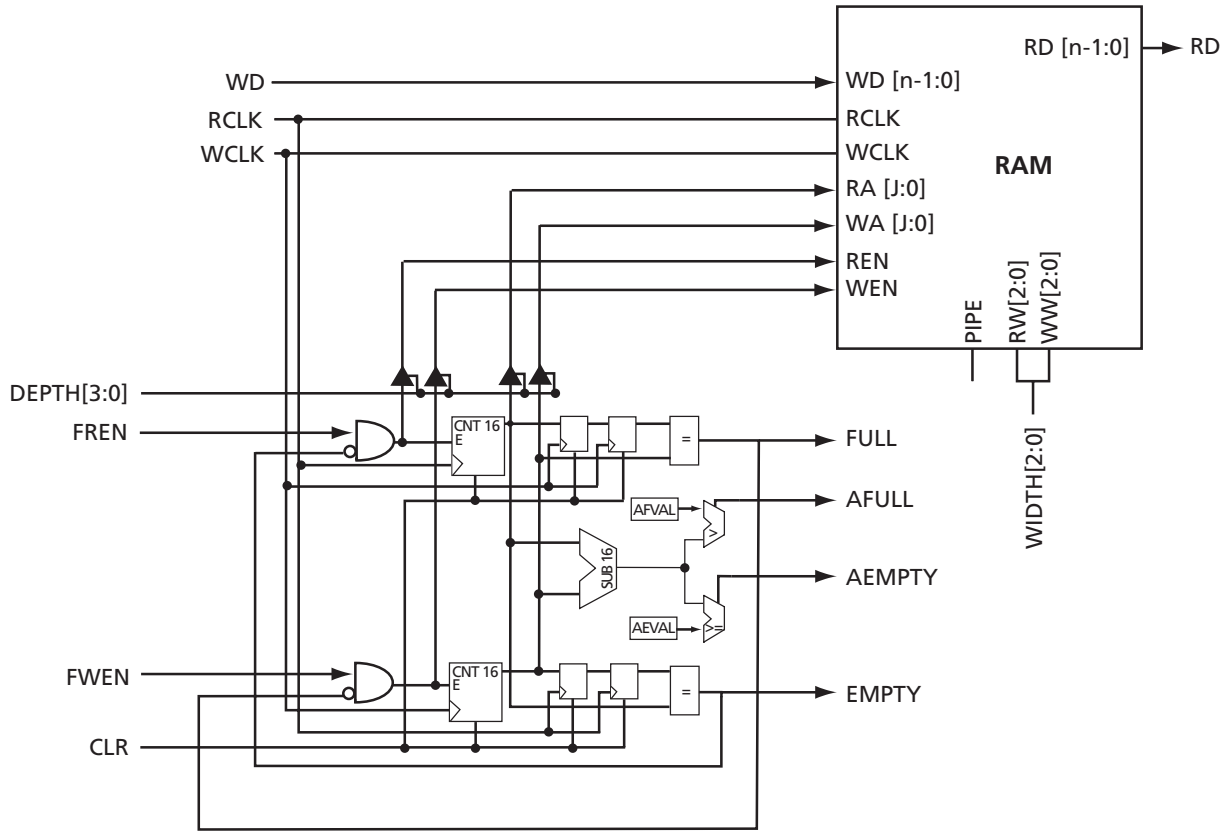

Figure 1 • **Simplified Architecture of FIFO Controller**

Table 1 • **FIFO Signal Description**

Signal	Direction	Description
WCLK	Input	Write clock (active either edge).
RCLK	Input	Read clock (active either edge).
FWEN	Input	FIFO write enable. When this signal is asserted, the WD bus data is latched into the FIFO, and the internal write counters are incremented.
FREN	Input	FIFO read enable.
WD [N-1:0]	Input	Write data bus. The value N is dependent on the RAM configuration and can be 1, 2, 4, 9, 18, or 36.
RD [N-1:0]	Output	Read data bus. The value N is dependent on the RAM configuration and can be 1, 2, 4, 9, 18, or 36.
FULL	Output	Active high signal indicating that the FIFO is FULL. When this signal is set, additional write requests are ignored.
AFULL	Output	Active high signal indicating that the FIFO is AFULL.
AFVAL	Input	8-bit input defining the AFULL value of the FIFO. Should not be set to all zeros.
EMPTY	Output	Empty flag indicating that the FIFO is in the empty state. When this signal is asserted, attempts to read the FIFO will be ignored.
AEMPTY	Output	Active high signal indicating that the FIFO is AEMPTY.
AEVAL	Input	8-bit input defining the almost-empty value of the FIFO.
PIPE	Input	Sets the pipe option on or off.
CLR	Input	Active high asynchronous clear. Asserting this signal initializes the FIFO's read and write addresses to '0'. The flags are initialized as well: FULL = '0'; AFULL = '0'; EMPTY = '1'; and AEMPTY = '1'.
DEPTH	Input	Determines the depth of the FIFO and the number of FIFO blocks to be cascaded.
WIDTH	Input	Determines the width of the data-word / width of the FIFO, and the number of the FIFO blocks to be cascaded.

EMPTY Flag Behavior after Clear

This section illustrates the true EMPTY flag behavior without and with a write operation occurring.

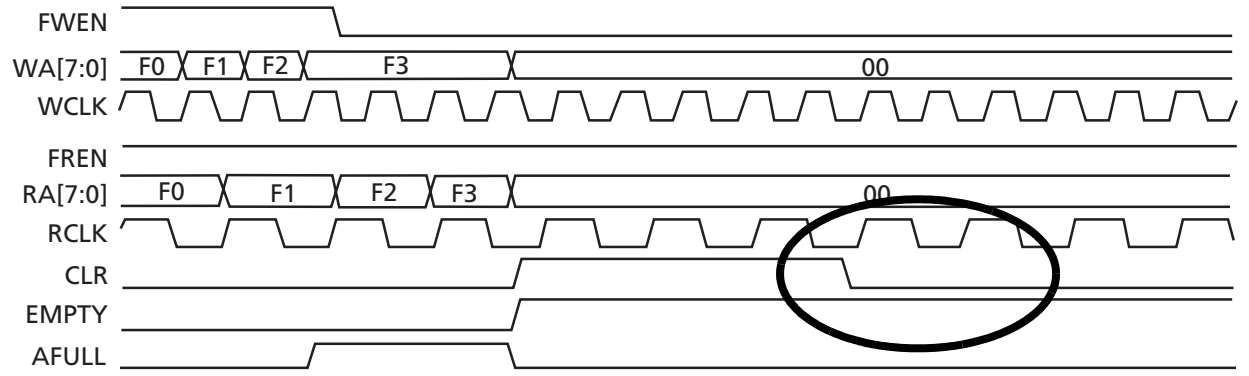

Figure 2 • **EMPTY Flag Behavior after Release of CLR During Low RCLK Pulse without Write Operation**

Figure 2 shows the EMPTY flag remaining high, after the CLR signal is released during the low RCLK pulse, indicating that the FIFO is still in the empty state.

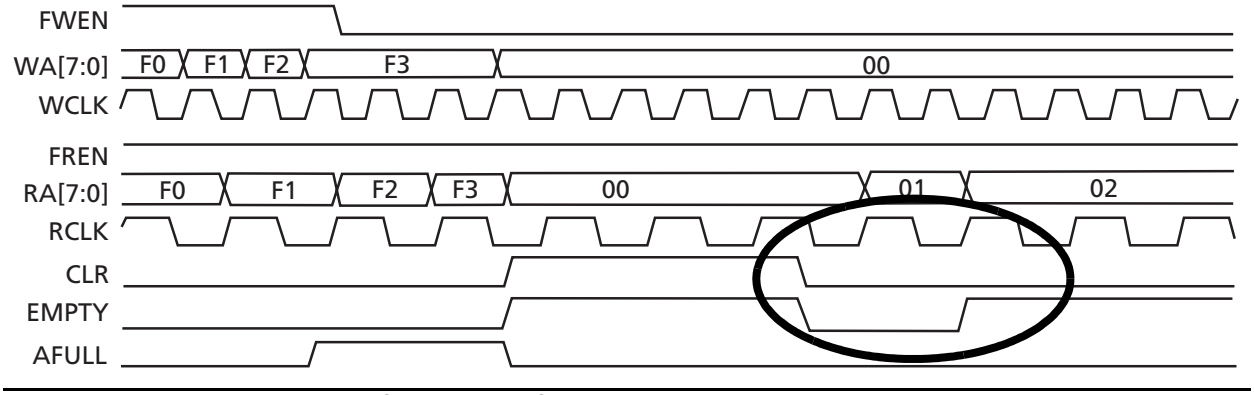

Figure 3 • **EMPTY Flag Behavior after Release of CLR During High RCLK Pulse without Write Operation**

Figure 3 shows the EMPTY flag temporarily going low, after the CLR signal is released during the high RCLK pulse. The EMPTY flag remains low for two rising edges of RCLK, indicating that the FIFO is not in the empty state even though the WA and RA are still in their initialized states.

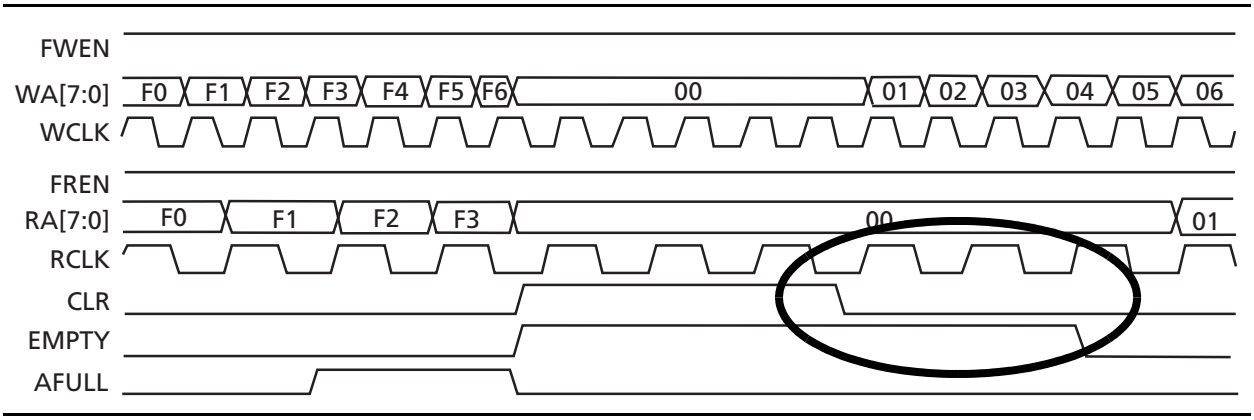

Figure 4 • **EMPTY Flag Behavior after Release of CLR During Low RCLK Pulse with Write Operation**

Figure 4 shows the EMPTY flag remaining high after the CLR signal is released during the low RCLK pulse, indicating that the FIFO is still in the empty state. After three RCLK rising edges the EMPTY flag de-asserts, indicating that the written data is valid and available for reading.

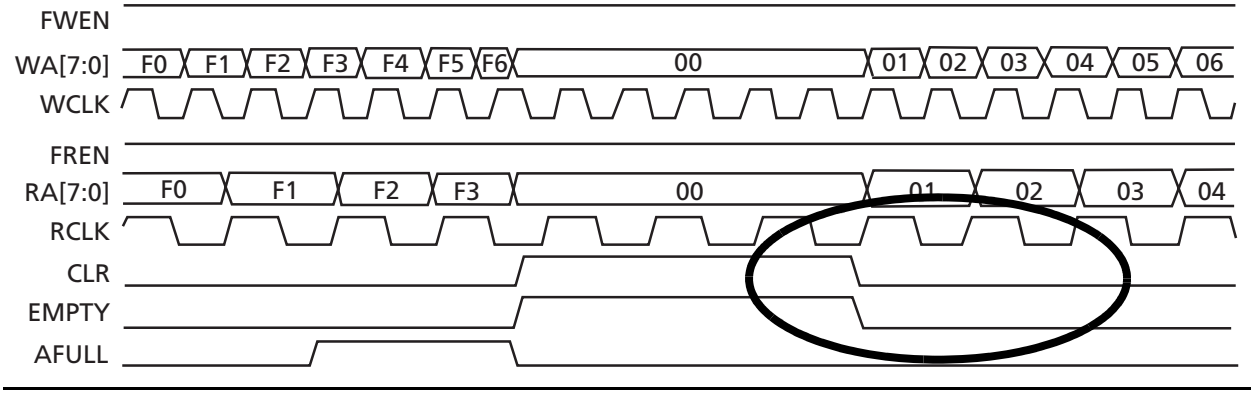

Figure 5 • **EMPTY Flag Behavior after Release of CLR During High RCLK Pulse with Write Operation**

Figure 5 shows the EMPTY flag de-asserting low, immediately after the CLR signal is released during the high RCLK pulse. The EMPTY flag remains low indicating that the FIFO is not in the empty state even though the WA and RA are still in their initialized states. The written data is actually not valid until the third RCLK rising edge. Failure to accommodate this behavior may result in invalid data acquisition.

Simulation Libraries

Prior to Libero® Integrated Design Environment (IDE) and Designer v6.1 Service Pack 1, the Axcelerator and RTAX-S simulation libraries did not accurately reflect the EMPTY flag silicon behavior. The libraries did not model the EMPTY flags temporary low period after the release of CLR during the high RCLK pulse. These simulation libraries have been corrected in Libero IDE and Designer v6.1 SP1 to reflect the true silicon behavior of the EMPTY flag.

Design Solutions

If the design requires an asynchronous release of the CLR signal, the design should only execute a read operation on the third RCLK cycle after the release of CLR and after the first write operation.

As discussed earlier, the EMPTY flag low pulse behavior only occurs if the CLR signal is released during the high pulse of the RCLK. In order to mask this behavior, a simple active-low Latch or negative-edge flip-flop circuit can be implemented in the design. These solutions may be feasible if the design can allow up to

one extra RCLK cycle delay on the final release of the EMPTY signal. A Clock Duplication solution may also be feasible. When implementing these circuits, they should be inserted in the design between the intended CLR signal (called USER_CLR) and the FIFO Controller CLR port (called CLR).

The following solutions are designed for FIFOs configured with rising-edge triggered RCLK. The corresponding waveforms result when the write enable (FWEN) is high. If the FIFO is configured with a RCLK that is falling-edge triggered, the solutions must be modified to maintain a balanced RCLK polarity. Timing analysis should be performed when implementing these solutions.

Latch Solution

If the design requires that the RCLK be driven by a routed clock resource, the simple active-low Latch circuit in Figure 6 can be implemented as a solution. This Latch circuit, which implements the DLP1A macro, is designed to provide a CLR signal that is latched during the low pulse of the RCLK.

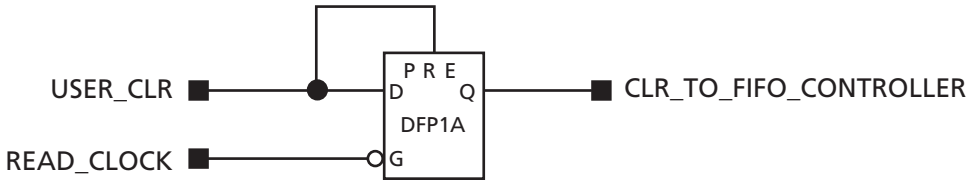

Figure 6 • **Latch Solution**

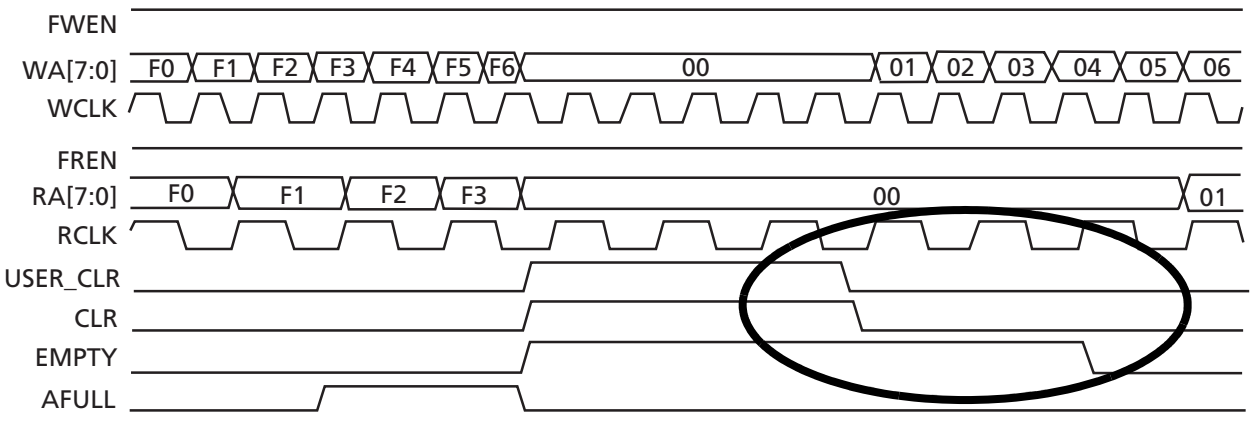

Figure 7 • **EMPTY Flag Behavior after Release of CLR During Low RCLK Pulse with Latch Solution**

Figure 7 shows that when the USER_CLR signal de-asserts during the low RCLK pulse, the latch solution adds only a small propagation delay to the CLR signal that routes to the FIFO controller. Note that if the USER_CLR de-asserts too close to the rising edge of RCLK it will not be latched until the next low pulse of RCLK, which will delay the CLR signal by half an RCLK cycle.

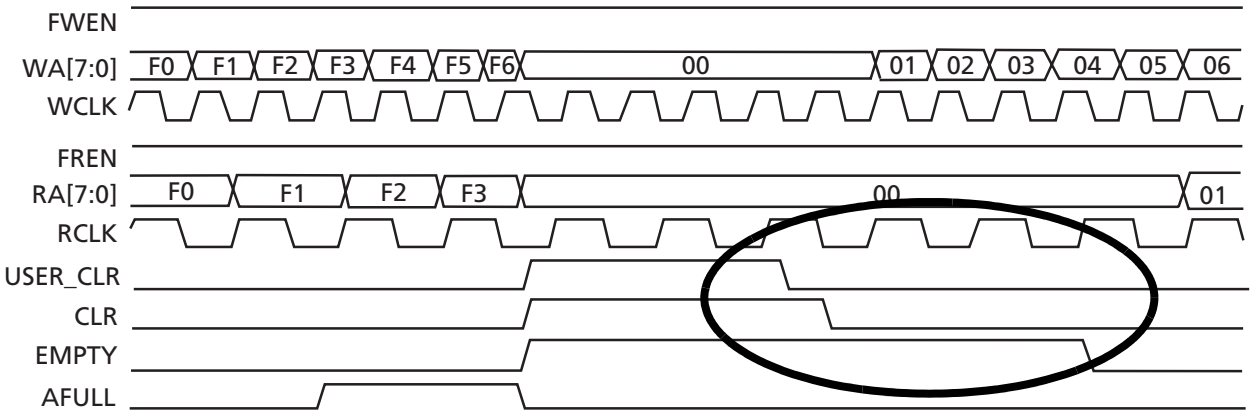

Figure 8 • **EMPTY Flag Behavior after Release of CLR During High RCLK Pulse with Latch Solution**

Figure 8 shows that the USER_CLR signal is not latched in until the low RCLK pulse. Note that in both Figure 7 on page 5 and Figure 8 the EMPTY flag ultimately de-asserts on the third rising edge of RCLK after CLR is de-asserted.

Flip-Flop Solution

If the design requires that the RCLK be driven by a hardwired clock resource, the simple negative-edge Flip-Flop circuit in Figure 9 can be implemented as a solution. A flip-flop is needed because a latch clock signal cannot be driven from a hardwired clock resource. This flip-flop circuit, which implements the DFP1A macro, is designed to provide a CLR signal that was captured during the falling edge of the RCLK. Note that this Flip-Flop may cause the read data to be valid after four RCLK rising edges instead of three.

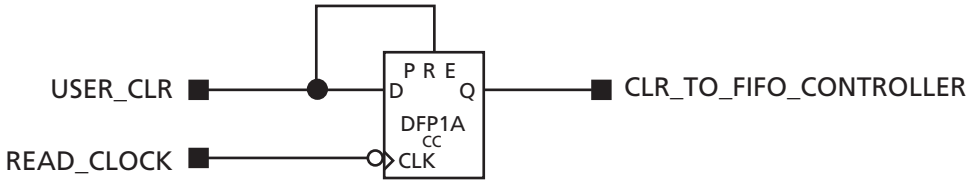

Figure 9 • **Flip-Flop Solution**

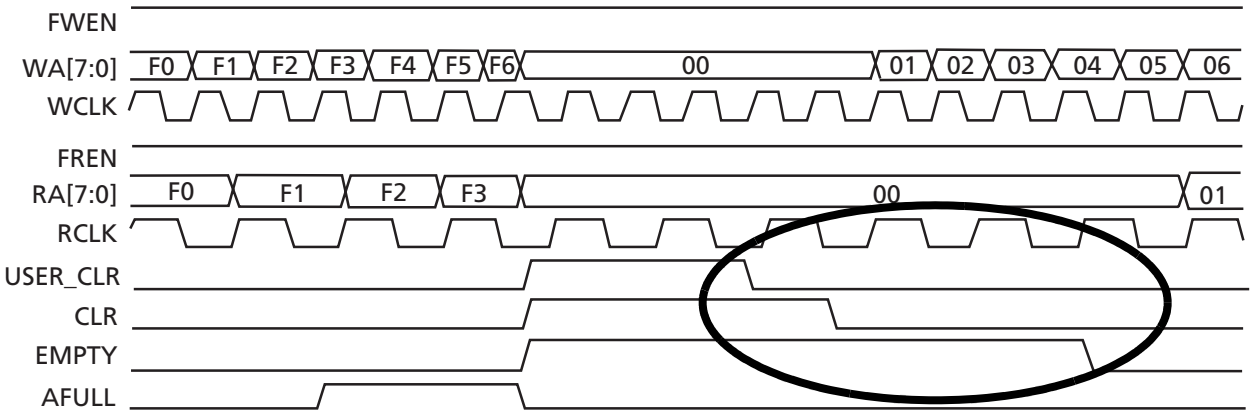

Figure 10 • **EMPTY Flag Behavior after Release of CLR During Low RCLK Pulse with Flip-Flop Solution**

Figure 10 shows that if the USER_CLR de-asserts during the low RCLK pulse, the USER_CLR is not captured until the next falling edge of RCLK. The resulting CLR that drives the FIFO controller is therefore delayed by at least a half cycle of RCLK, causing the EMPTY flag to de-assert after the forth RCLK rising edge from the release of USER_CLR.

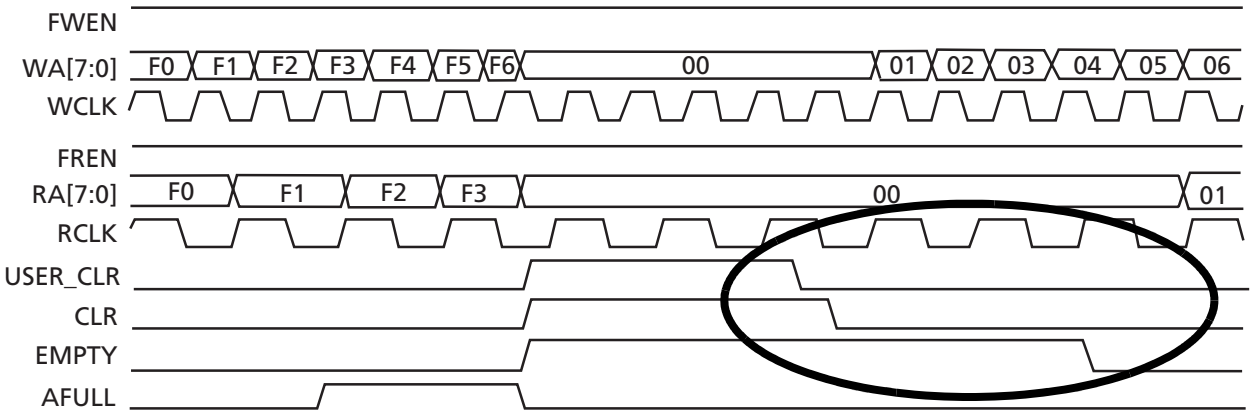

Figure 11 • **EMPTY Flag Behavior after Release of CLR During High RCLK Pulse with Flip-Flop Solution**

Figure 11 shows that the USER_CLR signal is not captured until the low RCLK pulse. Note that in both Figure 10 and Figure 11 the EMPTY flag ultimately de-asserts on the third rising edge of RCLK after CLR is de-asserted.

Clock Duplication and Latch Solution

This circuit is designed to create a copy of the RCLK and use this generated copy to latch in the low level of the CLR signal. This solution can be driven by the routed clock or hardwired clock resource. It implements the DF1, DF1B, DLP1A, XOR2, and INV macros.

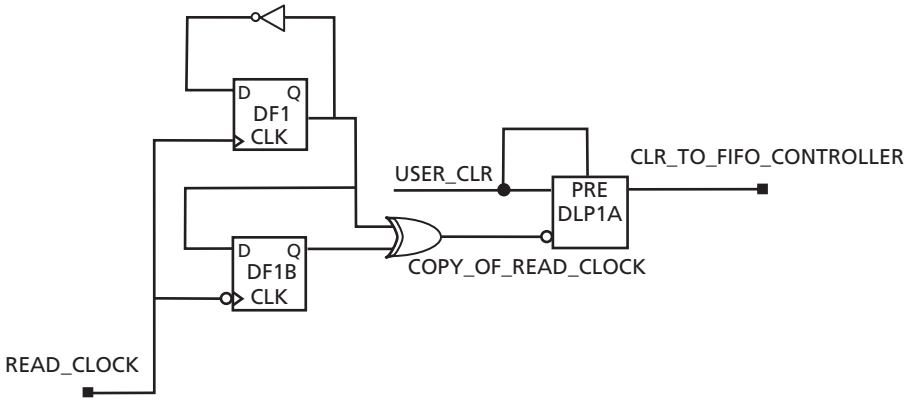

Figure 12 • **Clock Duplication and Latch Solution**

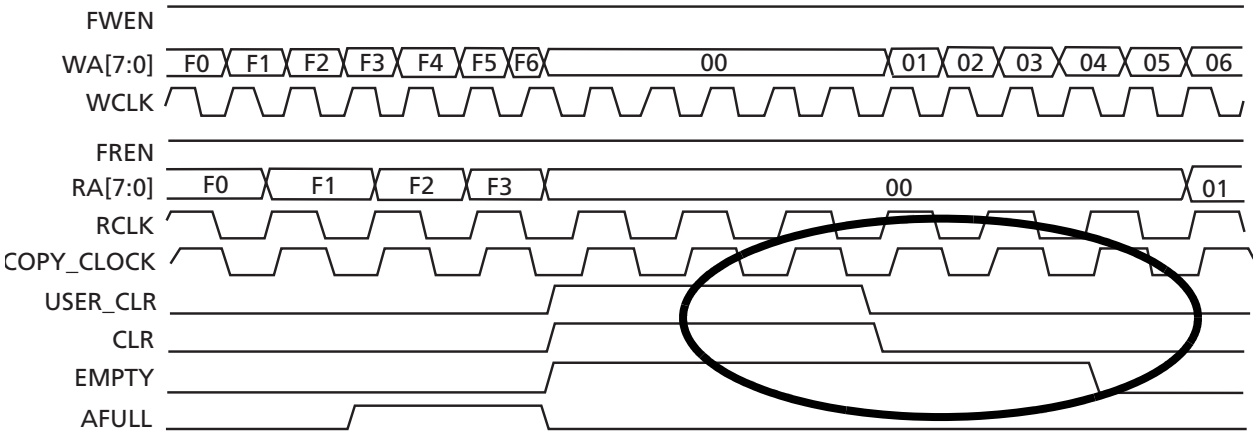

Figure 13 • **EMPTY Flag Behavior after Release of CLR During Low RCLK Pulse with Copy of RCLK and Latch Solution**

Figure 13 and Figure 14 on page 9 show that the EMPTY flag de-asserts three rising edges of RCLK after the USER_CLR or CLR de-assert. Note that COPY_CLOCK only drives the clock pin of the latch. It does not drive the FIFO Controller. Therefore, there is a design-dependent skew between RCLK and COPY_CLOCK, so if the CLR should de-assert during the clock skew gap when the COPY_CLOCK is low and RCLK is still high, the low pulse EMPTY flag behavior will not be masked. The advantage of using this circuit is that it behaves like the simple Latch Solution without losing the half RCLK cycle.

FULL and EMPTY Flag Behavior When Separate RCLK and WCLK are Used

When the RCLK and WCLK of the FIFO are different in frequency and not in phase, it is possible to observe a false assertion on the FULL and EMPTY flag.

Figure 15 on page 9 shows the EMPTY flag asserting while the FULL flag is high. After one RCLK pulse, the false EMPTY flag deasserts. Figure 16 on page 10 shows the FULL flag asserting while the EMPTY flag is high. After one WCLK pulse, the false FULL flag deasserts.

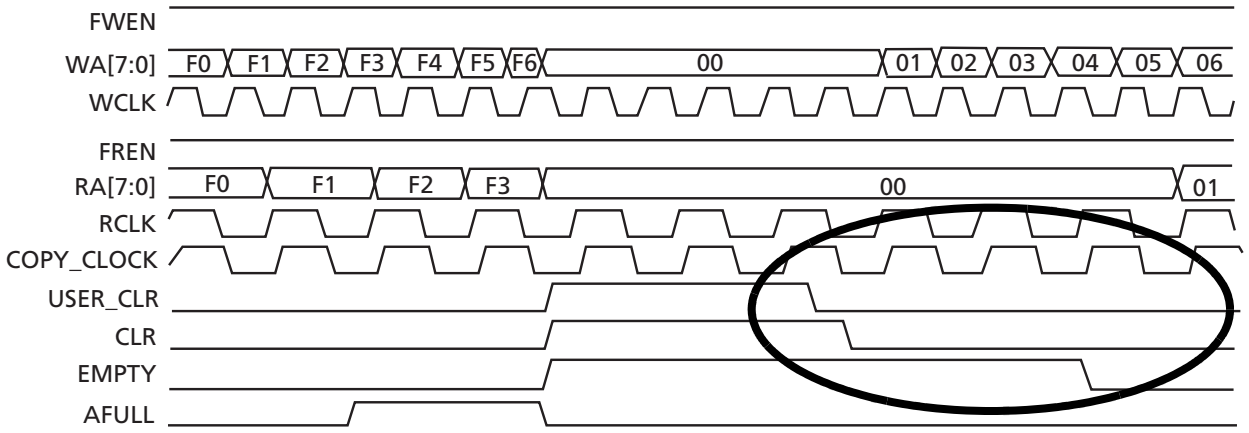

Figure 14 • **EMPTY Flag Behavior after Release of CLR During High RCLK Pulse with the Copy of RCLK and Latch Solution.**

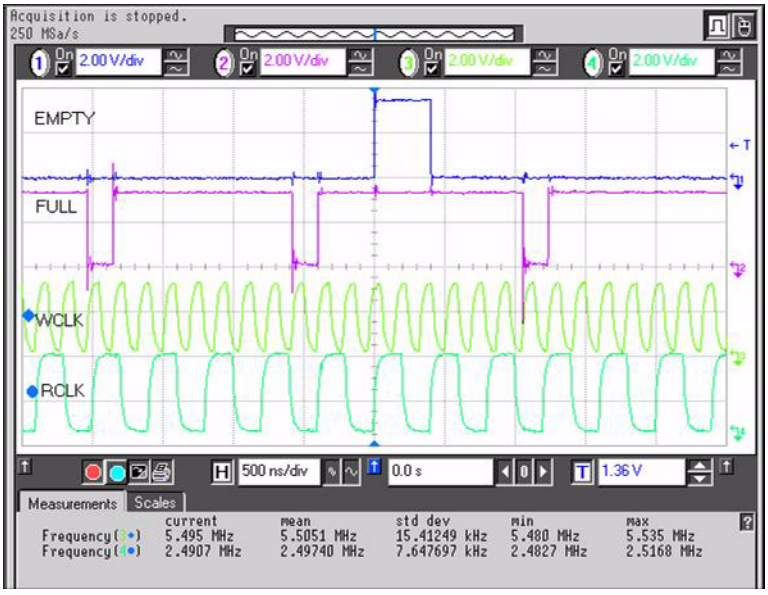

Figure 15 • **EMPTY Flag Behavior When RCLK and WCLK Are Not in Phase**

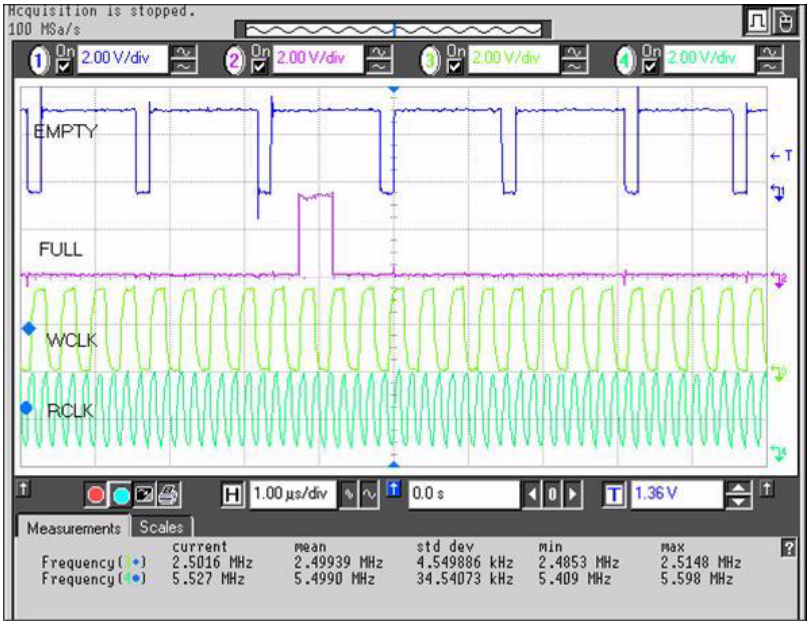

Figure 16 • **FULL Flag Behavior When RCLK and WCLK Are Not in Phase**

RTAX-S

The RTAX-S family is designed for space applications and is derived from the Axcelerator family. Being a derivative of the Axcelerator family, it has inherited the optional FIFO Controller feature and the FULL and EMPTY flag behaviors. Therefore, the solutions above may be implemented in the design to achieve the same results.

Designs are targeted to RTAX-S FPGAs because they contain core register cells that are triple modular redundant (TMR) and thus protected from single event upsets (SEU). However, the flip-flops in the FIFO Controller circuitry are not triple modular redundant and are susceptible to SEU. It is recommended that the embedded FIFO Controller not be implemented in RTAX-S designs, or only allowed for non-critical data-paths. Rather, a FIFO Controller should be implemented using SEU-protected core logic.

Conclusion

Due to the FULL and EMPTY flag assertion issue mentioned above, Actel recommends that the FIFO should be used with a single read and write clock. If separate clocks must be used, care must be taken to handle the false assertion issue. The EMPTY flag low pulse behavior only occurs if the asynchronous CLR signal transitions from high to low during the RCLK high pulse. However, simple solutions can be implemented in the design to mask this behavior. Designs should be upgraded to Libero IDE and Designer v6.1 SP1 or later in order to use simulation libraries that reflect the true EMPTY flag behavior.

List of Changes

Previous Version	Changes in Current Version (51900094-1/8.05*)	Page
51900094-0/3.05*	The "FULL and EMPTY Flag Behavior When Separate RCLK and WCLK are Used" section on page 8 is new.	8
	The "RTAX-S" section on page 10 was updated.	10

Note: **This is the part number located on the last page of the document.*

Related Documents

Application Notes

Axcelerator Family Memory Blocks
http://www.actel.com/documents/AX_Blocks_AN.pdf

Datasheets

Axcelerator Family FPGAs
http://www.actel.com/documents/AX_DS.pdf
RTAX-S RadTolerant FPGAs
http://www.actel.com/documents/RTAXS_DS.pdf

www.actel.com

Actel Corporation

2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.

Dunlop House, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Phone +44 (0) 1276 401 450
Fax +44 (0) 1276 401 490

Actel Japan
www.jp.actel.com

EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668

Actel Hong Kong
www.actel.com.cn

Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488

51900094-1/8.05