XAPP629 (v1.1) November 21, 2002

Interfacing the IDT 3.3V Multi-Queue FIFO to a Virtex-II FPGA

Summary

The Virtex™-II series of FPGAs provide access and interface to a variety of memory resources, both off and on the FPGA. In addition to the on-chip distributed RAM and block RAM features, Virtex-II FPGAs interface to a variety of external high-speed memory devices. One such device is the new Multi-Queue™ FIFO family, a powerful new memory architecture manufactured by Integrated Device Technology (IDT™).

Introduction

Data buffering and FIFO queuing are common challenges in high-performance data acquisition and data communication applications. Looking particularly at communication systems, there is a real need for FIFO buffering and queuing functions. FIFOs are useful for short-term buffering and for coupling between two different clock domains. Typically, data streams and data paths are buffered by traditional FIFOs. IDT's Multi-Queue FIFO provides the traditional benefits and features of a standard FIFO, with the added functionality of independently accessed discrete queueing by the Write and Read ports. Each queue is a FIFO buffer. Based on a new industry architecture combining high-speed queuing logic with an embedded FIFO memory core, the Multi-Queue FIFO is fully programmable. The number of queues, depth of each queue, and flag offset positions are all programmable.

Multi-Queue FIFOs are available in densities up to 2 Mb and clock speeds up to 200 MHz. By integrating high-speed logic and memory on silicon, multi-queuing FIFOs are able to achieve sustained throughput rates up to 7.2 Gb/s

Since the devices are programmable and the queues are addressable on both the Write and Read ports, some control is involved to operate the ports. This application note provides general guidelines and suggestions for control and interface requirements between a Virtex-II FPGA and a Multi-Queue FIFO. The example in this application note uses a 3.3V Multi-Queue FIFO. The timing for a 2.5V Multi-Queue FIFO is slightly different. For more information on the Virtex-II devices, refer to the data sheet. More information and data sheets for Multi-Queue devices are available on the IDT web site at: www.idt.com.

Multi-Queue FIFO at a Glance

- 2.5V and 3.3V families
- Q-Number options: 4Q, 8Q, 16Q, 32Q
- Memory density options: 256 Kb, 512 Kb, 1 Mb, 2 Mb
- Up to 200 MHz high-speed operation
- x9, x18, x36 bit-wide data port options
- Individual, active queue flags
  - Output Valid ($\overline{OV}$)
  - Full Flag ($\overline{FF}$)
  - Almost Empty ($\overline{AE}$)
  - Almost Full ($\overline{AF}$)

- Programmable Flags
  - Programmable-Almost-Full ($\overline{PAF}$) Flag
  - Programmable-Almost-Empty ($\overline{PAE}$) Flag
- Dedicated 8-bit flag buses to monitor all $\overline{PAF}_n$ and $\overline{PAE}_n$ queues
- Packet Ready (PR) mode of operation
- 256-pin Ball Grid Array (BGA) package with JTAG functionality
- Bus-width matching on both ports
- Available with selectable LVTTL or HSTL I/O (2.5V family)
- Echo read clock and enable available for source-synchronous clocking (2.5V family)
- Cascade up to eight Multi-Queue devices for depth and queue expansions (256 queues)

Multi-Queue FIFO Architecture

Basic Concept

Between one and 32 discrete FIFO queues are available in a Multi-Queue FIFO device. All queues within the device have a common data input bus (the Write port) and a common data output bus (the Read port). Data written into the Write port is directed to a respective queue via an internal de-multiplex operation addressed by the designer. Data read from the Read port is accessed from a respective queue via an internal multiplex operation addressed by the designer. Data Write and Data Read work at speeds up to 200 MHz and are totally independent of each other. A queue can be selected on the Write port and a different queue on the Read port or both ports may select the same queue simultaneously. Figure 1 shows the block diagram of IDT's Multi-Queue FIFO.

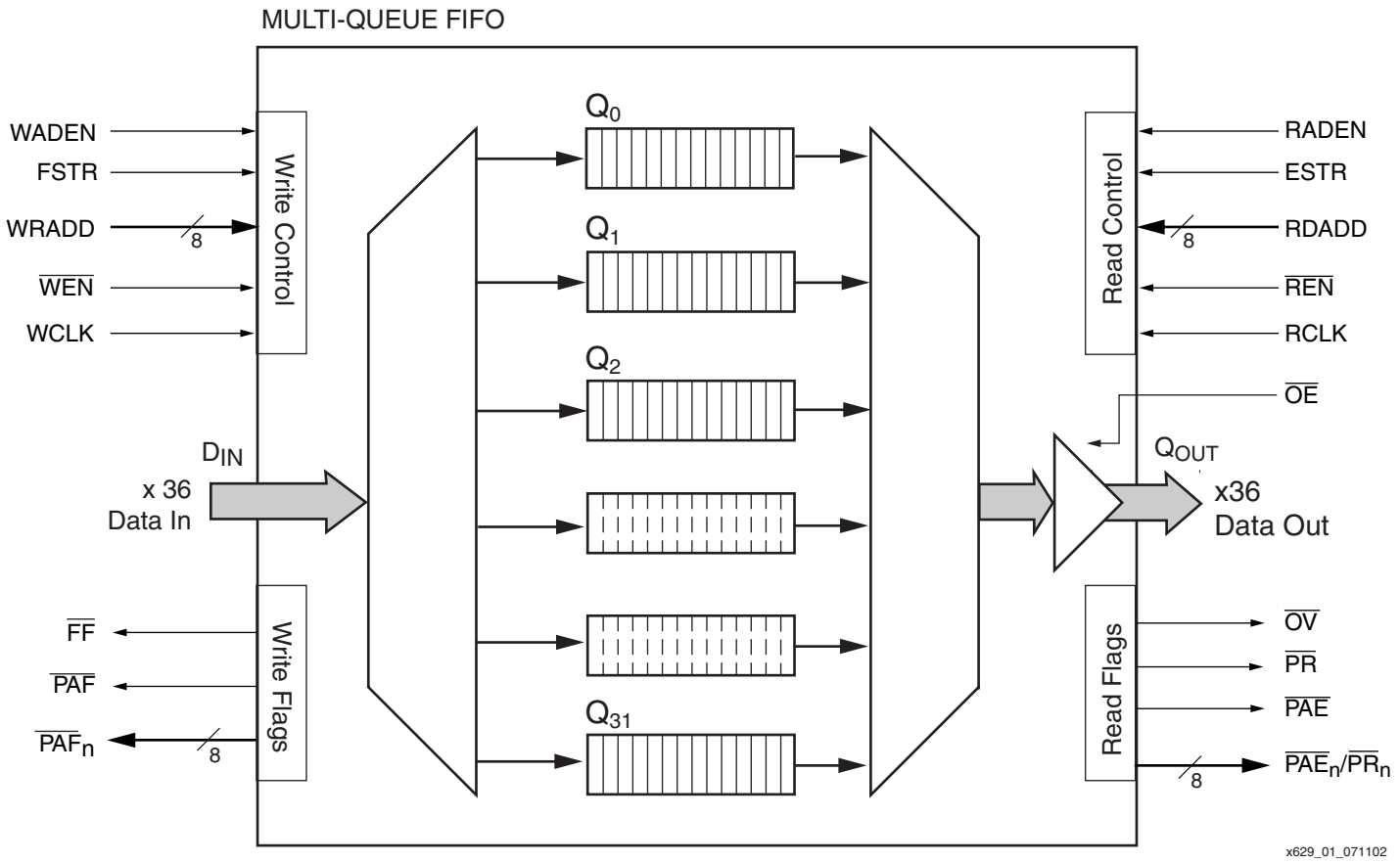

Figure 1: **Block Diagram**

Queue Configuration

Configuring queues within the device is fully flexible. The total number of queues between one and 32 and the individual queue depths are programmable. A default option is available to configure the device in a predetermined manner. Each Multi-Queue device has a total amount of available memory. Queues are configured within the device using some or all of this available memory. A single device can configure queues of varying depths.

Queue Flags

For the selected queue, the FIFO provides Full flag ($\overline{FF}$) status for Write operations and Output Valid ($\overline{OV}$) flag status for Read operations.

Also, an independent Almost Full ($\overline{AF}$) and Almost Empty ($\overline{AE}$) value can be set for each queue. These values are monitored using the $\overline{PAF}$ and $\overline{PAE}$ flags for the selected queue.

The device provides two 8-bit flag buses ($\overline{PAF}_n$/$\overline{PAE}_n$) to retrieve the $\overline{AF}$ and $\overline{AE}$ status of queues not selected for Read or Write operations. When eight or less queues are configured in the device, each bit of these flag buses represents an individual flag per queue. When more than eight queues are used, a block of eight queues can be monitored at a time.

Packet Ready Mode (PR)

A PR mode of operation is also available on the Multi-Queue family. Here the device provides status of whether or not one or more *full* packets of data are present in the respective queue. A PR flag status is available for each queue in a device.

Bus-Width Matching

Bus-Width matching is available on the FIFO. Either port can be 9-bit, 18-bit, or 36-bit provided that at least one port is 36-bits wide. This allows seamless interface buses of different widths.

Depth Expansion and Queue Expansion

Expansion of Multi-Queue FIFOs is also possible. Up to eight devices can be connected providing the possibility of both depth expansion and queue expansion. Depth Expansion means expanding the depths of individual queues; queue expansion means increasing the total number of queues available.

Interface Connections

Figure 2 shows the connections between the a single Virtex-II FPGA controlling both the Write and Read ports of a Multi-Queue FIFO. The Write port control operates on a separate clock than the Read port control. This illustrates one advantage of using the combination of the Virtex-II FPGA and Multi-Queue FIFO since both devices provide data transfer between different clock domains. For example, the Write clock runs independently even at a different speed than the Read clock.

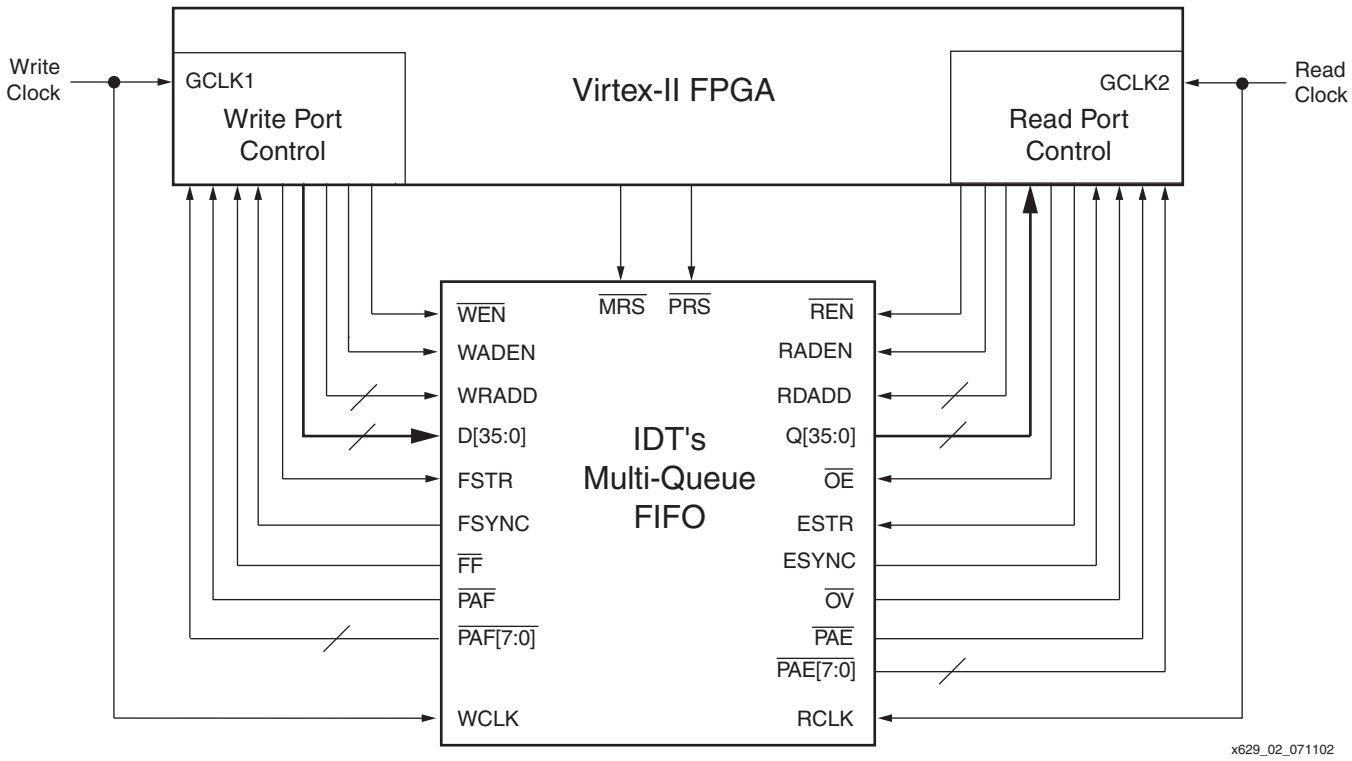

Figure 2: **Interconnect Diagram**

The Virtex-II series and Multi-Queue family also complement each other because the I/O of each device can be configured to different I/O standards. The 2.5V Multi-Queue family supports 2.5V LVTTL, 1.5V HSTL and 1.8V HSTL. The Virtex-II series support these three standards and others.

Set-Up, Configuration, and Basic Operation

The 3.3V Multi-Queue FIFO

There are three main stages to using an IDT Multi-Queue FIFO:

1. Master Reset
2. Programming
3. Normal Operation (Writes and Reads)

This application note guides the designer through each of these stages and makes suggestions on how to control each stage.

Reset, Programming, and Configuration Operation

In this application note, the same device and/or module shows both the Write and Read ports being controlled. However, it is possible to have a separate device controlling the Read port and a separate device controlling the reset logic.

Any input that does not toggle can be tied High or Low. For example, if a Partial Reset is never required, the $\overline{PRS}$ input does not need to be controlled by the controller and can be tied inactive, High. This discussion also requires an idle JTAG port.

Control Pins Involved

$\overline{MRS}$, $\overline{PRS}$, $\overline{WEN}$, WCLK, WRADDn, WADEN, FSTR, $\overline{REN}$, RDADDn, RADEN, ESTR, $\overline{SENI}$, DFM, DF.

Master Reset and Programming

The master reset described in Figure 3 is initiated with a *System Reset* command. When a System Reset occurs, a Master Reset of the Multi-Queue FIFO follows.

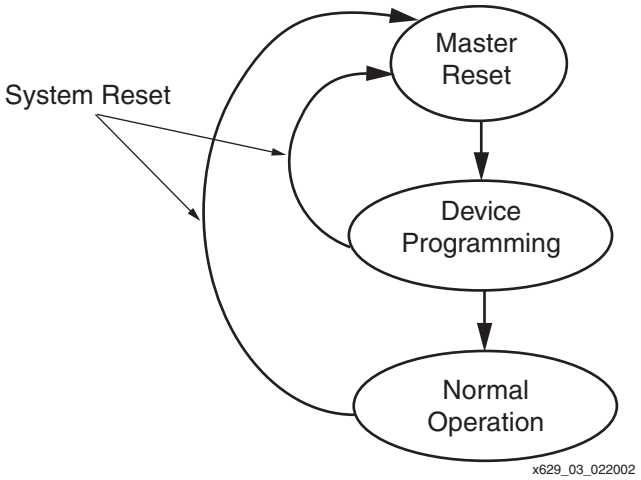

Figure 3: **Reset Flow Chart**

Control of the Master Reset function is available in the "Master Reset and Programming Function" sample Verilog code files on the IDT site at:

http://www.idt.com/docs/MQ_Verilog_Example.txt.

Default programming to set up the Multi-Queue device in a pre-determined configuration should be completed initially. The IDT Application Note, **AN-303** “Multi-Queue FIFO – Serial Programming” gives a detailed example for customizing the configuration of the Multi-Queue FIFO.

Partial Reset

A Partial Reset can be performed on any individual queue within the device. A partial reset will reset the Read and Write memory pointers to the first location of the queue. To perform a partial reset the respective queue must be selected on both the Write and the Read ports before the partial reset is initiated. The "Partial Reset" Verilog code illustrates how to achieve a partial reset. The code assumes a single Virtex-II FPGA is used to control both the Write and Read ports, and the same clock is applied to the Write and Read ports. When a Partial Reset is required, the code assumes a Partial Reset request is provided.

Normal Operation

Write Port Control

The Write Timing port is illustrated in the timing diagram of Figure 4. To write into a queue, first select it. Two write clock cycles later after an actual write operation, the queue occurs and data on the D_{IN} input bus is written into the queue on the rising edge of the write clock. On the previous two clock cycles data can be written into the previously selected queue. This illustrates 100% bus utilization. The timing diagram assumes all FIFO queues are *not* Full and are, therefore, available to accept data.

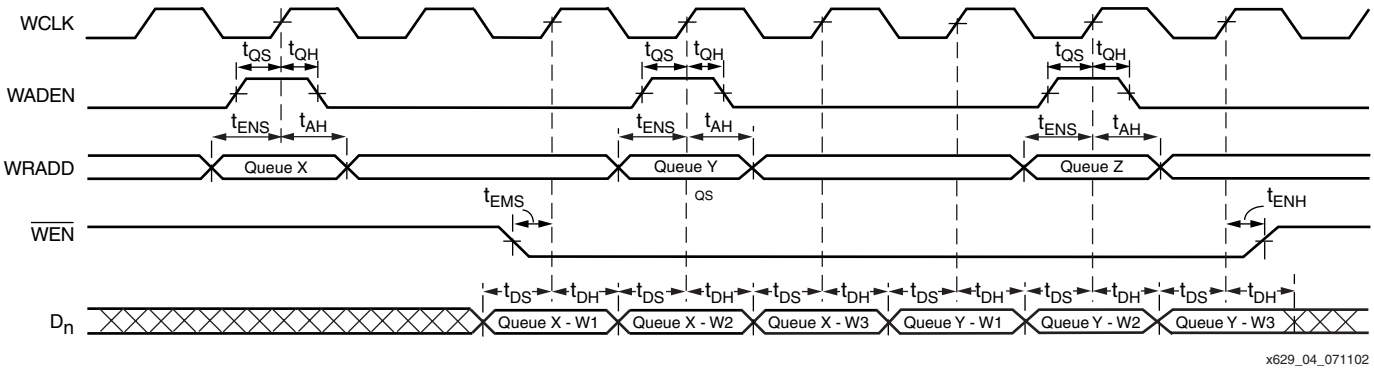

Figure 4: **Write Control**

The FPGA controls the Write port and all Write operations. The FPGA Write control must take into consideration the two-cycle latency between the selection and the writing of data to the queue. The FPGA Write control must also detect when a queue being selected is Full by monitoring the flip-flop ($\overline{FF}$) output from the IDT Multi-Queue device. The $\overline{FF}$ output provides status of the queue selected on the next WCLK clock cycle after the queue selection is made.

The "Write Port Control" Verilog code is a simple example of a Virtex-II FPGA providing effective control of the Multi-Queue Write port. The code assumes the following:

1. A Master Reset has been performed on the Multi-Queue FIFO.
2. Either default or serial programming of the Multi-Queue FIFO has occurred.
3. The FPGA determines when a queue is written to. The initial part of the code is waiting for a queue request to be made.
4. The Multi-Queue contains 32 addressable queues.

Read Port Control

The control algorithm and logic controlling the Read port must be able to recognize a valid data Read. A new data word is accessed and placed onto the Multi-Queue output bus. At least one of four possible events helps to recognize a valid Data Read:

1. By taking the $\overline{REN}$ input Low, a Read can occur from a previously selected queue. A new word is accessed on the rising edge of the Read clock and is available for the reading device to process on the following rising edge of the Read clock.
2. The Multi-Queue device operates in a First-Word-Fall-Through (FWFT) mode. If the same queue is selected on both the Write and Read ports, then the first word written into the queue automatically falls through to the outputs, regardless of the state of $\overline{REN}$.
3. During a queue switch on the Read port, a final Read from the old queue occurs. Due to the Next-Word-Fall-Through (NWFT) nature of the device, data from the old queue is accessed and placed onto the Multi-Queue data outputs, regardless of the state of $\overline{REN}$. The word from the old queue is forced out allowing an automatic Read from the new queue. The next word available in the new queue falls through to the outputs of the Multi-Queue FIFO. This leads to the fourth event.
4. Also during a queue switch on the Read port, the first word from the newly selected queue is accessed and placed onto the Multi-Queue data outputs, regardless of the state of $\overline{REN}$. Again this is due to the NWFT operation, where the next word available in the new queue falls through to the outputs of the Multi-Queue FIFO.

This application note reviews these four events and shows an overall solution implemented into the FPGA control logic to handle any event.

Event 1: Read Operations on the Current Queue

Figure 5 illustrates the first potential event. The diagram shows a situation where a queue has previously been selected on the Read port. Read operations are occurring based on valid words being available in the queue and the $\overline{REN}$ input being toggled Low to read out the data words.

The $\overline{OV}$ flag is essentially provided to indicate when a queue is empty or is read as empty. It does not indicate when a new word is placed on the data outputs.

Figure 5 illustrates the FPGA control logic implementation of a flip-flop called *lastren*. When an enabled read is performed and the word on the output bus should be processed by the reading device, a *lastren* is implemented. Using cycle *C* as an example, an enabled read is performed placing word "LW-5" onto the outputs. In the next the cycle, cycle *D* (the reading device) should process "LW-5." The state of *lastren* is used to determine whether the word is a new word and, therefore, whether it should be processed.

Based on this set-up, the reading device processes words on the following cycles:

D, *G*, *H*, *I*, *J*, and *K*

On cycle *L* the $\overline{OV}$ flag is High and the queue is essentially read to empty. In summary, when a normal Read event occurs on the current queue, a new word should be processed by the reading device on an RCLK cycle, where the $\overline{OV}$ flag is true (Low) and the *lastren* flip-flop is true (Low).

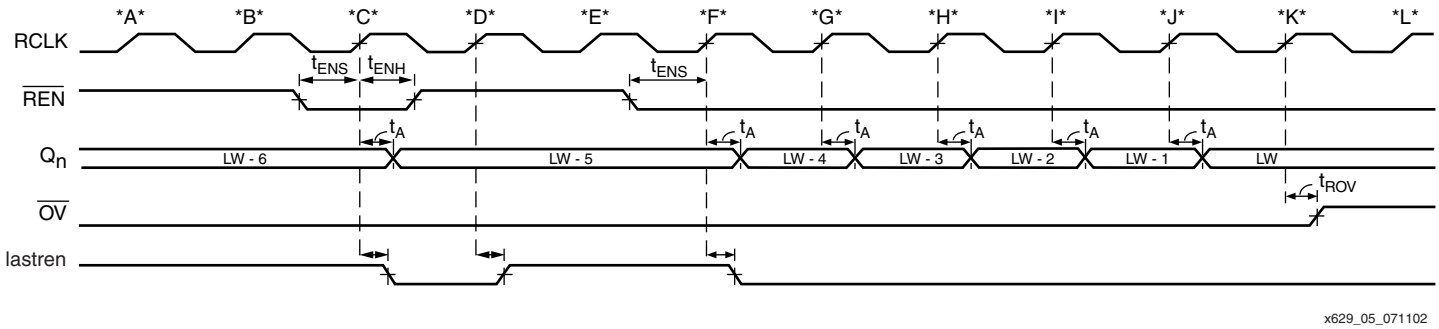

Figure 5: **Read Control - Normal Read**

Event 2: Read Operation Due to FWFT in Current Queue

Figure 6 illustrates the FWFT effect, the second possible Read event. In this diagram both the Write and Read ports are selected for the same queue. A new word, W1, is written into the queue. Before the write operation, it was an empty queue. The $\overline{OV}$ flag is High. The first word written into the queue automatically falls through to the data outputs, causing the $\overline{OV}$ flag to go active (Low). This first word has fallen through regardless of the state of $\overline{REN}$. In fact, $\overline{REN}$ is High throughout this diagram. Therefore, the use of the *lastren* flip-flop in Event 1 is no longer an option to determine if a new word is available for the reading device to process. A second flip-flop, *lastov* is included into the control logic of the reading device. This flip-flop provides the reading device with a status of the $\overline{OV}$ flag delayed by one read clock cycle. Therefore, the reading device monitors both $\overline{OV}$ and *lastov* to determine whether a new word is available. In Figure 6, on the read clock cycle *C*, the first word (W1) should be processed by the reading device.

In summary, in the case of an FWFT Read event, a new word should be processed by the reading device on an RCLK cycle, where the $\overline{OV}$ flag is true (Low) and the *lastov* flip-flop is false (High).

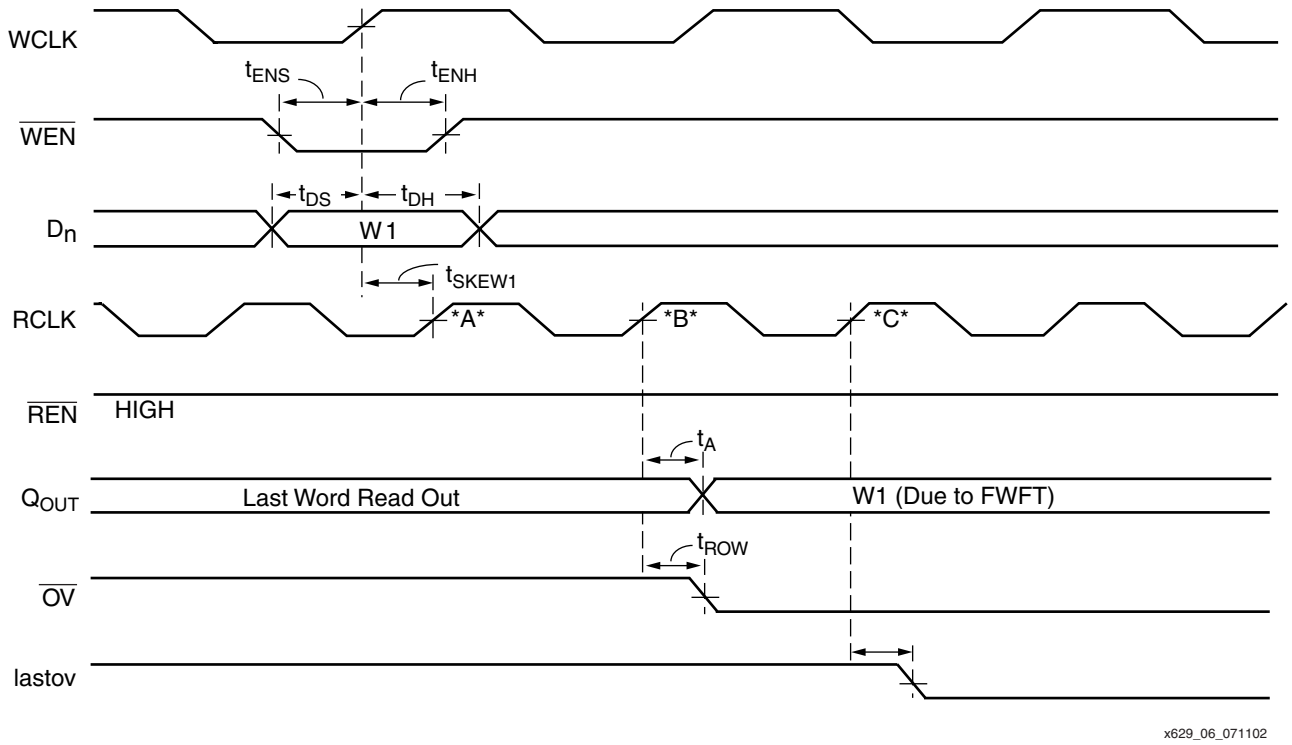

Figure 6: **Read Control - FWFT**

Event 3: Read Operation Due to a Queue Switch on the Read Port

Figure 7 illustrates the third and fourth possible Read events. These events are indirectly due to the FWFT effect caused by a queue switch on the Read port. In the diagram, a new Q is selected on the Read port on RCLK cycle *A*. RADEN is High and the RDADD address bus is addressing the new queue. When a queue selection is made, the first word of the new queue falls through to the data outputs of the Read port, forcing a final word to be read from the previous (old) queue. This occurs regardless of the state of $\overline{REN}$. In fact, the $\overline{REN}$ input is shown as High in this diagram.

During a queue switch, the $\overline{REN}$ input can remain High and the $\overline{OV}$ flag can remain Low. This indicates that both the old queue and the new queue have not been read to empty. Monitoring of the *lastren* and *lastov* flip-flops alone does not indicate to the reading device that a new word is available in this example. Hence, the register *sinceraden* is introduced into the control logic of the reading device. This register is a 2-bit register. In the event of a queue switch, the register is simply used to provide a count from zero to three. When RADEN is asserted High, this register should be reset to the value zero. On the following three RCLK cycles, the count increments up to a value of three. At three, the count ceases incrementing and waits until RADEN is toggled High once more.

In the event of a queue switch, a minimum of two new words are automatically read out from the Multi-Queue device and need to be processed by the reading device (this again is an effect of the NWFT operation). On RCLK cycle *B*, the “Word N” is read from the “Old Queue”. This word must be processed by the reading device on RCLK cycle *C*. Similarly “Word 1", the first word of the New Queue, must be processed by the reading device on RCLK cycle *D*. The reading device should monitor the *sinceraden* register. When the register has a value of either one or two, the data word on the bus is valid, providing the $\overline{OV}$ flag is Low. During a queue switch on the Read port, the reading device must realize that the new word processed with *sinceraden* at value “1” is appropriately processed data from the old queue. Secondly, the new word processed with *sinceraden* at value “2” is data appropriately processed from the new queue.

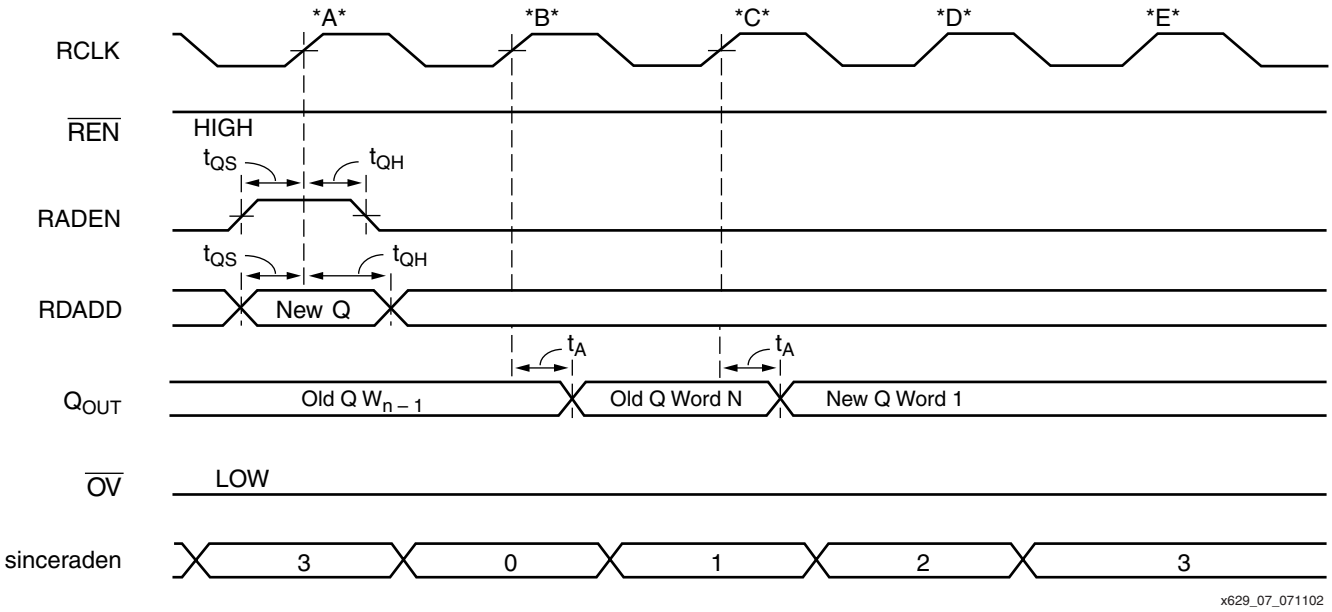

Figure 7: **Read Control -Queue Switch**

Event 4: An Example of Consecutive Read Events

Figure 8 shows an example where a number of Read events occur consecutively. An enabled read has occurred on the same RCLK cycle as the queue switch, cycle *A*. In this example, the *sinceraden* register serves exactly the same purpose as before, indicating in conjunction with the $\overline{OV}$ flag that a new word must be processed on RCLK cycles, *C* and *D* (*sinceraden* = 1 or 2). However, in this event a word from the old queue accessed on RCLK *A* must also be processed. This word from the old queue, word N – 1, must be processed by the reading device on RCLK *B*. The reading device control logic processes word N – 1 because the *lastren* flip-flop is active (Low) on this cycle. In conjunction with $\overline{OV}$ being true, the word N – 1 is processed. As with the previous example, the data words processed by the reading device on RCLK cycles *B* and *C* are appropriately processed from the old queue.

In summary, it is possible for up to three words to be processed by the reading device when a queue switch is made. The first word is processed due to the queue switch performing an enabled Read. The state of *lastren,* in conjunction with the $\overline{OV}$ flag, ensures the first word processing. When register *sinceraden* has the value of “1” or “2” and the $\overline{OV}$ flag is active (Low), the following two words are processed. Again, a valid word read with *sinceraden* equal to “1” is an appropriately processed word from the old queue.

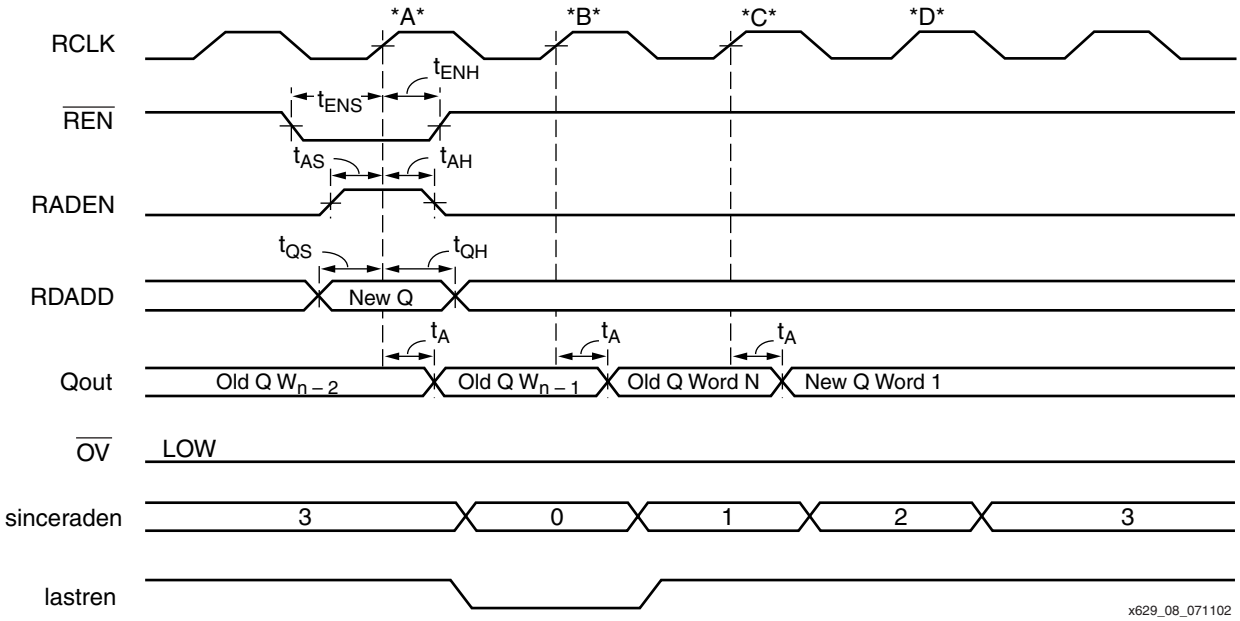

Figure 8: **Read Control - Multiple Reads**

Read Port Verilog Implementation

The control logic required to handle all four events is easily implemented in the FPGA. The "IDT 2.2V Multi-Queue Read Port Control" Verilog code example produces the control logic required to detect when a word out of the Multi-Queue Read port is a new valid word to process.

Flag Bus Operation

There are two flag buses provided on the Multi-Queue device. Depending on the selected device, the flag bus is either 4-bits wide or 8-bits wide. If a 4-queue device is used, the flag buses are 4-bits wide and provide constant status of $\overline{AF}$ and $\overline{AE}$ conditions for each queue. An 8-queue, 16-queue, or 32-queue device has 8-bit wide buses. The 8-bit buses on the 8-queue device provide continuous status of each queue. The flag buses for the 16-queue and 32-queue devices provide status of all queues in a multiplex manner. There are two modes of operation for the flag bus in a 16-queue or 32-queue part:

1. Direct Mode – The designer addresses the queues required on the flag bus.
2. Polled Mode – The queues are cycled on the flag bus.

In the following discussion, an example is provided for utilizing a 32-queue device with the flag bus operating in Polled mode.

In a 32-queue device both the $\overline{AF}$ flag bus ($\overline{PAF}_n$) and the $\overline{AE}$ flag bus ($\overline{PAE}_n$) are 8 bits wide. The status of all 32 queues can be obtained over the period of four 8-bit cycles, each 8-bit status is called a "Quadrant." Figure 9 illustrates the timing of the $\overline{AF}$ flag bus and the relationship between the FSYNC pulse and the flag bus data. Each quadrant is placed onto the bus with respect to a WCLK cycle, the first quadrant is marked with a logic High on the FSYNC output. The $\overline{AE}$ flag bus has equivalent timing based on an RCLK input and providing an ESYNC output.

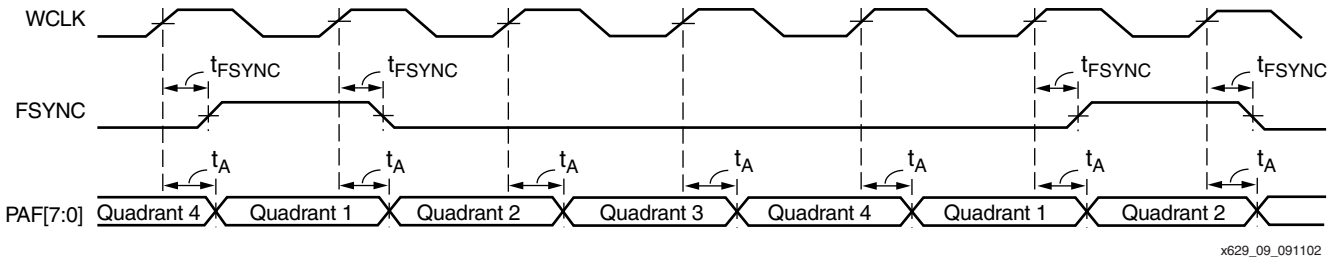

Figure 9: **Flag Bus Control**

The Flag Bus Control Verilog code example shows how the Virtex-II FPGA keeps track of all 32 queues by updating a 32-bit flag bus register. This code assumes that a 32-queue device is being used in Polled mode. The same code can be applied to the $\overline{AE}$ flag bus operation.

Packet Mode Operation

Applicable Devices

Packet mode operation of the Multi-Queue FIFO is only available on these 36-bit wide devices:

IDT72V51236, IDT72V51246, IDT72V51256

IDT72V51336, IDT72V51346, IDT72V51356

IDT72V51436, IDT72V51446, IDT72V51456

IDT72V51546, IDT72V51556

Introduction

Packet mode is a selectable device mode and must be selected during a Master Reset. When in Packet mode the Packet Ready ($\overline{PR}$) flag becomes available and the $\overline{PAE}_n/\overline{PR}_n$ bus operates in Packet mode, providing Packet Ready status of queues. When in Packet mode, the Output Valid ($\overline{OV}$) flag is still available.

When in Packet mode the designer must utilize the most significant 3 bits of the data bus as Packet markers. D35/Q35 is the "End Of Packet" (EOP) marker, D34/Q34 is the "Start Of Packet" (SOP) marker and D33/Q33 is the "Almost End Of Packet" (AEOP) marker. When writing packets into a queue, the designer must include these markers at the appropriate positions. The 36-bit wide family of Multi-Queue devices contains logic to monitor these markers entering and leaving the Multi-Queue FIFO. This device then provides $\overline{PR}$ status for both the queue selected on the Read port as well all other queues within the device. Please refer to an IDT data sheet for more details on the Packet mode operation.

This section of the application note provides a clearer understanding of the control requirements of the Multi-Queue FIFO when operate in Packet mode. It is also makes suggestions and recommendations on controlling and operating the Read port.

Operation

When using the Multi-Queue FIFO in Packet mode, the designer utilizes a SOP marker, an EOP marker, and an AEOP marker. The purpose of the SOP and EOP markers is apparent when reviewing the data sheet. To provide accurate $\overline{PR}$ (packet availability) status for a respective queue, the Multi-Queue Packet Ready logic must track these markers on both queue entrance and exit. The purpose of the AEOP marker is not as apparent, although important to the efficient operation of the Multi-Queue FIFO.

It is very important to have appropriate control of the Read port in Packet mode to maintain effective switching between queues and packet integrity. Due to the pipelining structure of the Read port, one or two words will automatically read out on the Read port when a queue switch is made. When a Read port queue switch is made from Queue 1 to Queue 2, the next available

word in Queue 1 followed by the next available word in Queue 2 will be read out consecutively. This happens regardless of the state of the $\overline{REN}$ input, as long as the queues have complete packets available. A word cannot be read from a queue unless a complete packet is available in that queue.

There are a number of scenarios to consider in Packet mode operation of the read port.

A. Reading out one (or more) packets until the last complete packet of a queue is read.

B. Reading out a packet and then switching to a queue with another packet waiting.

C. Reading out a packet with another complete packet behind it in the same queue, and stopping reads without accessing the next packet.

D. Switching from a queue with no packets available (maybe empty or containing just a partial packet), to a queue with a packet ready.

Mode A

This is shown in Figure 10. Essentially, when the packet is completely read out and the last word in the queue has been accessed, the $\overline{OV}$ flag will be asserted High thus preventing any further reads. This also covers the situation where multiple packets are read from the same queue until the queue has gone empty, or there are no more complete packets available.

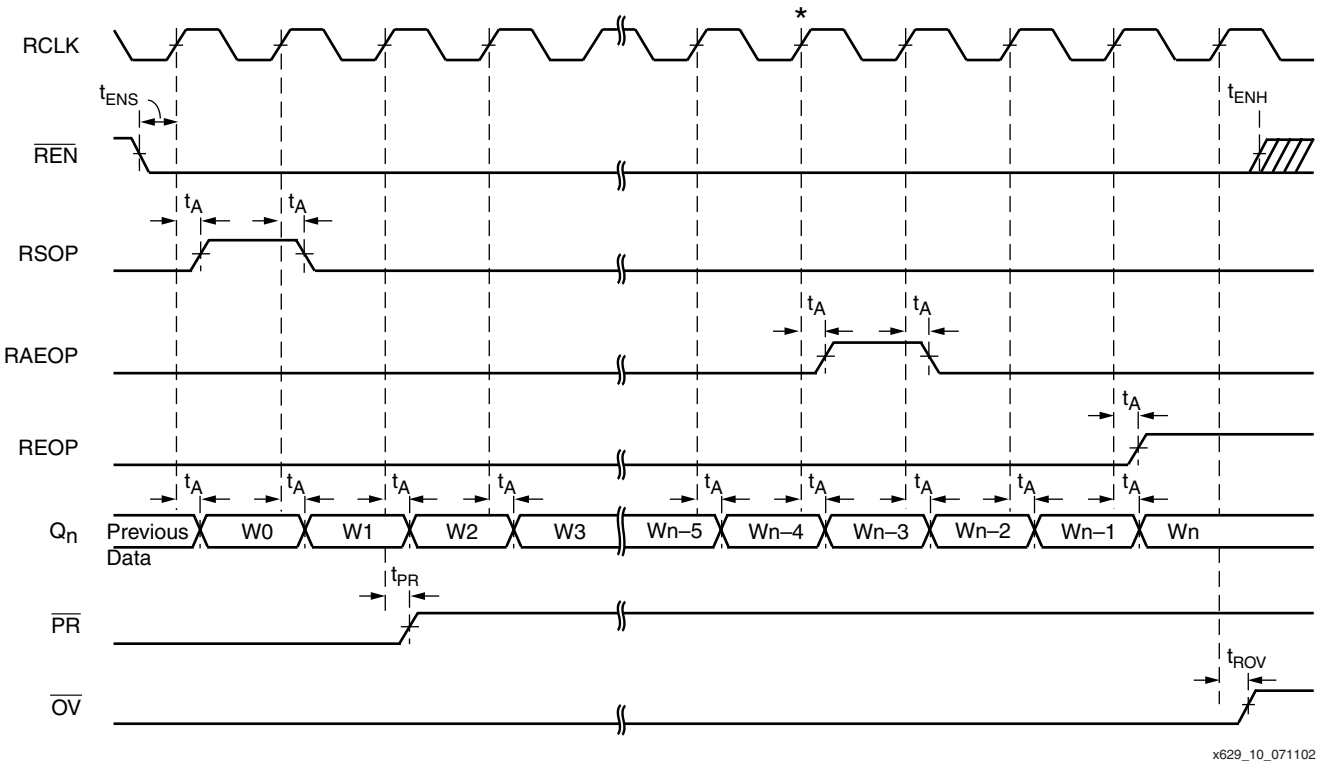

Figure 10: **Read Port Packet Mode A**

Notes:

1. This diagram shows a Queue being empties of it's final "complete" packet.
2. On cycle *, a decision must be made whether there is another packet to be read and where it is located. In this diagram the current queue is read to empty.

Mode B

This operation brings about the possible need for the implementation of a Null-Queue. As explained in the data sheet for the Multi-Queue FIFO, the Null-Queue is a default queue to be selected at the end of read operations on a respective queue to prevent the pipeline from being filled with the next word available in the current queue. This is also applicable in Packet Ready mode. If the designer requires a packet to be read from a queue and NOT begin reading the

next complete packet in that same queue, a queue switch should be made out of that queue, either into a queue that has no complete packets available or into a Null-Queue. The timing of the queue switch is important for two reasons:

1. To ensure the last word of the required packet is read out, mark the word with an "EOP of Packet A in Queue 1".
2. To ensure the first word of the next packet (marked with SOP) is not pushed into the read pipeline, mark the word with an "SOP of Packet B in Queue 1".

If a queue is available with no complete packets ready, a queue switch can be made to this queue by virtue of the fact that data cannot be read from a queue that does not have a complete packet available. However, there may be an instance where this is not possible, i.e. all queues have packets available, but reading these packets is not currently desired. In this case a Null-Queue must be selected, to effectively, flush the last EOP word of Packet A from Queue 1 and not read any further words or push the SOP word of Packet B from Queue 1 into the pipeline.

In the Multi-Queue data sheet, the definition of a Null-Queue is a queue that can never be written to or read from. To the Write port this queue always behaves as though it is full (Full flag Low) and to the read port of this queue always behaves as though it is empty ($\overline{OV}$ flag HIGH). Figure 11 shows this mode.

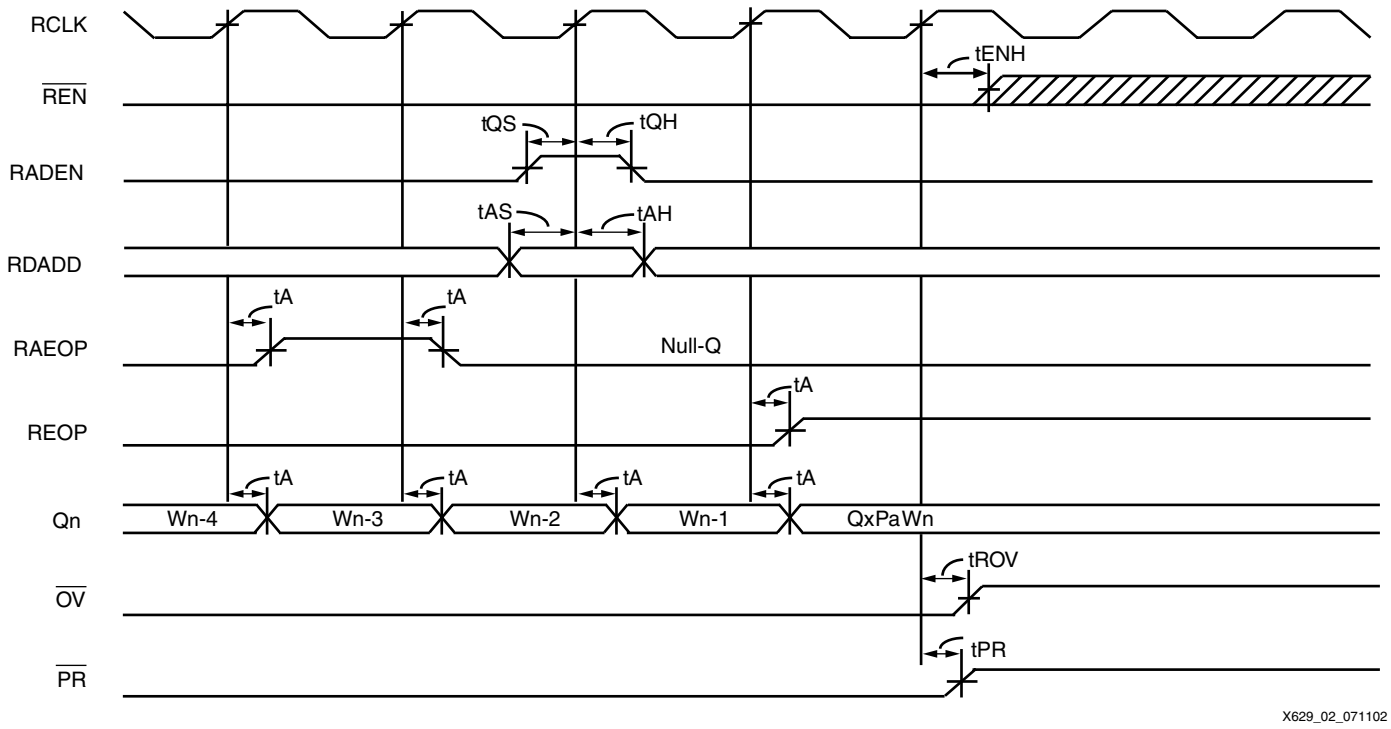

Figure 11: **Read Port Packet Mode B**

Mode C

When a queue switch is required at the end of a packet, making the next read into a packet from the new queue, the device reading data from the Multi-Queue FIFO must ensure a queue switch occurs four RCLK cycles ahead of the EOP. To do this, the designer should utilize the AEOP marker. Figure 12 shows the mode.

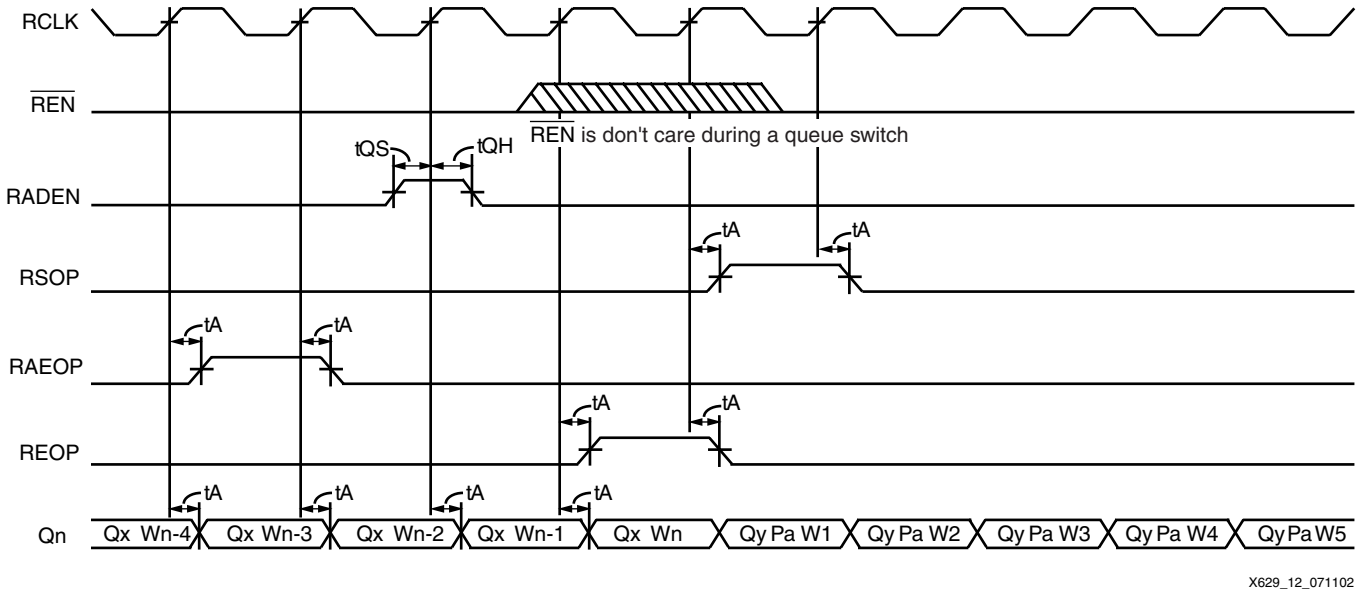

Figure 12: **Read Port Packet Mode C**

Mode D

Mode D can happen when the Read port has previously been selected for a queue that either contains no packets available (or only a partial packet available), or the Read port has been selected for the Null-Queue. When packet becomes available within another queue, thus requiring a queue switch to service that packet. This operation is quite straightforward, the first word of the new packet is accessed two RCLK cycles after the queue selection is made. Figure 13 shows a Null-Queue selection being made, followed at a later time by a selection of Queue Y being made. When the Null-Queue is selected, complete the reading of Packet 1 from Queue X. This last word from Queue X will remain on the output bus until a new queue (with a complete packet) is selected.

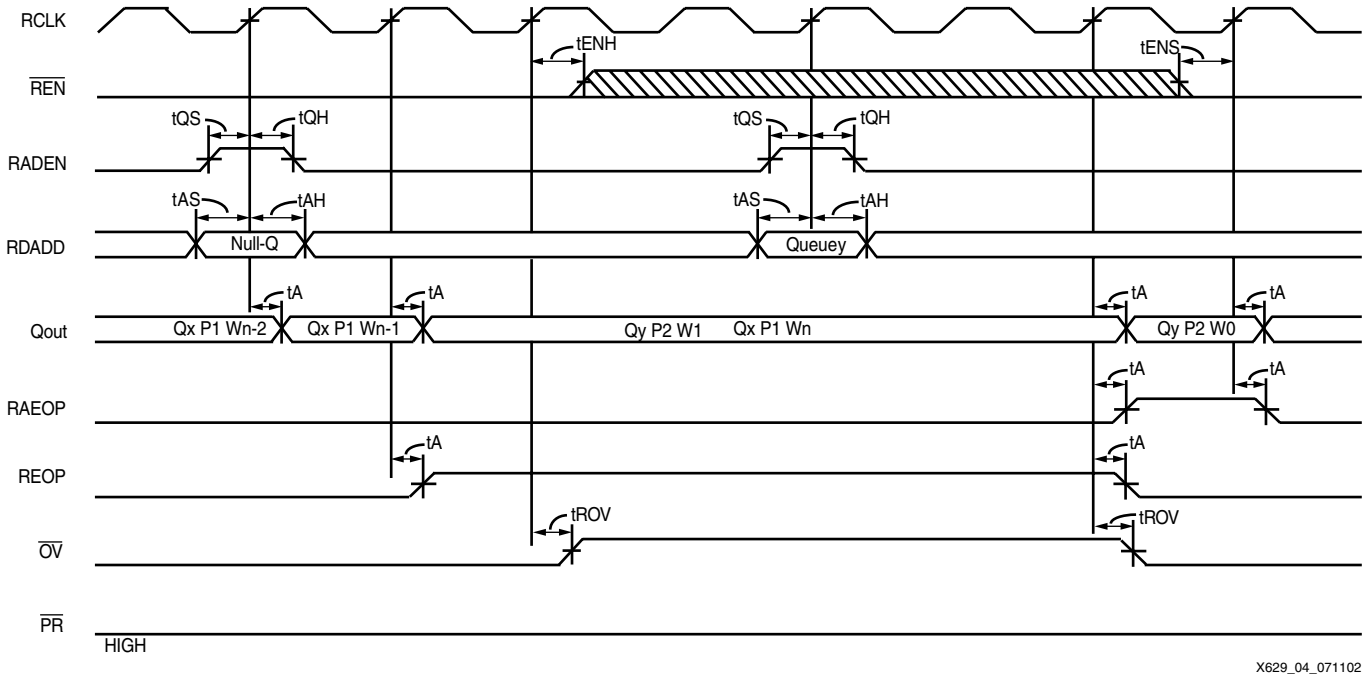

Figure 13: **Read Port Packet Mode D**

Packet Mode Summary

An important guideline when using the Multi-Queue FIFO in Packet mode involves the use of an AEOP marker. The AEOP marker must be set a minimum of four locations away from the EOP. The controlling device must always determine what action must be taken before the EOP marker (last word of the packet), is accessed. For the reasons discussed above, when an AEOP is detected the Read port controlling device needs to make a decision on where the next packet will be accessed from. If no further packets are to be accessed, the selection of an empty queue (or even Null-Queue) could be required to "flush the pipe".

To help determine a queue to switch to (if any), the Multi-Queue FIFO provides a $\overline{PR}$ flag bus. In Packet mode this $\overline{PR}_n$ bus becomes available and provides a $\overline{PR}$ status for all queues, (selected or not selected). The designer should monitor this bus and to determine which queue a packet will be read from at any given time.

An example block diagram of process when an AEOP marker is detected on the read port is shown in Figure 14.

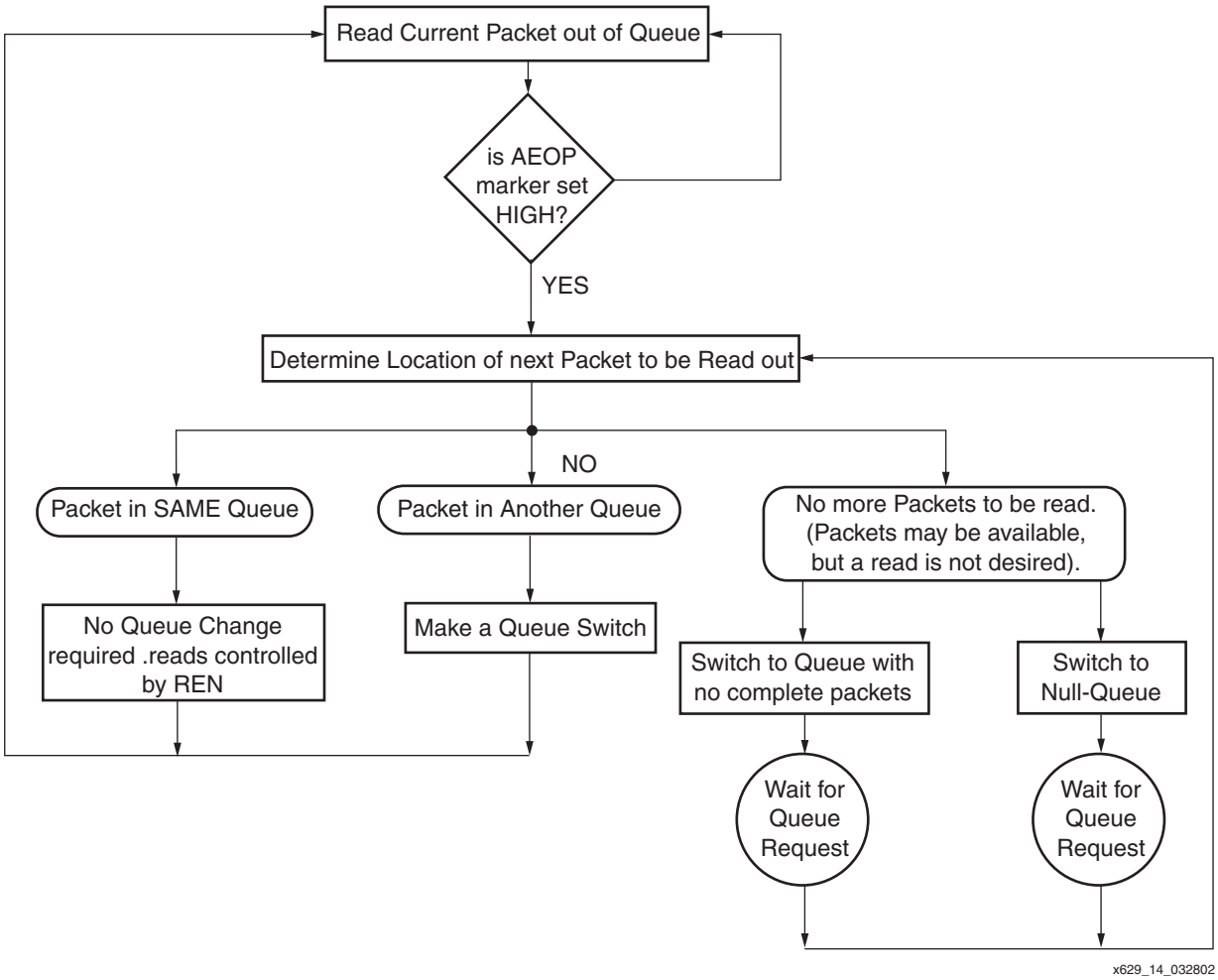

Figure 14: **Packet Mode Flow Diagram**

If any switching latencies are within the Read port controlling device when making a queue switch decision, the AEOP marker needs to be set more than four locations away from EOP to allow for this additional delay.

Conclusion

The Multi-Queue FIFO and Virtex-II FPGA solution is ideal for solving queuing and scheduling problems in high-performance networking applications, such as terabit routers, multi-service switching platforms, and wireless base stations. The Multi-Queue FIFO memory devices

provide the ability to buffer multiple types of data traffic in a single device, allowing the user to select differentiated queues for performing independent write or read operations.

Queuing, scheduling, and classification issues are becoming increasingly important for system designers attempting to provide leading-edge performance in the area of guaranteed bandwidth and quality of service (QoS). The combination of IDT Multi-Queue FIFO and Xilinx Virtex-II FPGA provides a solution to queuing and scheduling issues inherent in the design of switches and routers. In the past, queuing capabilities were implemented with discrete custom logic and memory approaches. By integrating high-speed logic and memory on silicon, multi-queue FIFOs are able to achieve sustained throughput rates up to 7.2 Gb/s, while also providing the high-speed queuing logic required to assist in the implementation of packet queuing and scheduling engines. The IDT multi-queue devices use less power, fewer I/O pins, and less board space than discrete solutions, while also reducing overall system cost and development risk. The family offers a wide range of configurations with densities up to 2 Mbits per device.

Within the unique multi-queue architecture, between one and 32 discrete queues may be configured on the device at initialization. Individual queue sizes and full/empty flag boundaries are user-programmable, allowing system designers to differentiate the priority of data traffic. If higher queuing densities are required, the devices may be cascaded to implement up to 256 prioritization queues, thus allowing for scalability.

In addition to byte boundaries, the multi-queue FIFO can operate in packet boundaries, helping to service data traffic more efficiently. The devices provide independent read and write port-clocking domains to simplify hardware design and improve system performance, while continuous monitoring of queue status helps to improve system real-time performance. By offering speeds up to 200 MHz in x36 configurations, these products enable customers to achieve the requisite line-speed performance, equivalent to three OC-48 flows in one device. The IDT multi-queue family also offers bus-width matching capabilities, allowing each port to be configured at x9, x18 and x36 widths, and providing a seamless interface between systems operating at different data widths.

This application note provides the design engineer with guidelines and examples on how to implement control logic into the Virtex-II FPGA to enable effective and efficient operation a Multi-Queue device connected to the FPGA. The application example shown illustrates a single Virtex-II FPGA controlling all operations of the Multi-Queue FIFO. However, the Multi-Queue Write and Read ports can be controlled by totally independent FPGAs. As with all IDT FIFOs the Write and Read ports are totally independent having their own clock inputs, WCLK and RCLK respectively. This code can be modified to suit the exact requirements of the application and also to incorporate more than one Multi-Queue device, for example when queue or depth expansion is performed. For more information on the Multi-Queue family the reader may refer to the IDT web site at: **www.idt.com**. The sample Verilog code is available on the IDT site at: **http://www.idt.com/docs/MQ_Verilog_Example.txt**.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/11/02	1.0	Initial Xilinx release.
11/21/02	1.0	Updated link to IDT's web site.