XAPP701 (v1.1) November 1, 2004

# Memory Interfaces Data Capture Using Direct Clocking Technique

Author: Maria George


## Summary

This application note describes the direct-clocking data capture technique for memory interfaces in a Virtex-4 device. The direct-clocking scheme utilizes some of the architectural features unique to the Virtex-4 family (for example: the 64-tap absolute delay line provided in each I/O block (IOB)).


## Introduction

Most memory interfaces are source-synchronous interfaces where the data and clock/strobe transmitted from the external memory device is edge aligned. To capture this transmitted data in the Virtex-4 device, either the clock/strobe or the data is delayed. In the direct-clocking technique, the data is delayed and is center aligned with respect to the internal FPGA clock. In this scheme the internal FPGA clock captures the transmitted data. The clock/strobe transmitted from the memory is used to determine the delay value for the associated data bits. As a result, there are no restrictions on the number of data bits associated with a strobe. Since the strobe does not need to be distributed to the associated data bits, no additional clocking resources are required.

The Virtex-4 resource used by the clock/strobe and the data bits is a 64-tap absolute delay line. This 64-tap absolute delay line can be implemented using the IDELAY and IDELAYCTRL primitives. Both the clock/strobe and the data bits are routed through the 64-tap absolute delay line. Although the strobe is not used to capture data, it is used to determine the number of taps required to center the data with respect to the internal FPGA clock. The design and implementation details of the direct clocking scheme are explained in the following sections.

## Strobe Edge Detection

The delay value for the data bits associated with a clock/strobe is the phase difference between the rising edge of internal FPGA clock and the center of the clock/strobe pulse. The assumption is that clock/strobe and data are edge aligned. In order to determine this phase difference, the clock/strobe is input through the 64-tap absolute delay line in the IOB and is sampled at incremental tap outputs using the internal FPGA clock.

At least two edges or transitions of the clock/strobe have to be detected to determine the center of the clock/strobe pulse. The difference between the number of taps required for detection of the second transition (second edge taps), and the number of taps required for detection of the first transition (first edge taps) is the clock/strobe pulse width. Half of this difference is the pulse center (pulse center taps). The number of taps required from the rising edge of internal FPGA clock to the center of the clock/strobe pulse is the sum of first edge taps and pulse-center taps.

- Number of taps required to detect first transition of clock/strobe = first-edge taps
- Number of taps required to detect second transition of clock/strobe = second-edge taps
- Pulse width of clock/strobe = (second-edge taps – first-edge taps)
- Pulse-center taps = Pulse width of clock/strobe divided by two
- Number of taps required to center data with internal FPGA clock (data-delay taps) = first-edge taps + pulse-center taps

Figure 1 illustrates two scenarios of centering data with respect to the internal FPGA clock by delaying it by the data delay taps value. Case 1 shows the falling edge of clock/strobe as the first edge being detected and this results in the delayed data being centered on the rising edge of the internal FPGA clock. Case 2 shows the rising edge of clock/strobe as the first edge being detected and this results in the delayed data being centered on the falling edge of the internal FPGA clock.

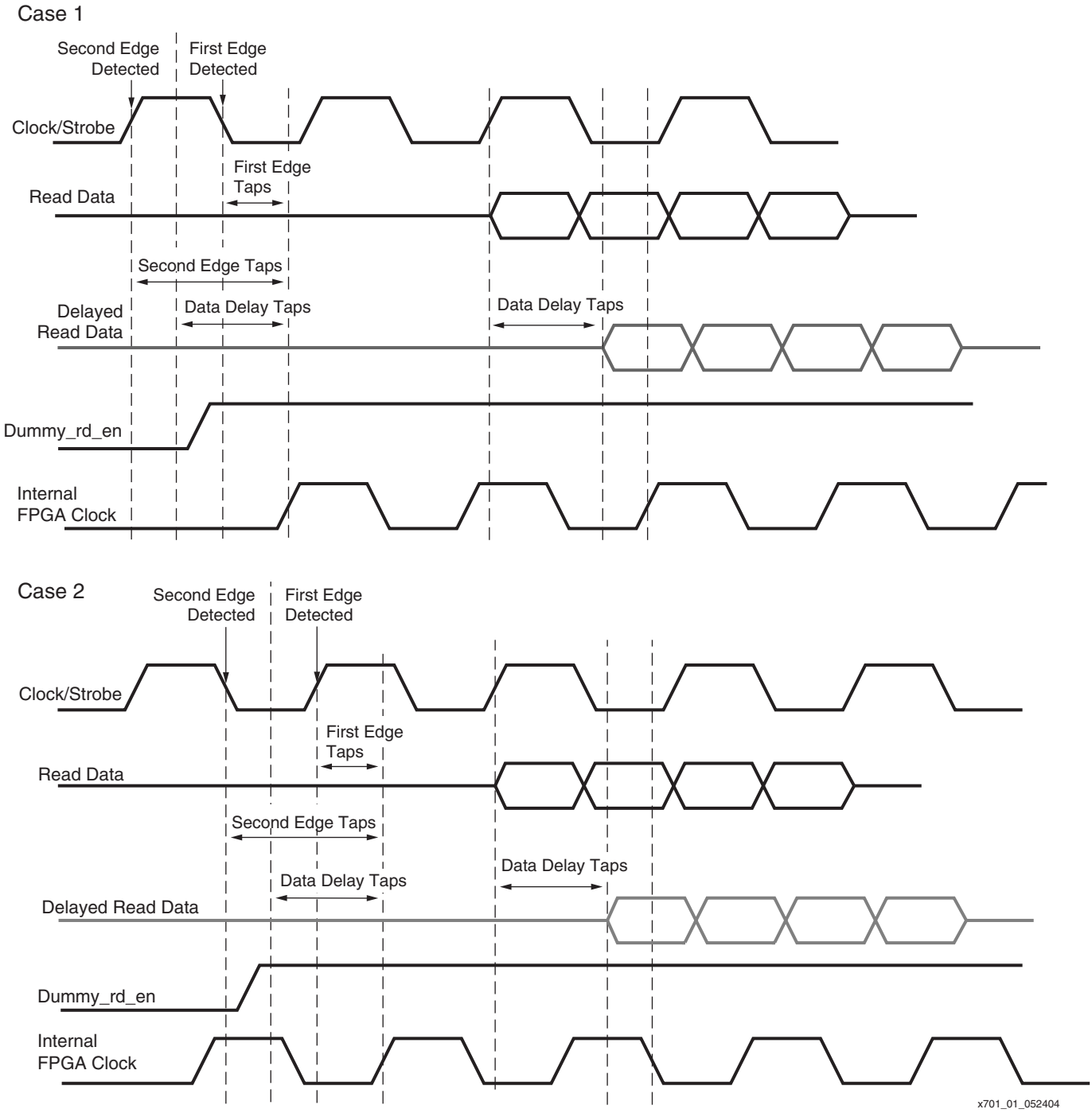


*Figure 1:* **Case 1 and Case 2 - Clock/Strobe Center to Internal FPGA Clock Phase Detection**

## Strobe Edge Detection Implementation

The implementation of the delay value determination circuit in a Virtex-4 device is easy because of the dedicated IDELAY and IDELAY_CTRL circuits. The block diagram of the implementation of the delay value determination scheme is shown in Figure 2.

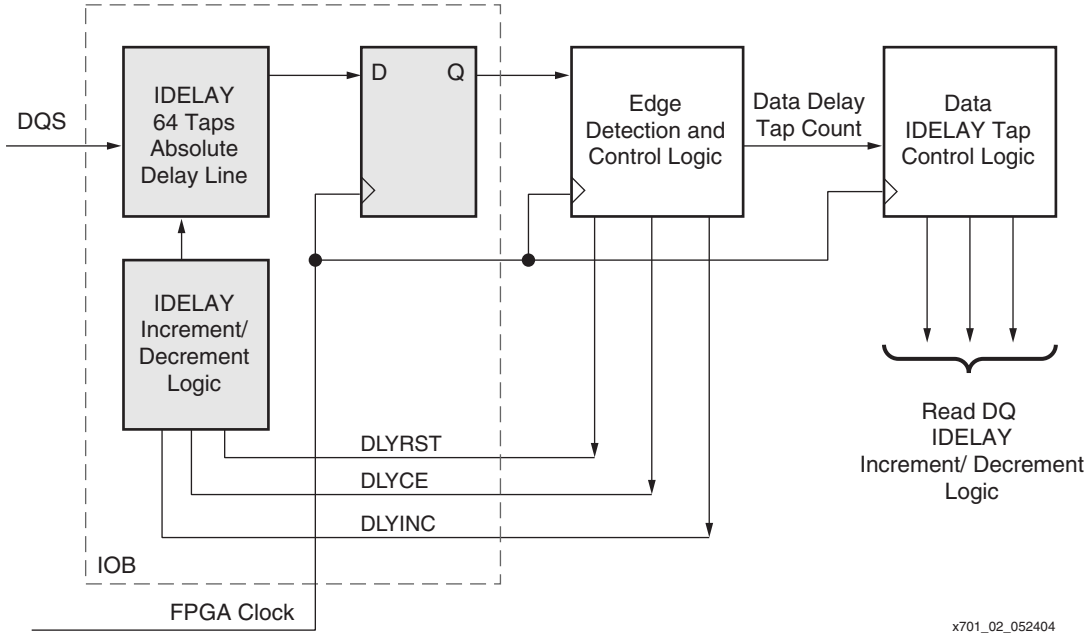


*Figure 2:* **Strobe Edge Detection**

A simple algorithm is used for detecting edges of memory clock/strobe. The clock/strobe is input to the IDELAY block with an initial value of 0. The clock/strobe is delayed in one tap increments until the first edge is detected. The number of taps required to detect the first edge is then recorded. The clock/strobe continue to be delayed in one tap increments until the second edge is detected. The number of taps required to detect the second edge is then recorded. The pulse width is computed using both recorded values. Once the pulse width of the clock/strobe is determined in number of taps, the midpoint is obtained by dividing it by two. The sum of the midpoint and the number of taps required to detect the first edge is the required taps to delay data.

The total number of taps available in the IDELAY block is 64. Therefore for a frequency of 200 MHz and below, it is not possible to detect two edges. At the end of 64 taps, if only one edge is detected, the number of taps required to delay data is the sum of the number of taps required to detect the first edge and 16 taps (~1.25 ns with a tap resolution of ~80 ps). Quarter cycle of a 200 MHz clock/strobe is about 16 taps. Based on timing analysis, this value can also be used for lower frequencies down to 110 MHz. For frequencies below 110 MHz, if no edges are detected at the end of 64 taps the number of taps required to delay data is 32 taps (~2.5 ns with a tap resolution of ~80 ps). This value is sufficient to set the internal FPGA clock edge within the data window.

Only a small state machine is required for the first and second edge detection. This state machine is only enabled during a dummy read operation issued for data delay tap value determination. A dummy read operation comprised of multiple, back-to-back read commands is issued to the external memory device before normal operation.The state machine controls the inputs to the IDELAY circuit namely: DLYRST, DLYCE, and DLYINC.

DLYRST - The delay line reset signal that resets the number of taps in the delay line to a value set by the IOBDELAY_VALUE attribute. Set to "0" in this design.

DLYCE - The delay line enable signal that determines when the delay line increment/decrement signal is activated.

DLYINC - The delay line increment/decrement signal that increments or decrements the number of taps in the delay block. Table 1 describes the operation of the delay line.

*Table 1:* **Delay Block Operation**

| Operation | DLYRST | DLYCE | DLYINC |
|---|---|---|---|
| Reset to configured value of tap count | 1 | X | X |
| Increment tap count | 0 | 1 | 1 |
| Decrement tap count | 0 | 1 | 0 |
| No Change | 0 | 0 | X |

The state diagram to control these delay block inputs is shown in Figure 3. The four states in this state machine are: DELAY_RST, IDLE, DELAY_INC, and DETECT_EDGE.

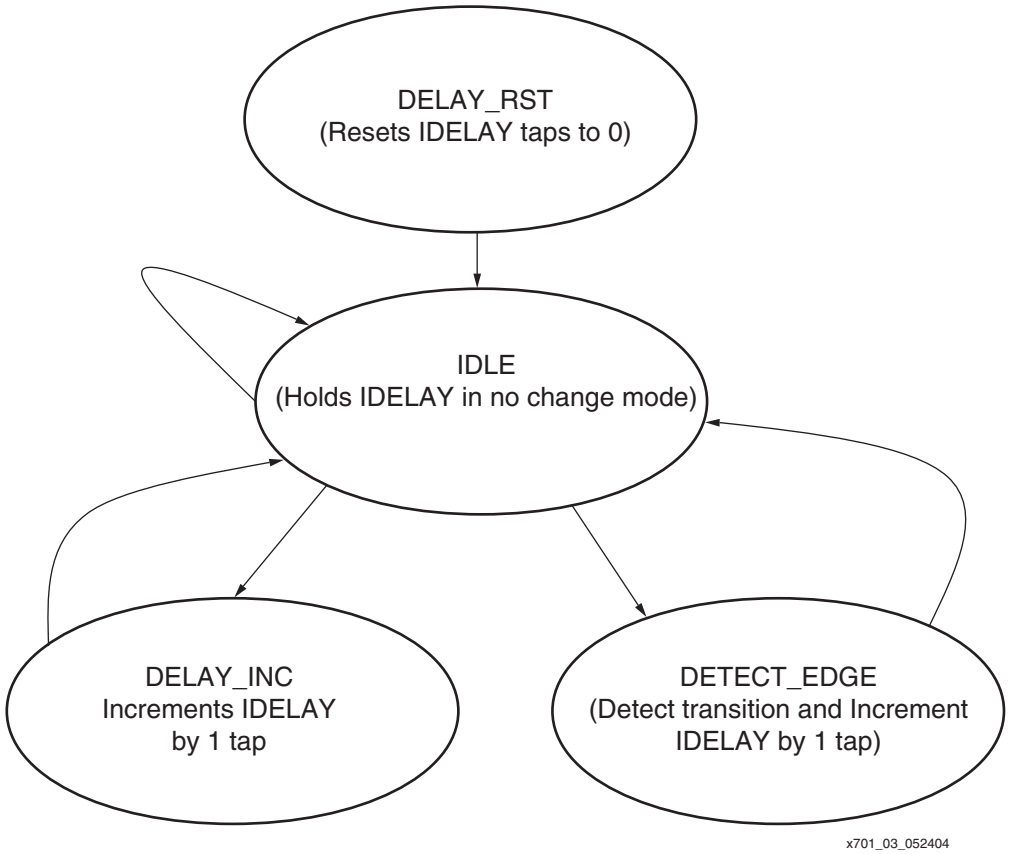


*Figure 3:* **Strobe Edge Detection State Diagram**

### DELAY_RST

This is the first state in the state machine enabled with the start of the dummy read operation. In this state the delay block is reset to "0" taps. This state is followed by multiple IDLE states.

### IDLE

In this state the delay block is maintained in No change operation. Every state other than IDLE is followed by multiple IDLE states. This is done to allow the tap output value to settle. This state is followed by either another IDLE, or DELAY_INC, or DETECT_EDGE state.

### DELAY_INC

This state increments the tap of the delay block by one. This state is followed by multiple IDLE states.

### DETECT_EDGE

In this state, the output of the delay block is compared with its previous value to detect an edge or transition and increments the delay block tap by one. This state is followed by multiple IDLE states.

After the number of taps to delay the data is determined, the data IDELAY circuit is enabled and increments to this value. This is done by incrementing the data IDELAY circuit for the same number of clock cycles as the number of taps required. The block diagram of the read/write data path with the data IDELAY circuit is shown in Figure 4.

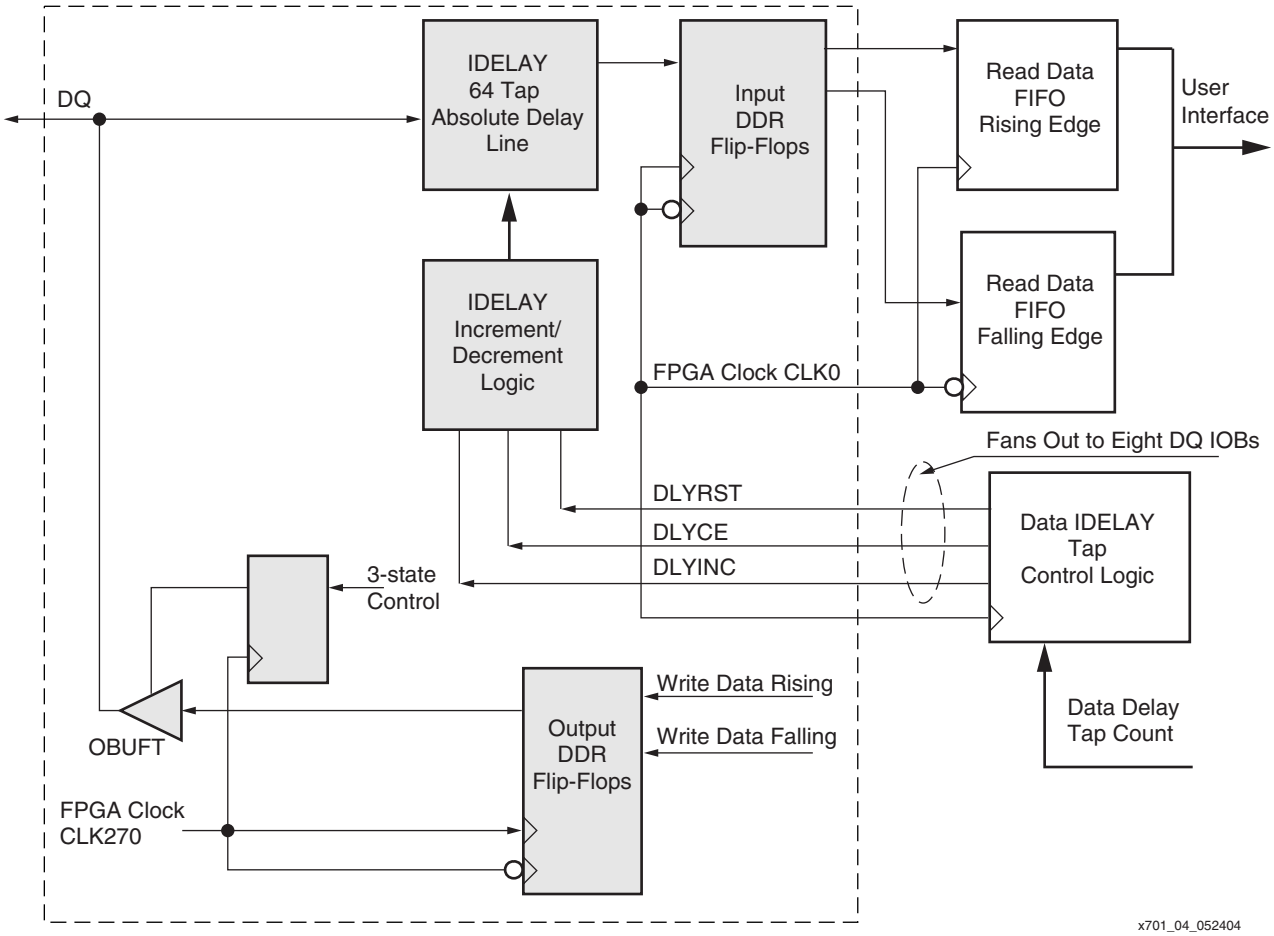


*Figure 4:* **Read/Write Data Path**

## Data Capture and Re-capture

The delayed data is captured in the input DDR flip-flops using the internal FPGA clock as shown in Figure 4. The outputs of these flip-flops are then stored in two FIFOs; one for rising edge data and the other for falling edge data. These FIFOs are implemented using the LUT RAMs. The write enable for these FIFOs is provided by a read enable signal normalized for system parameters. The read enable signal also goes through the 64-tap delay line with the same number of tap delays as the data bits.

The DDR2 SDRAM memory devices do not provide a read valid or read enable signal along with read data. Therefore, the controller generates this read enable signal based on the CAS latency and the burst length. The read enable signal must be asserted during the read preamble and de-asserted after the last rising edge of the strobe. The read enable signal is normalized to make it system independent. The normalization is implemented with a loop back on the PCB. The READ_EN_OUT is output from the FPGA, loops back on the PCB and is input as READ_EN_IN. The trace delay of this loop back must equal the sum of trace delays of clock (CK/$\overline{CK}$) forwarded to the memory device and the strobe (DQS)/data (DQ). The trace delays of CK, DQSs, and DQs must be closely matched. This loop back signal is used to generate write enable signals to the read data capture FIFOs. For interfaces that span multiple banks, a loop back is recommended per bank in order to manage the fan out on this enable signal.

The first data word can be captured using either the rising edge or the falling edge of the internal FPGA clock. Therefore, additional logic is required for the write enable of the rising edge FIFO. The circuit implementing the write enable logic for the read re-capture FIFOs is shown in Figure 5. If the first data is captured on the rising edge of the FPGA clock, then the write enable to rising edge FIFO is the output of the first flip-flop. If not, it is the output of the second flip-flop. The timing diagram for the read data capture and write enable for re-capture FIFOs is shown in Figure 6.

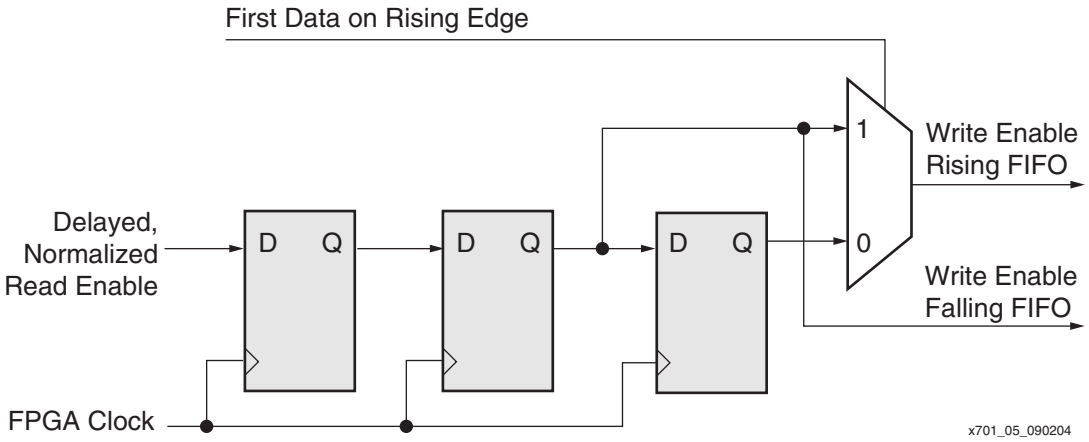


*Figure 5:* **Write Enable Logic for Read Re-capture FIFOs**

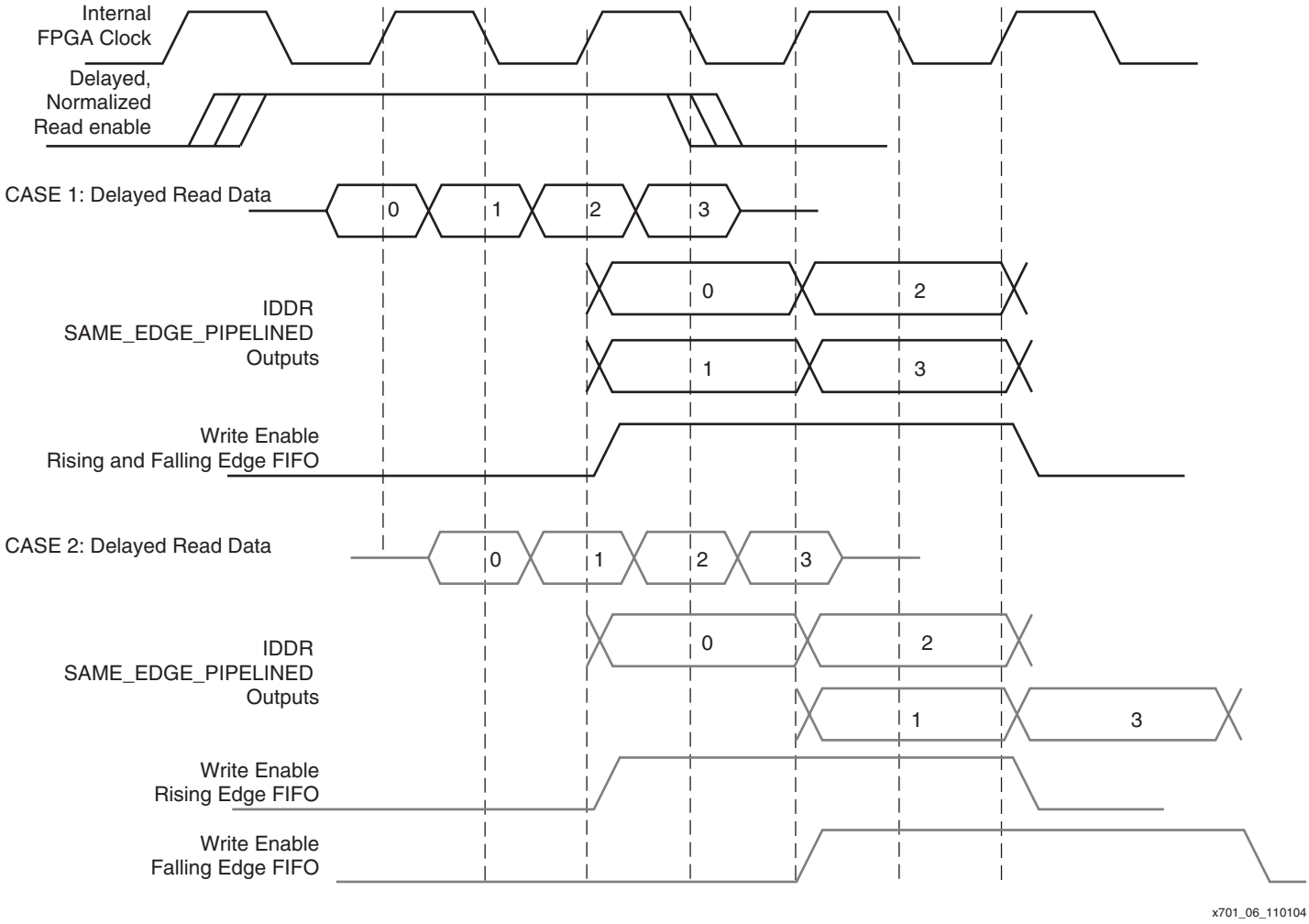


*Figure 6:* **Data Capture and Transfer to FIFOs**

## Read Timing Analysis

Read timing analysis with the direct clocking technique is described in this section. Read data is captured directly in the FPGA clock domain, therefore the memory parameter used for the data valid window analysis is the access time ($T_{AC}$). The following is a brief description of each parameter used in this timing analysis.

External memory parameters considered for this timing analysis are:

- $T_{AC}$ - Access time of read data (DQ) with respect to clock forwarded to memory by FPGA
- $T_{MEM\_DCD}$ - Duty cycle distortion tolerance specified by memory vendor

Read data (DQ) is captured using the FPGA clock and not the memory clock/strobe (DQS), therefore $T_{AC}$ (access time of data with respect to clock) is considered for this analysis. The DQS to DQ memory parameters such as $T_{DQSQ}$, and $T_{QHS}$ are not considered in this analysis since $T_{AC}$ overrides them.

FPGA parameters considered for this timing analysis are:

- $T_{GLOBAL\_CLOCK\_TREE-SKEW}$ - Skew on the global clock tree
- $T_{JITTER}$ - DCM clock output jitter
- $T_{PACKAGE\_SKEW}$ - Package skew for a particular device/package

The FPGA parameters not considered in this analysis are:

- $T_{SETUP}$ - Setup time of the flip-flop I/O
- $T_{HOLD}$ - Hold time of the flip-flop I/O

The delay on data bits associated with a DQS is computed by detecting the DQS edge. Capturing the DQS in an I/O flip-flop using the global clock performs this detection. The final delay value for the data therefore already takes into account the setup and hold time of the I/O flip-flop. Hence, $T_{SETUP}$ and $T_{HOLD}$ are not considered in this analysis.

PCB layout skew is also considered to account for the skew between data bits and the associated strobe.

Table 2 shows the read timing analysis at 267 MHz for a DDR-II interface. All the parameters are specified in picoseconds. $T_{DATA\_PERIOD}$ is half the clock period minus $T_{MEM\_DCD}$. The sum of uncertainties before clock is the start of the valid data window (685 ps). The difference between $T_{DATA\_PERIOD}$ and the sum of uncertainties after clock is the end of the valid data window (1002 ps). This results in a 317 ps margin at 267 MHz. The uncertainly in the number of taps selected is a tap resolution. There is sufficient margin even after accounting for this uncertainty.

*Table 2:* **Read Timing Analysis at 267 MHz for a DDR-2 Interface**

| Uncertainty Parameters | Value (ps) | Uncertainties Before Clock | Uncertainties After Clock | Description |
|---|---|---|---|---|
| $T_{CLOCK}$ | 3750 | | | Clock period |
| $T_{MEM\_DCD}$ | 188 | 0 | 0 | Duty cycle distortion tolerance is subtracted from clock phase (equal to half clock period) to determine $T_{DATA\_PERIOD}$. |
| $T_{DATA\_PERIOD}$ | 1687 | | | Data period is half the clock period with 10% duty cycle distortion subtracted from it. |

*Table 2:* **Read Timing Analysis at 267 MHz for a DDR-2 Interface *(Continued)***

| Uncertainty Parameters | Value (ps) | Uncertainties Before Clock | Uncertainties After Clock | Description |
|---|---|---|---|---|
| Memory uncertainties ($T_{AC}$) | 500 | 500 | 500 | This parameter considers the worst of all the memory parameters since there is overlap between these parameters ($T_{DQSQ}$, $T_{QHS}$, $T_{DQSCK}$, $T_{AC}$). $T_{AC}$ = 500 ps for 267 MHz. |
| $T_{DQSQ}$ | 0 | 0 | 0 | Due to overlap with other memory parameters. Only the worst case parameter $T_{AC}$ is considered. |
| $T_{QHS}$ | 0 | 0 | 0 | Due to overlap with other memory parameters. Only the worst case parameter $T_{AC}$ is considered. |
| $T_{PACKAGE\_SKEW}$ | 30 | 15 | 15 | Package skew |
| $T_{SETUP}$ - Minimum | 0 | 0 | 0 | DQS edge detection is performed by registering it in the I/O flip-flop with a global clock. The final data delay value therefore already accounts for the setup and hold times of the I/O flip-flops. Hence these parameters are not considered in this analysis. |
| $T_{HOLD}$ - Maximum | 0 | 0 | 0 | |
| $T_{JITTER}$ | 100 | 100 | 100 | Clock jitter that indirectly causes strobe and data jitter. |
| $T_{CLOCK\_TREE\_SKEW}$ - Maximum | 50 | 50 | 50 | Small value considered for Skew on "global clock" line since detection of DQS and associated DQ are placed close to each other |
| $T_{PCB\_LAYOUT\_SKEW}$ | 20 | 20 | 20 | Skew between data lines and associated strobe on the board |
| Uncertainties | 387 | 685 | 685 | |
| Window | 317 | 685 | 1002 | |

Figure 7 illustrates the calculated data valid window.

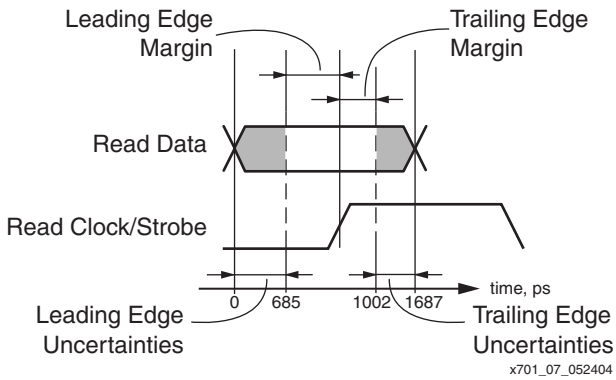


*Figure 7:* **Data Valid Window**

## Reference Design

The reference design is available on the Xilinx web site at:

http://www.xilinx.com/bvdocs/appnotes/xapp702.zip

## Conclusion

The Virtex-4 I/O architecture enhances the implementation of source-synchronous memory interfaces. The architectural features used in this application note and reference design include:

- IDELAY block – Continuously calibrated delay elements with small tap resolution.
- FIFO16 primitive – Block RAM used as FIFO with no additional CLB resources required for status flag generation.
- High-speed differential global clocking resources provide better duty cycle. The number of global clock resources required in a design is reduced as a result of differential clocking.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 09/09/04 | 1.0 | Initial Xilinx release. |
| 11/01/04 | 1.1 | Revised description under “Data Capture and Re-capture” section. Revised Figure 6. Reference design is updated on web. |