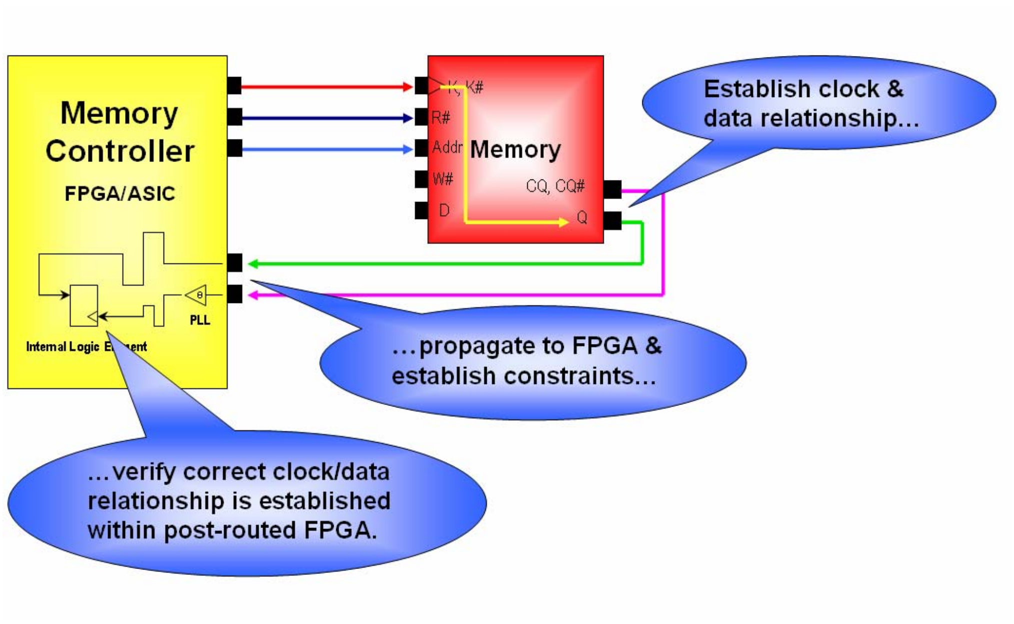
WHITE PAPER:

HOW TO EFFECTIVELY MANAGE TIMING OF FPGA DESIGN FLOWS

Integrating EMA TimingDesigner with Xilinx and Altera Development Systems

Rev. 04/07



Memory Controller
FPGA/ASIC
PLL
Internal Logic Element
K, K#
R#
Addr
W#
D
Memory
CQ, CQ#
Q
Establish clock & data relationship…
…propagate to FPGA & establish constraints…
…verify correct clock/data relationship is established within post-routed FPGA.

EMA Design Automation™

Table of Contents

Table of Figures

1. Abstract

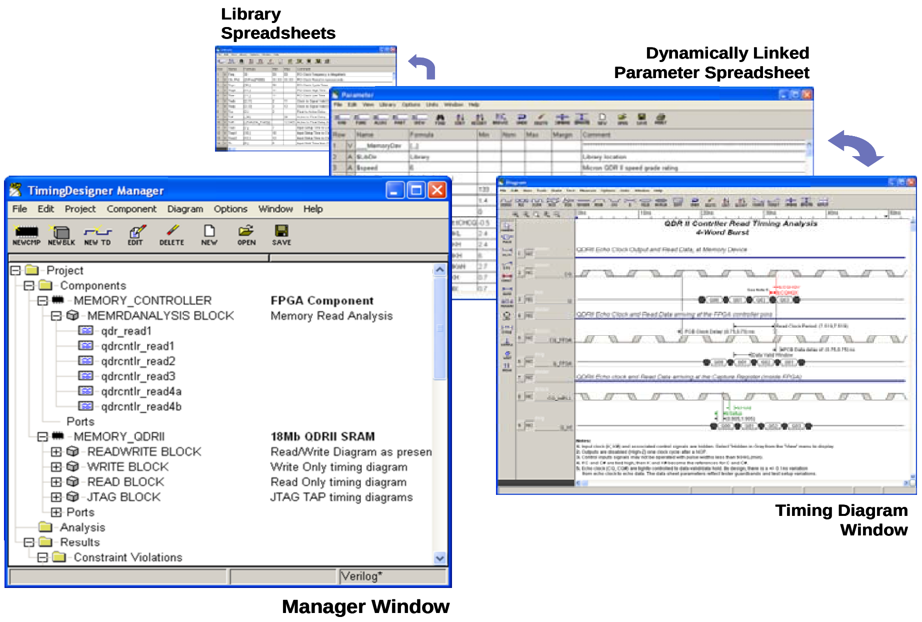
When combined with advances in FPGA technologies for interface design efforts, EMA TimingDesigner® can simplify design issues and provide advanced accurate control of virtually any interface. From simple SRAM interface protocols to high-speed source synchronous interface protocols, TimingDesigner allows designers to identify potential timing problems early in the design process and thereby providing the greatest opportunity to get the timing right the first time.

Detecting timing problems early in the design process not only saves time but also permits much easier implementation of design alternatives. TimingDesigner, from EMA Design Automation, allows you to create interactive timing diagrams for capturing interface specifications, analyze component interface timing characteristics, and communicate design requirements among project engineering teams. In addition, TimingDesigner contains features that allow exchange of critical timing data with Xilinx ISE™ and Altera Quartus® II tool sets, throughout the design process. TimingDesigner can communicate place-and-route constraints that reference design specific timing measurements, and allows direct use of post place-and-route timing information to provide visual verification of the interface signal relationships required for desired FPGA interface operation. Integration with other FPGA tool sets such ispLEVER™ from Lattice®, Designer from Actel®, and QuickWorks® from QuickLogic® is also planned with future releases of TimingDesigner.

2. Introduction

Designing FPGAs with the high-speed interface technologies available today helps you meet market demands, but it also presents some interesting design challenges. To ensure accurate data transfer for memory interfaces that operate at 200 MHz and beyond, timing analysis needs to play a more prominent role in the identification and resolution of system operation issues. At these frequencies, the margins for setup and hold times are tight, leaving minimal room to secure an accurate data capture and presentation window. Faster edge rates also magnify physical design effects, which cause signal integrity issues that require additional settling time, shrinking timing margins further.

FPGA devices now include advanced features that directly support Double Data Rate (DDR) interface technology within the I/O blocks, and on-board phase locked loop (PLL) networks for accurate clock control. These advances in FPGA technology help reduce the interface design effort by providing advanced building blocks that are explicitly designed for these advanced interfaces, and when combined with the unique capabilities of TimingDesigner, provide a powerful solution that can provide that most accurate solution in the least amount of time. This white paper explores a technique for determining necessary clock skew for balanced read data capture margins with DDR type memory interface designs.


Library Spreadsheets
Dynamically Linked Parameter Spreadsheet
TimingDesigner Manager
File Edit Project Component Diagram Options Window Help
NEWCMP NEWBLK NEW TD EDIT DELETE NEW OPEN SAVE
Verilog*
Timing Diagram Window
Manager Window

```
Project
  Components
    MEMORY_CONTROLLER                 FPGA Component
      MEMRDANALYSIS BLOCK             Memory Read Analysis
        qdr_read1
        qdrcntlr_read1
        qdrcntlr_read2
        qdrcntlr_read3
        qdrcntlr_read4a
        qdrcntlr_read4b
      Ports
    MEMORY_QDRII                      18Mb QDRII SRAM
      READWRITE BLOCK                 Read/Write Diagram as presen
      WRITE BLOCK                     Write Only timing diagram
      READ BLOCK                      Read Only timing diagram
      JTAG BLOCK                      JTAG TAP timing diagrams
      Ports
  Analysis
  Results
    Constraint Violations
```

Figure 1 - TimingDesigner's GUI windows allow easy capture of design interface characteristics

3. DDR/QDR Memory Interface Design Problem

DDR and/or Quad Data Rate™(QDR™) memory devices provide and accept source-synchronous data at twice the rate of the device's clock frequency. This means that data is transferred on the rising and falling edges of the capture clock. In addition, these devices require capture clock skew adjustments to ensure proper clock/data relationships. As noted earlier, several FPGA devices now include DDR interface technology support within I/O blocks and on-board PLL networks. In using these advanced building blocks, you need to conform to memory design requirements. This means that you have to have a way to manipulate the blocks accurately and reliably. To illustrate this point, let's take a look at a design requirement for a read operation with a QDR II SRAM source-synchronous interface.

In synchronous memory systems such as QDR SRAM, data is presented coincident with a provided clock, so clock derivatives must be created that are shifted by 90 degrees in order to safely latch memory data. This phase shifting is commonly referred to as center-alignment of the clock within the data valid window and is an important QDR design characteristic for accurate data capture (see Figure 2 below). To shift the clock for a center-alignment, we can simply delay the clock signal by phase shifting using the PLL network on board the FPGA.

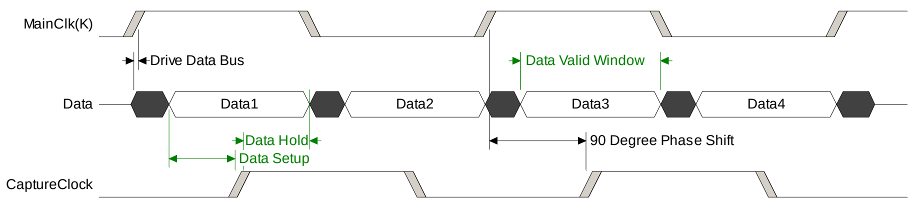


Figure 2 – Illustration of center-aligned clock/data relationship

Capturing Read Data

Delaying the clock signal to achieve center-alignment ensures that various temperature changes and other similar effects the design may encounter won't cause an excessive amount of shift in clock/data position and therefore violate the setup or hold time requirements of the receiving register. In theory, a center-aligned clock edge will maximize the setup and hold times for most devices, allowing sufficient safety margins for drift. However, unless the setup requirement is equal to the hold requirement, center-alignment of the clock signal will provide more margin for one than the other.

The ideal solution is to provide a maximum safety margin for both setup and hold requirements of the device, which translates to balancing the margins, providing equal amounts of safety for both. To balance the margins, we determine the minimum data valid window for the receiving device, and center that window within the actual data valid window provided from the memory device given our design parameters.

Using the minimum setup and hold characteristics of our receiving device, we determine a minimum "safe" data valid window with the following formula:

```
Minimum Setup + Minimum Hold = Minimum Data Valid Window
```

The resulting data valid window is centered within the actual data valid window provided by the memory device, as shown in Figure 3. To ensure data capture, the data bus must transition within the indicated "safe" regions outside of the receiver's minimum data valid window. With this clock/data relationship, we ensure the maximum possible safety margin for read data capture when the design experiences signal drift in either direction.

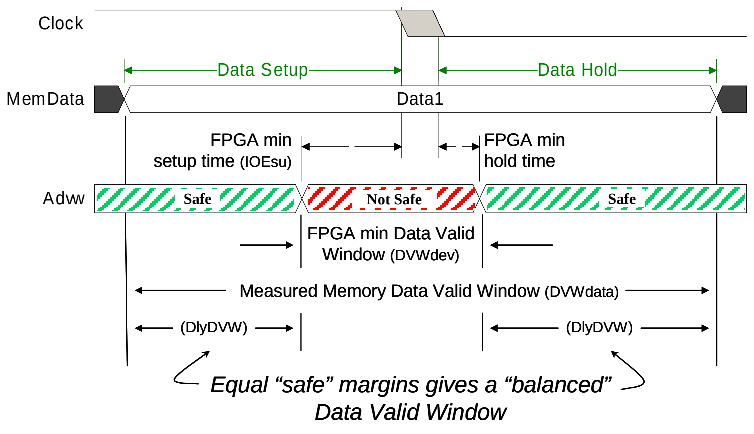

Clock
Data Setup
Data Hold
MemData
Data1
FPGA min setup time (IOEsu)
FPGA min hold time
Adw
Safe
Not Safe
Safe
FPGA min Data Valid Window (DVWdev)
Measured Memory Data Valid Window (DVWdata)
(DlyDVW)
(DlyDVW)
Equal "safe" margins gives a "balanced" Data Valid Window

Figure 3 – Balancing the minimum Data Valid Window within actual Data Valid Window

Achieving Proper Clock Skew

Skewing the source-synchronous clock will effectively shift the minimum data valid window of the receiving registers in the memory controller, and will therefore serve as the mechanism for balancing the data valid window. Clock skew adjustments are made with one of the PLL components inside the FPGA device. To determine the skew value, we must take into account routing delays and any external delay mechanisms that will affect the signal relationships.

We start by using TimingDesigner to create a diagram for the read operation of the QDR SRAM directly from the memory datasheet (Figure 4). We use this diagram to determine the clock and data signal timing relationships as they appear at the pins of the memory device as well as the data valid window characteristics of the design. The objective is to begin at a point where the signal relationship is well defined (the memory device), and propagate that relationship across the PCB to the FPGA where the unknowns begin to have an impact.

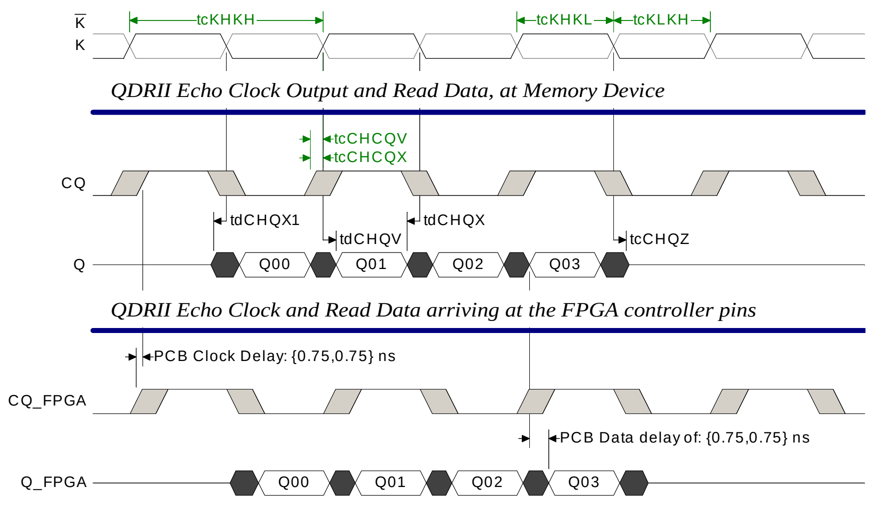

K̄
K
tcKHKH
tcKHKL
tcKLKH
QDRII Echo Clock Output and Read Data, at Memory Device
tcCHCQV
tcCHCQX
CQ
tdCHQX1
tdCHQV
tdCHQX
tcCHQZ
Q
Q00
Q01
Q02
Q03
QDRII Echo Clock and Read Data arriving at the FPGA controller pins
PCB Clock Delay: {0.75,0.75} ns
CQ_FPGA
PCB Data delay of: {0.75,0.75} ns
Q_FPGA

Figure 4 – QDR Memory Read timing diagram for MT54W1MH18J

Figure 4 shows how the PCB propagation delay is accounted for in the clock (CQ_FPGA) and the data (Q_FPGA) signals as they appear at the pins of the FPGA. Using separate variables in TimingDesigner's dynamically linked parameter spreadsheet allows easy variation of PCB delay values while showing the effect on the associated signals. We can now fit the design into the FPGA device to obtain the internal routing delays and determine the correct phase shift necessary for the capture clock.

4. Utilizing FPGA Design Elements

Most FPGA implementations make use of constraint-driven place-and-route. Timing constraints provide desired timing report information on specific signals. TimingDesigner offers a unique feature designed specifically for referencing timing diagram information such as measurements and variable calculation results, from within lines of text adhering to vendor specific constraint syntax. For example, measured delay tolerances for a particular signal are referenced in a timing diagram within the Dynamic Text Window that conforms to syntax required for FPGA constraints for routing. Then we can either save the resolved syntax to a text file for import into the FPGA development system, or we can copy the values directly into the FPGA constraint editor.

For high-speed memory interface design, the read data capture registers are placed into the I/O elements of the FPGA device to minimize route delay impact. The I/O blocks have only one routing path for input data signals so the data path delay is consistent for each respective bit of the data bus. The PLLs of the FPGA are also utilized for proper clock control and typically have several possible paths from the input pad to the capture register. PLL characteristics such as phase shift, multiply, and divide-by factors are controlled through manufacturer specific attributes, either in primitive instantiations of the design code or through a constraint entry of some form. As a result, the routing and element delays for the clock and data paths must be determined in order to achieve proper phase shifting of the clock.

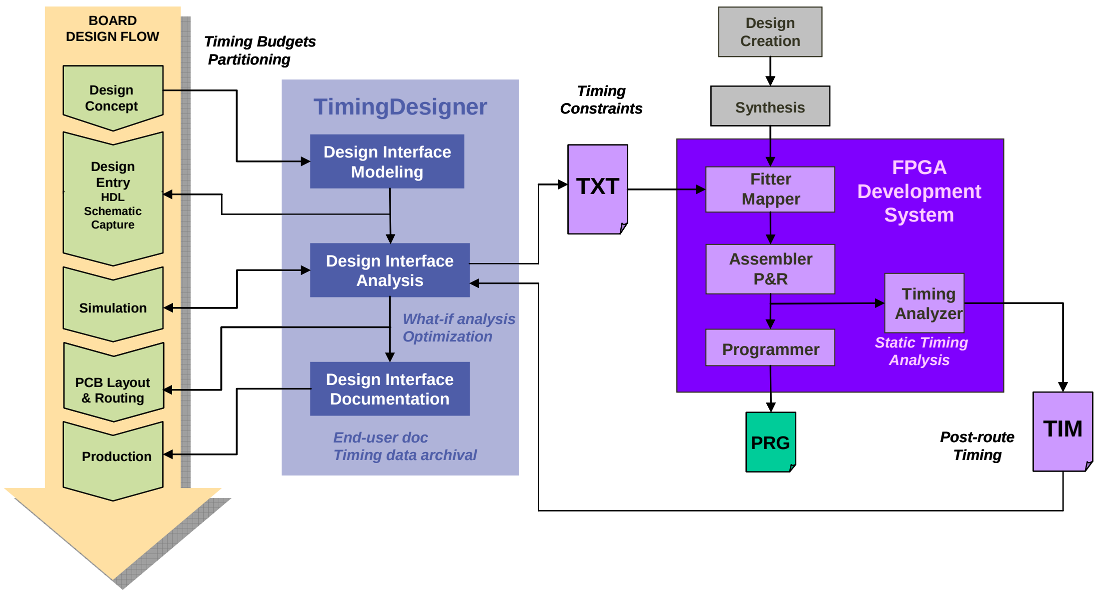


Figure 5 - TimingDesigner offers intuitive interface to FPGA design flow.

After the initial place-and-route of the FPGA is complete, a timing report provides delay information details of each timing path for the data bus. If necessary, the FPGA development system can be configured to report delay paths associated with specific signals of interest, and TimingDesigner can extract that information and utilize it for diagram updates. In this design example, we need to report the timing paths of our input data bus and its associated clocking signal.

Importing Post-Route Timing to TimingDesigner

To import FPGA timing report information, we need to map the worst-case setup path identified in the report to the associated waveform in the diagram (Q_FPGA). Signals in the design code may not be named the same as the waveforms in the timing diagram. By mapping the ports, TimingDesigner can filter the timing report and extract only the information of interest. This mapping is stored in the diagram file and pre-filled for successive place-and-route executions.

Providing Visualization at the Capture Register

From imported timing report files, TimingDesigner creates variables for all associated signal delay elements in the parameter spreadsheet from the mapped port/signal entries. The variables are used to update the clock/data relationships in the timing diagram. Now, edge relationships can be determined as they arrive at the internal capture register within the FPGA device.

The internal register setup and hold requirements are extracted from the timing report and relevant constraints applied to the diagram. Next, two more signals are added to the timing diagram, skewed by the amount of routing delay reported in the timing report. The data and clock are now represented as they appear at that capture register, and then the setup and hold constraints of the FPGA device are applied. The offset between the latching clock edge and the leading edge of the data valid window is measured to determine phase shift necessary to balance the data valid window in the design.

Balancing the Data Valid Window

We can use the following procedure to determine the amount of PLL generated phase shift needed for the clock signal:

1. Subtract the minimum data valid window for the I/O elements of the FPGA device, from the design's actual data valid window, and then divide that result by two. This accounts for the difference between the two data valid windows (DlyDVW). Reference Figure 3.

```
DlyDVW = (DVWdata – DVWdev) / 2
```

2. Add the DlyDVW value to the required data setup time for the I/O element registers. This determines the location for the clock edge relative to the data valid window (DlyRelSU).

```
DlyRelSU = DlyDVW + IOEsu
```

3. Finally, subtract the offset measurement of the clock signal relative to the data valid window as it appears at the capture register (obtained from the timing diagram measurement), from the relative setup time (value obtained in 2 above).

```
Clk_offset = DlyRelSU – EdgeOffset
```

Using the above formulas and procedures, we determine the necessary PLL phase shift to be entered into the FPGA development system, and execute a second place-and-route.

Verifying Results

Simply re-importing the new post-route timing file with the previous import settings will automatically update all the necessary values within TimingDesigner, and correctly re-position the I/O element clock signal CQ_intPLL. This is illustrated in Figure 6. Depending on the resolution of the shift capacity of the PLL element, an exact balance for the setup and hold margins may not be possible. For these situations, you should be able to achieve a balance within the incremental resolution of the PLL of the FPGA device.

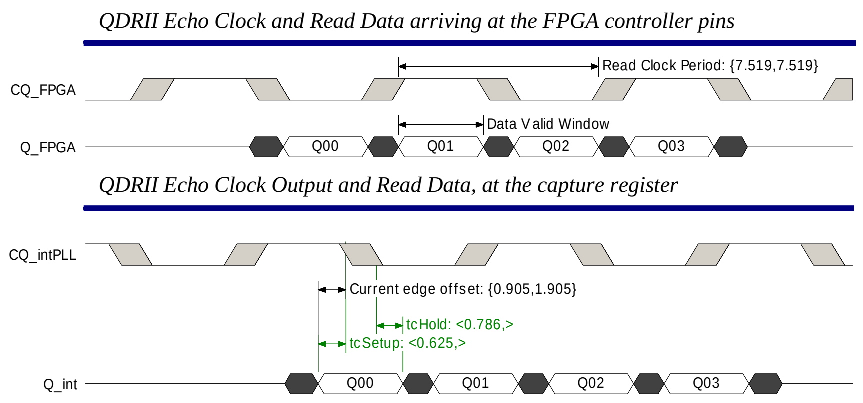

QDRII Echo Clock and Read Data arriving at the FPGA controller pins
Read Clock Period: {7.519,7.519}
CQ_FPGA
Data Valid Window
Q_FPGA
Q00
Q01
Q02
Q03
QDRII Echo Clock Output and Read Data, at the capture register
CQ_intPLL
Current edge offset: {0.905,1.905}
tcHold: <0.786,>
tcSetup: <0.625,>
Q_int
Q00
Q01
Q02
Q03

Figure 6 – Final timing diagram for data capture analysis with shifted clock and balanced setup/hold margin.

5. Summary

High-speed designs often have stringent specifications and tight release schedules, so there's a need for an interactive timing specification and analysis tool to obtain fast and complete timing margin analyses to address all of the factors that can ultimately affect design success. This white paper illustrates how TimingDesigner can be used to accurately capture and exchange timing information within FPGA design flow to help manage timing margins throughout the design process and provide visual verification that your design performs as expected. The versatile clock configurations and abundant I/O resources of today's FPGA device families, allow high throughput data transfer and TimingDesigner delivers accurate critical path timing analysis results required for interface design with high-speed memory devices such as DDR & QDR SRAM.

EMA Design Automation
Corporate Headquarters
225 Tech Park Drive
Rochester, New York 14623

Phone (within North America): 800.813.7494
Phone (outside North America): 585.334.6001
Fax: 585.334.6693
www.ema-eda.com
www.timingdesigner.com
info@ema-eda.com