XAPP768c (v2.0) October 14, 2004

Interfacing Spartan–3 Devices With 166 MHz or 333 Mb/s DDR SDRAM Memories

Author: Karthikeyan Palanisamy

Summary

This application note describes how (using minimal resources) to use any side of a Xilinx Spartan-3 device to create an interface to a double data rate (DDR) SDRAM device. The techniques described herein are similar to those described in application note XAPP758c. The examples in this application note are based on a DDR interface between an XC3S1500-5FG456 Spartan-3 device and a Micron MT46v32M16TG-6T device. The design concepts are applicable to other types of memories and other Spartan-3 devices. Interface performance is up to 166 MHz or 333 Mb/s

Theory of Operation

High-speed double data rate (DDR) SDRAM interfaces are source synchronous. In a source-synchronous design, clocks are generated and transmitted along with the data. The memory interface receives the data using the received clock, and then transfers the data into the receiver's clock domain.

The memory interface described in this application note is part of a system as shown in Figure 1. The memory interface has a 72-bit data bus with a data path capable of receiving and transmitting 144-bit data every clock cycle. This is because the double data rate interface requires data on both the positive and the negative edges of the clock.

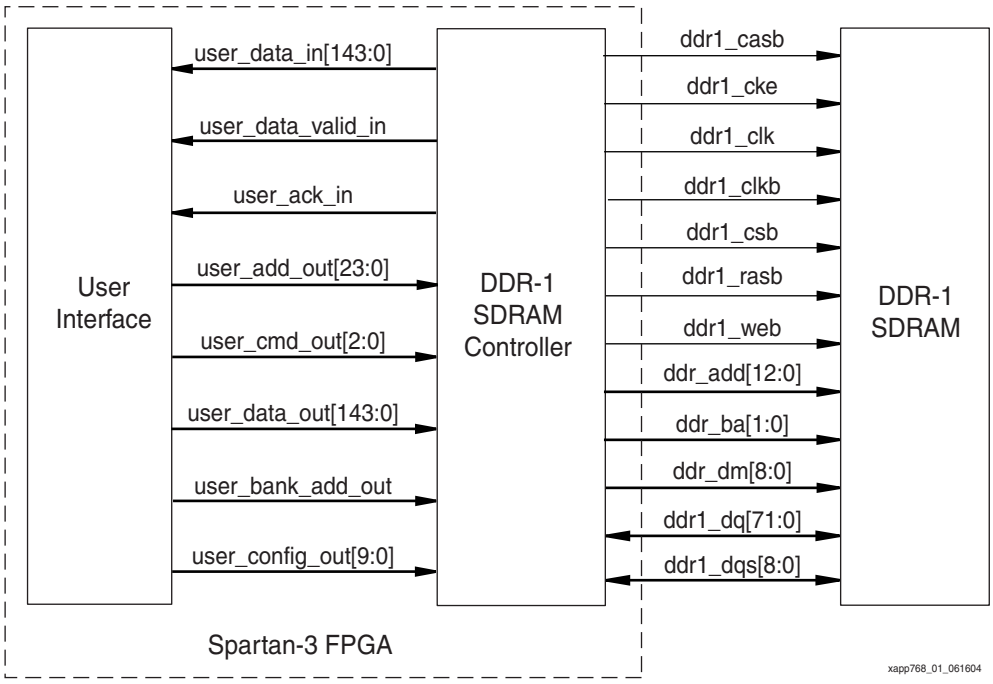

Figure 1: **System Diagram**

The data bus can be implemented using any of the four sides of the FPGA, and the address and control signals can either use the same side as the data bus, or an adjacent bank. All address and control signals are compatible with SSTL_II standards, and the data is SSTL_II class II compatible.

Signals from the user interface are labeled with a "user_" prefix. For more information on the user interface, refer to the *Xilinx Memory Interface Generator (MIG 007) User Guide* (UG067).

Key Challenges

High-speed memory interfaces are challenging to design, due to factors such as:

- Source-synchronous data transmit (data write function)
- Source-synchronous data receive (data read function)

While these functions are common in all high-speed interfaces, implementing the design in an FPGA is particularly challenging because of the following requirements:

- The implementation must be able to use any of the four sides (top, bottom, left, or right) of the FPGA device in the interface. (The left and right sides of the device cannot be mixed with the top and bottom sides of the device).
- The design should minimize logic resources by using look-up table (LUT)-based RAMs for data capture.
- The design should implement Delay Calibration Circuits without using global clock buffers (BUFGs) and digital clock managers (DCMs).

Controller

The Controller is responsible for transmitting and receiving all signals to and from the memories. These major functions include:

- Writing data to the memory
- Reading data from the memory
- Providing all the necessary control signals
- Transferring the read data clock domain from the memory domain to the FPGA domain

Implementing these functions is a major challenge in a memory interface design. The write data and strobe must be clocked out of the FPGA where the strobe is center-aligned with respect to data and is non-free-running. The FPGA must also generate the control signals needed to read from and write data to the memory.

Further details on generation of the control signals, write data, and the related timing margin calculations are explained in detail in application notes XAPP688 and XAPP678. Note that the data capture scheme outlined herein is different from the method used in application notes XAPP678 and XAPP688.

Data Read

For DDR SDRAMs, the read data is edge-aligned with a source-synchronous clock, and the clock is a non-free-running strobe. The design must be able to receive the data using the strobe and transfer it to the FPGA clock domain.

Read Data Capture

During a read transaction, the DDR SDRAM device sends the clock/strobe (DQS) and associated data to the FPGA. The DQS is edge-aligned with the data. Capturing the data is a challenging task in source-synchronous interfaces that run at higher frequencies. The data changes at every edge of the DQS and the strobe is not free-running in DDR. Read data from the memory device is captured directly into the FPGA fabric using a delayed DQS. The mechanism to delay the DQS is explained in the “Delay Circuits” section.

Instead of registering the data in the input or output blocks (IOBs), a LUT-based dual-port distributed RAM is used for data capture. The LUT RAM is configured as a pair of FIFOs, and each data bit is input into both FIFOs (Figure 2). These 16-entry deep FIFOs are asynchronous with independent read and write ports.

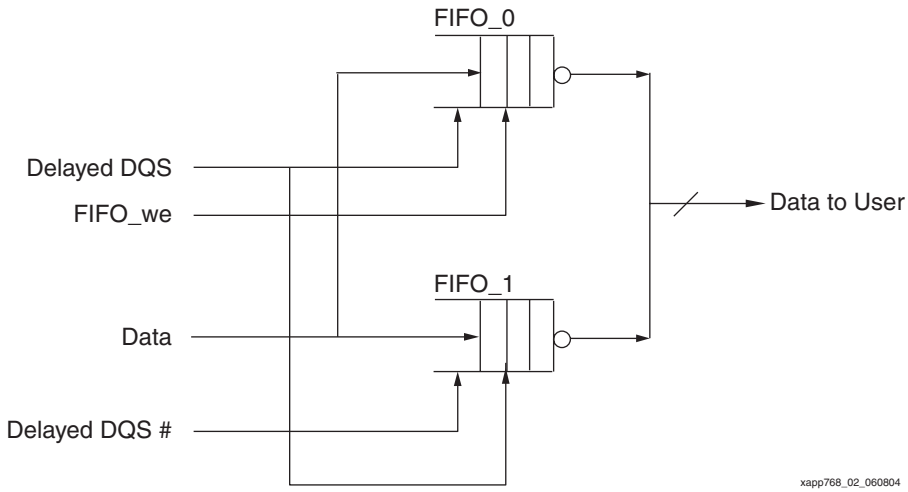

Figure 2: **FIFO Block Diagram**

Read Data Clocking

Read data from the memory is written into FIFO_0 on the rising edge of the delayed DQS, and into FIFO_1 on the falling edge of the delayed DQS. Data can be read out of both FIFO_0 and FIFO_1, simultaneously.

The FIFO write pointer is generated using the delayed DQS. FIFO read pointers are generated in the FPGA internal clock domain. The FIFOs are written when the FIFO write enable signal (FIFO_we) is asserted. Generation of the FIFO_we signal is explained in "Write Enable Generation".

The rst_dqs_div signal is asserted any time during the preamble for the DQS. This signal is deasserted any time during the last two DQS phases. Figure 3 illustrates the concept behind rst_dqs_div.

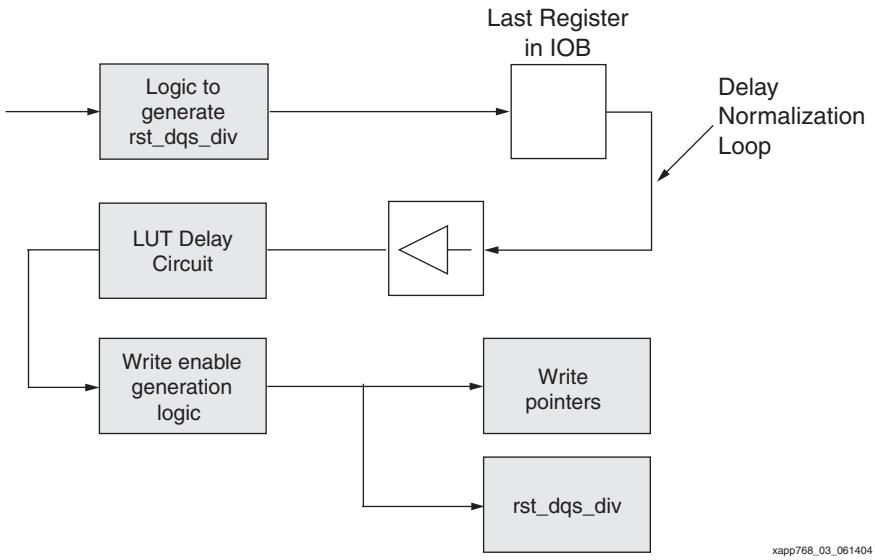

Figure 3: **Rst_dqs_div Circuit**

The rst_dqs_div signal is driven to an IOB as an output and is then taken as an input through the input buffer. This technique normalizes the IOB and trace delays between the rst_dqs_div and DQS Clock signals. The rst_dqs_div signal from the input PAD of the FPGA uses identical routing resources as the DQS before it enters the LUT delay circuit. The trace delay of the loop should be the sum of the trace delays of the clock forwarded to the memory and the DQS.

The DQS LUT delay circuit, shown in Figure 3, is identical to the LUT delay circuit used for the delayed DQS. This ensures that rst_dqs_div and the delayed DQS signals take identical paths, and have similar delays before being sent to the FIFO write pointers and FIFO write enable inputs.

Write Enable Generation

Figure 4 shows the logic for write enable generation. The FIFO_0 write enable signal is the logical OR of the rst_dqs_div and the registered version of the rst_dqs_div output. FIFO_1 is enabled after the first positive transition of the delayed DQS signal. This logic eliminates false latching of data into the FIFOs and false incrementing of the write pointers during the preamble period of the delayed DQS three-state to low transition.

The registered output enables the FIFOs and FIFO write pointers during the post-amble period, where the rst_dqs_div signal is deasserted. The registered rst_dqs_div flag is cleared on the last trailing edge of the delayed DQS, and the FIFOs and FIFO write pointers are disabled.

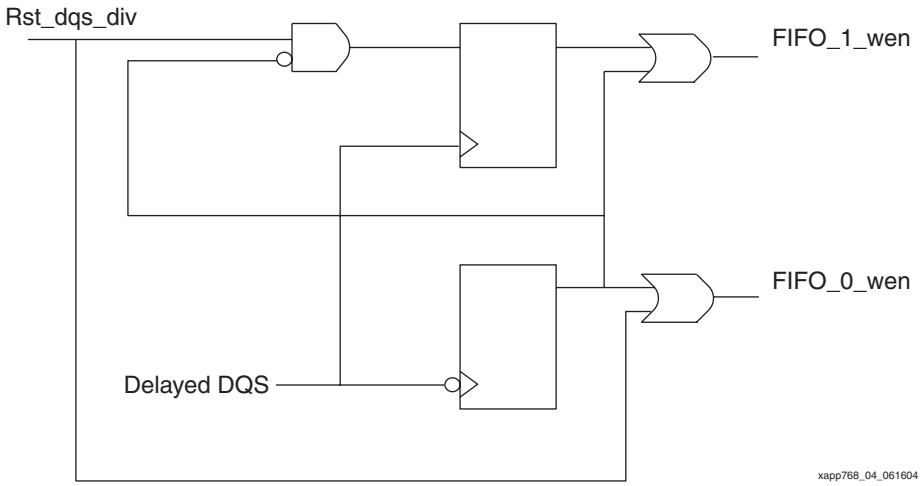

Figure 4: **FIFO Write Enable Generation**

Figure 5 shows the timing diagram for the rst_dqs_div signal and for the FIFO write enables and write pointer enables. It shows that there is enough margin between the write enable input and clock inputs of the write pointers and FIFOs.

Data is latched into the FIFOs when the delayed DQS toggles. The FIFO_0 and FIFO_1 write pointers are enabled when the rst_dqs_div is asserted. Data is latched into FIFO_0 on the rising edge of the delayed DQS, and the FIFO_0 write pointer increments on the same edge. Data is latched into FIFO_1 on the falling edge of the delayed DQS, and the FIFO_1 write pointer increments on this same edge.

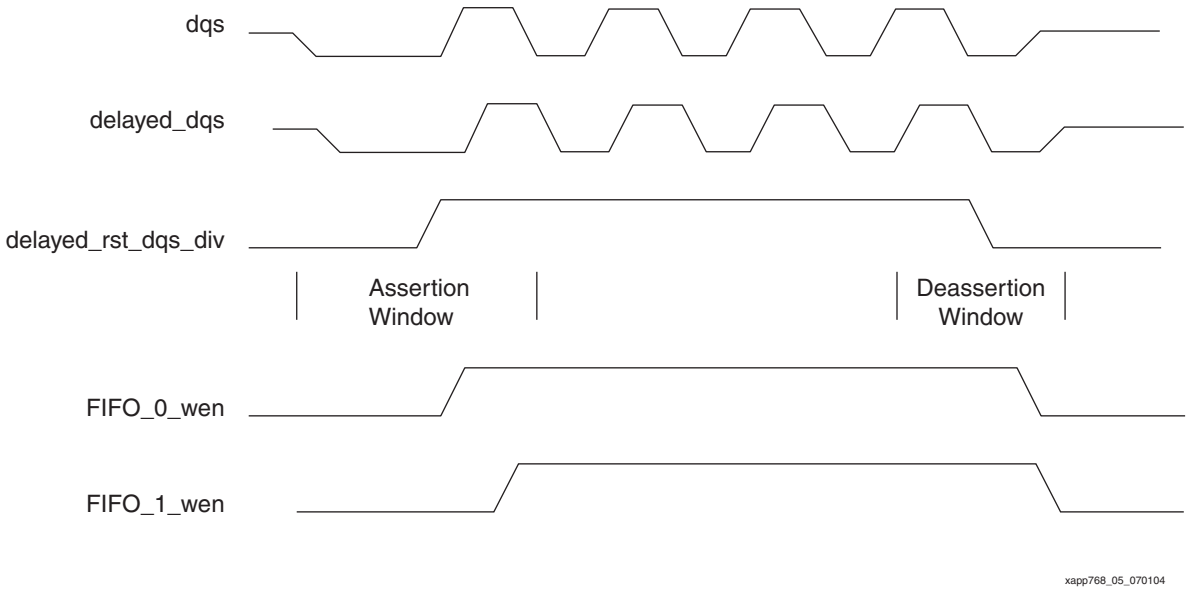

Figure 5: **FIFO Write Enable Timing Diagram**

Figure 6 is a timing diagram showing the data capture. Data from memory is clocked on the rising edge into FIFO_0 and on the falling edge into FIFO_1.

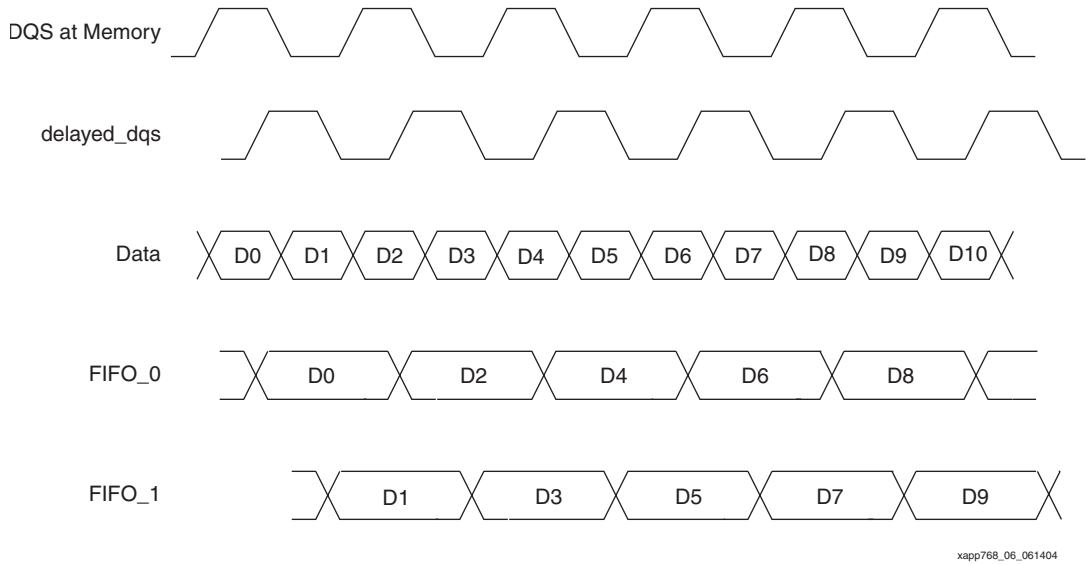

Figure 6: **Read Data Timing Diagram**

Delay Circuits

The total strobe delay must fall within a data valid window; therefore, the strobe is delayed using internal delay elements. The delay elements consist of look-up tables (LUTs) and other resources in the Xilinx fabric. Selecting the elements and routing resources used within those elements define the delay values precisely. The strobe delay circuit used here is similar to the circuit used in application notes XAPP678 and XAPP688.

Figure 7 illustrates the way LUTs are configured to create delay elements. The nominal delay for each LUT is 560 picoseconds (including the route delay) in a Spartan–3 (–5 speed grade) device.

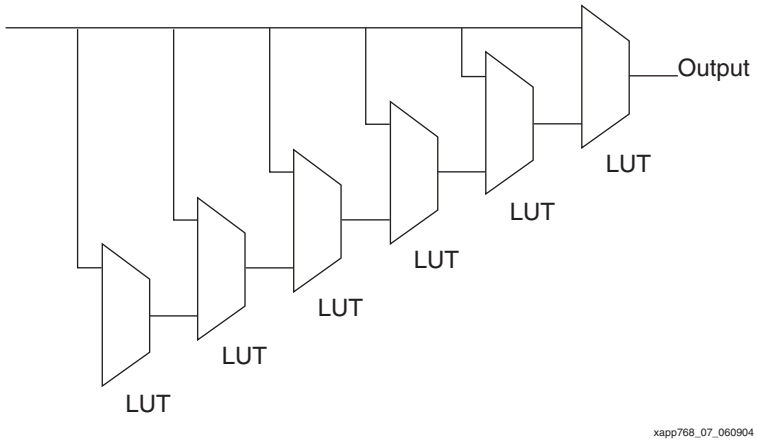

Figure 7: **Building Delay Elements Using LUTs in an Xilinx FPGA**

Local Clocking Resources

The delayed strobe in this design uses the local clocking resources available in the device for the clock routing. Full hex lines that have low skew are located throughout the device. Further details on local clock resources are explained in detail in XAPP769 "Local Clocking Resources in Spartan-3 FPGAs".

The left and right implementations use Vertical Full Hex (VFULLHEX) lines for local clock routing. The top and bottom implementations use VLONG, VFULLHEX, and HFULLHEX lines for local clock routing. This route is more complex than the left and right sides. The delay and skew of this local clock route is higher than the left and right local clock routes.

Timing Analysis

All timing analysis is done using a -5 speed grade 3S1500FG456 device for 166 MHz and using a -4 speed grade 3S1500FG456 device for 133 MHz.

Left and Right Banks

Table 1 illustrates the timing analysis for the FIFO-based data capture scheme already described. On the left and right banks, the local clock net delay is relatively small. A single Vertical Full Hex (VFULLHEX) line is used for local clock distribution. Because of this, it is possible to easily control the overall strobe delay to fall within the calculated window. As previously noted, the overall strobe delay is controlled with the delay circuit. The performance on a left and right side design is 166 MHz or 333 Mb/s on a -5 speed grade device and 133 MHz or 266 Mb/s on a -4 speed grade device.

Table 1: **Timing Analysis for Left and Right Implementation**

Parameter	Value	Leading Edge Uncertainties	Trailing Edge Uncertainties	Explanation	Comments
Tclock	6000			Clock period	Clock period at 133 MHz
Tclock_phase	3000			Clock phase	Half-clock period
Tclock_duty_cycle_dist	300	0	0	Duty-cycle distortion of clock to memory	Clock duty cycle distortion is 5% of clock period
Tdata_period	2700			Total data period; Tclock_phase-Tdcd	
Tdqsq	450	450	0	Strobe to data distortion from memory data sheet. (TSOP package).	
Tpackage_skew	90	90	90	Worst case package skew	
Tds	452	452	0	Set up time from Spartan-3 –5 data sheet	
Tdh	–35	0	–35	Hold time from Spartan-3 –5 data sheet	
Tjitter	100	0	0	Data and strobe jitter together since they are generated from the same clock	
Tlocal_clock_time	20	20	20	Worst case local clock line skew	Skew on local clock routes
Tpcb_layout_skew	50	50	50	Skew between data lines and strobes on the board	Skew between data lines and data strobe from memory output to FPGA input
Tqhs	550	0	550	Hold skew factor for DQ; from memory data sheet	Hold skew for data bits from memory data sheet
Total uncertainties		1062	675	Worst case for leading and trailing can never happen simultaneously	
Window for DQS position for normal case	963	1062	2025	Worst case window of 963 ps	

Top and Bottom Banks

On the top and bottom banks, the local clock net has a significantly higher delay. The TRCE static timing tool reports delays in the range of 1.7 to 1.8 ns for different parts. The performance on a top and bottom implementation is limited to 133 MHz on -4 and -5 speed grades. A timing analysis at 133 MHz for the top and bottom banks is shown in Table 2.

Table 2: **Timing Analysis for Top and Bottom Implementation**

Parameter	Value	Leading Edge Uncertainties	Trailing Edge Uncertainties	Explanation	Comments
Tclock	7500			Clock period	Clock period at 133 MHz
Tclock_phase	3750			Clock phase	Half-clock period
Tclock_duty_cycle_dist	375	0	0	Duty-cycle distortion of clock to memory	Clock duty cycle distortion is 5% of clock period
Tdata_period	3375			Total data period; Tclock_phase – Tdcd	
Tdqsq	450	450	0	Strobe to data distortion from memory data sheet. (TSOP package)	
Tpackage_skew	90	90	90	Worst case package skew	
Tds	519	519	0	Set up time from Spartan-3 –5 data sheet	
Tdh	–40	0	–40	Hold time from Spartan-3 –5 data sheet	
Tjitter	100	0	0	Data and strobe jitter together since they are generated from the same clock	
Tlocal_clock_time	25	25	25	Worst case local clock line skew	Skew on local clock routes
Tqhs	550	0	550	Hold skew factor for DQ; from memory data sheet	Hold skew for data bits from memory data sheets
Tpcb_layout_skew	50	50	50	Skew between data lines and strobes on the board	Skew between data lines and data strobe from memory output to FPGA input
Total uncertainties		1134	675	Worst case for leading and trailing can never happen simultaneously	
Window for DQS position for normal case	1566	1134	2585	Worst case window of 963 ps	

Delay Calibration Circuit

The LUT delays in the delay circuit are measured using the tap delay circuit shown in Figure 8. Each inverter in the tap delay circuit is created using a LUT. Similarly, each multiplexer in Figure 7 is built using a LUT. Compared to the delay calibration circuit used in XAPP678 and XAPP688, this calibration circuit does not require additional DCMs or global buffers (BUFGs).

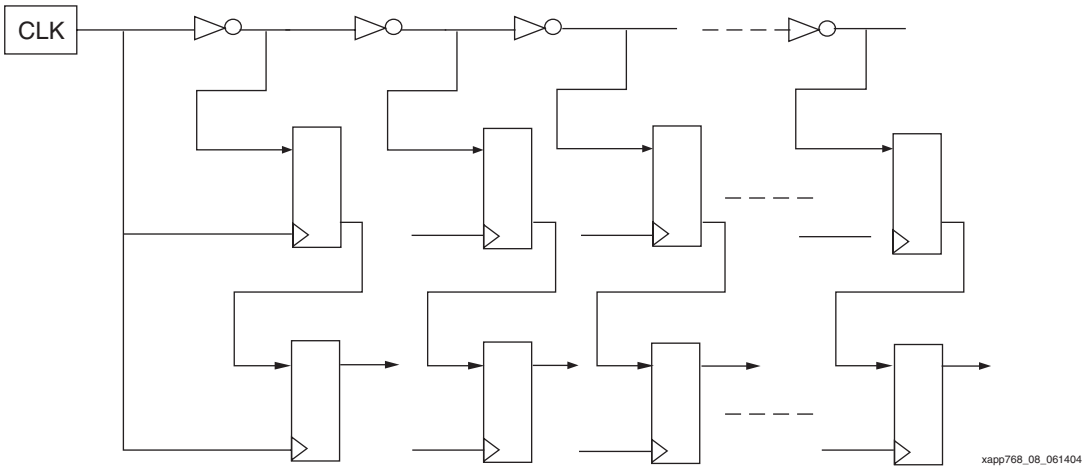

Figure 8: **Tap Delay Circuit Built Using LUTs**

Normally, the 166 MHz clock phase (6 ns period, 3 ns phase) traverses through five to six taps of the tap delay circuit. The number of taps required for a clock phase is determined by the specific FPGA used. If a faster FPGA is chosen, the number of LUT delays in the strobe path is increased by the delay calibration logic to ensure that the strobe is still within the valid data window. If the Spartan–3 FPGA gets faster due to process variations or any other reason, the number of tap delays for the same clock phase is automatically increased by the delay calibration logic.

LUT Selected Banks

Table 3 shows the LUT selection table when top and bottom banks are used for data and strobe pins. At normal temperature, one multiplexer of the delay circuit is selected by the calibration circuit to delay the strobe.

Table 3: **LUT Selection for Top and Bottom Implementation**

Frequency: 133 MHz Clock Period: 7500 ns	Nominal	90%	80%	70%	60%	50%	40%
Data delay	632	568.8	506	442	379	316	253
LUT delay	625	563	500	438	375	313	250
Number of LUTs in a clock phase	6	7	8	9	10	12	15
Number of multiplexers	1	1	1	2	3	4	5
Total DQS delay	2971	2674	2377	2518	2533	2425	2188
Total extra DQS delay	2339	2106	1871	2076	2153	2109	1936
Data valid window; lower-bound	1249	1169	1089	1009	929	850	770
Data valid window; upper-bound	2585	2609	2633	2657	2681	2705	2729

1. In this table, the clock frequency is 133 MHz.
2. The total DQS delay in the table is calculated based upon the Spartan–3 V1.31 speed files. DQS delay is the total delay for the route taken by the DQS from the input pad to the delay circuit, plus the delay circuit, and the local clock line. The total extra DQS delay is the total extra DQS delay minus the data delay. This value should always fall within the data valid window.

Table 4 shows the LUT selection table when the left and right banks are used for data and strobe pins. At nominal temperature, three multiplexers of delay circuit are selected by the calibration circuit to delay the strobe.

The number of multiplexers in the DQS path is dependent only on the number of LUTs in a clock phase. This mapping is constant across frequencies of up to 166 MHz. Hence, the entire design is frequency-independent. Additional analysis is available upon request.

Table 4: **LUT Selection for a Left and Right Implementation**

Frequency: 166 MHz Clock Period: 6000 ns	Nominal	90%	80%	70%	60%	50%	40%
Data delay	480	432	384	336	288	240	192
LUT delay	560	504	448	392	336	280	224
Number of LUTs in a clock phase	5	6	7	8	9	11	13
Number of multiplexers	2	3	3	3	4	4	4
Total DQS delay	1995	2300	2044	1789	1869	1558	1246
Total extra DQS delay	1515	1868	1660	1453	1581	1318	1054
Data valid window; lower-bound	1062	1001	940	878	817	756	695
Data valid window; upper-bound	2025	2038	2050	2063	2075	2088	2100

1. In this table, the clock frequency is 166 MHz. The parameters are from a -5 speed grade device.
2. The total DQS delay in the table is calculated based upon the Spartan–3 V1.31 speed files. DQS delay is the total delay for the route taken by the DQS from the input pad to the delay circuit, plus the delay circuit, and the local clock line. The total extra DQS delay is the total extra DQS delay minus the data delay. This value should always fall within the data valid window.

MIG 007 Tool

The interface described in this application note is quite intricate to implement. Several instances require precise routing delays and placement. To facilitate designs using the methods described here, the Xilinx Memory Interface Generator (MIG 007) tool has been developed. The MIG 007 DDR Compiler is a tool that generates a DDR interface for Xilinx Spartan-3, Virtex-II, and Virtex-II Pro FPGAs. The tool requires the following inputs: FPGA device, frequency, data width, banks to use, and so forth. Inputs are provided by users through an intuitive graphical user interface (GUI). The tool generates RTL, SDC, EDIF, UCF, and document files. For more information on the MIG 007 tool, refer to the *Xilinx Memory Interface Generator (MIG 007) User Guide* (UG067).

The MIG 007 tool is invoked from either a command line prompt or through Microsoft Windows (see Figure 9). This version is supported on Windows platforms for a number of Spartan-3 devices.

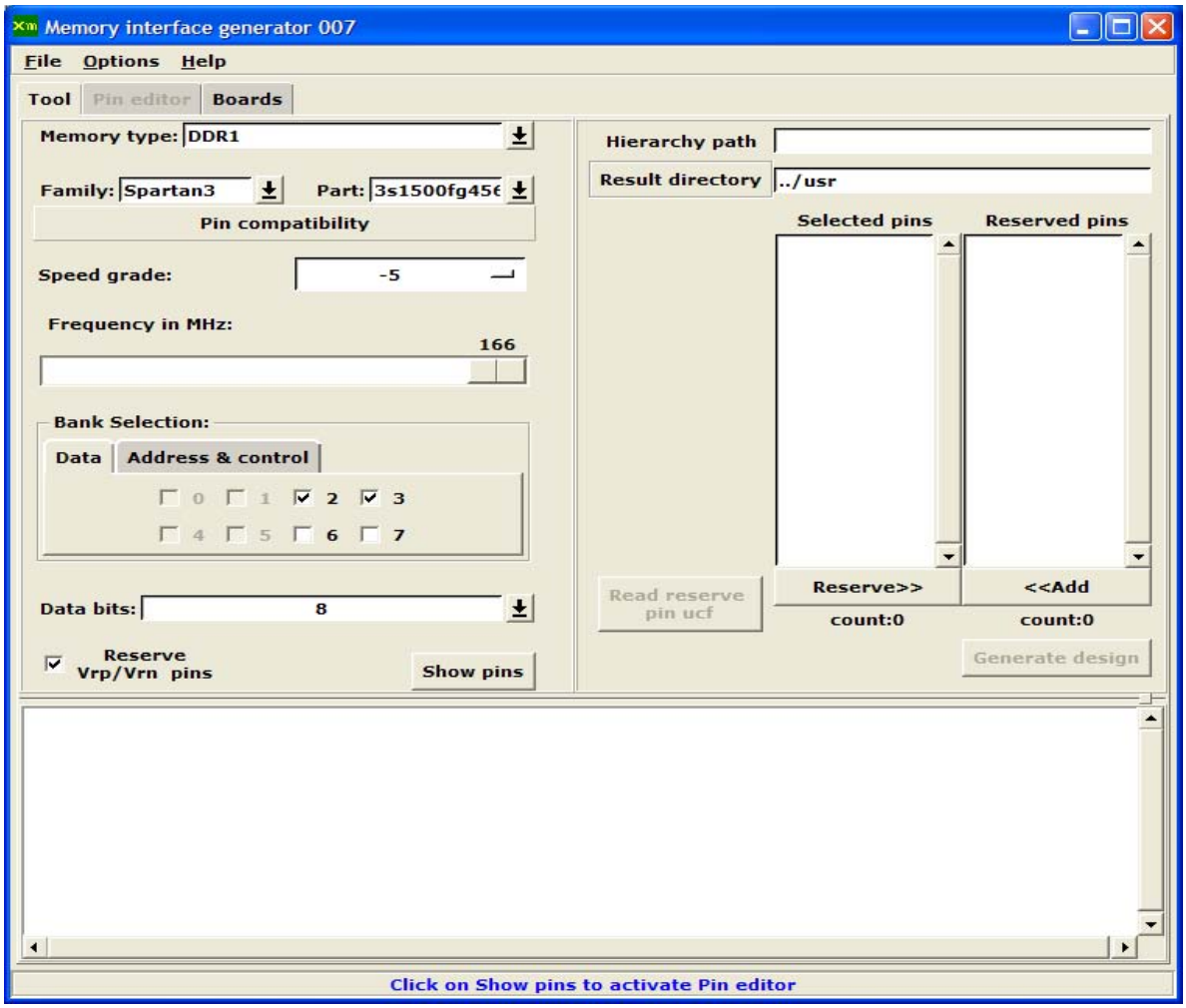

Figure 9: **MIG 007 Tool**

Table 5 shows the maximum data width possible on each side for left and right implementations.

Table 5: **Maximum Data Width per Side for Right and Left Implementations**

Device (1)	FG320	FG456	FG676	FG900	FG1156	FT256
3S400	32	32	N/A	N/A	N/A	24
3S1000	24	48	56	N/A	N/A	16
3S1500	24	48	72	N/A	N/A	N/A
3S2000	N/A	N/A	72	80	N/A	N/A
3S4000	N/A	N/A	N/A	88	104	N/A
3S5000	N/A	N/A	N/A	88	120	N/A

Notes:

1. All device packages are not shown.

Table 6 shows the maximum width possible for top and bottom implementations.

Table 6: **Maximum Data Width per Side for Top and Bottom Implementations**

Device (1)	FG320	FG456	FG676	FG900	FG1156	FT256
3S400	16	32	NA	NA	NA	16
3S1000	16	32	48	NA	NA	16
3S1500	16	16	64	NA	NA	NA
3S2000	NA	NA	48	80	NA	NA
3S4000	NA	NA	NA	96	96	NA
3S5000	NA	NA	NA	88	96	NA

1. All device packages are not shown.

References

Xilinx Documentation:

- XAPP253: "Synthesizable 400 Mb/s DDR SDRAM Controller"
- XAPP678c: "Data Capture Technique Using CLB Flip-Flops"
- XAPP688: "Creating High-Speed Memory Interfaces with Virtex-II and Virtex-II Pro FPGAs"
- XAPP759c: "Interfacing Virtex-II Devices With DDR Memories for Performance to 167 MHz"
- XAPP769: "Local Clocking resources in Spartan–3 FPGAs"
- Spartan–3 Data Sheet
- Virtex–II Data Sheet
- Virtex–II Pro Data Sheet
- UG067: Xilinx Memory Interface Generator (MIG 007) User Guide

Other Documentation:

- Micron Data Sheet MT46v32M16TG-6T

Conclusion

The innovative techniques described in this application note make it easy to design a DDR SDRAM memory interface with Spartan–3 FPGAs. This design has been simulated, synthesized (with Synplicity), and taken through the Xilinx Project Navigator flow.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/01/04	1.0	Initial Xilinx release.
10/14/04	2.0	Updated for MIG 007.