XAPP758C (v1.2) January 10 2005

Interfacing Virtex-II Devices With DDR SDRAM Memories for Performance to 167 MHz

Author: Nagesh Gupta

Summary

This application note describes how to use any side of a Xilinx Virtex™-II or a Virtex-II Pro™ device to interface to a Double Data Rate (DDR) SDRAM device with reduced resources. All examples in this application note assume a DDR SDRAM interface on an XC2VP20FF1152-6 Virtex-II Pro FPGA. The design concepts are applicable to other types of memories. The interface speed is up to 167 MHz.

Introduction

High-speed memory interfaces are typically source-synchronous with double-data rate. In a source-synchronous design, the generated clocks are transmitted along with the data. Typically the data is received using the received clock and then transferred into the receiver's clock domain.

Figure 1 shows the memory interface and controller as part of a system. The memory interface has a 72-bit data width. The data bus can use any of the four sides of the FPGA. The data width can be reduced if sufficient pins are not available in the FPGA on any particular side. Address and control signals can use either the same side as the data bus or an adjacent bank.

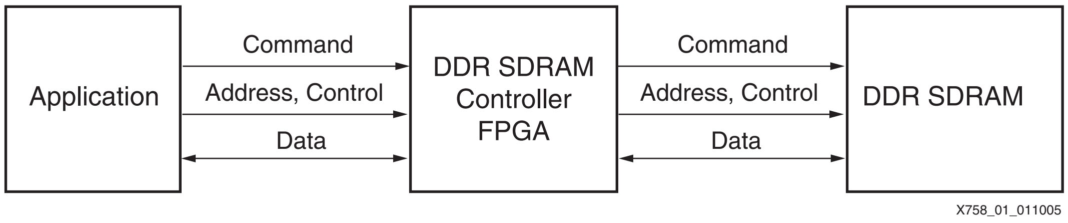


Figure 1: **System Diagram**

Data is double-pumped to DDR SDRAM memory on both the positive and the negative edges of the clock. All address and control signals are compatible with the SSTL_II standard, and data is SSTL_II Class II compatible.

Key Challenges

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

- Source-synchronous data transmission (data write function)
- Source-synchronous data reception (data read function)

While these functions are common in all high-speed interfaces, the following requirements make this design particularly challenging:

- Ability to use any of the four sides (top, bottom, left, and right) of the FPGA device in the interface
- Usage of fewer resources like LUT-based RAMs for data capture
- Designing of a delay calibration circuit without any BUFGs or DCMs

Controller

The controller transmits and receives all signals to and from the memories. Its major functions include:

- Writing data into 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

This functionality constitutes a major challenge in memory interface design. The FPGA clocks out the write data and strobe, where the strobe is center-aligned with respect to the data and is non-free-running. The FPGA also generates the necessary control signals for reads and writes of the memory. Further details on the generation of control signals, write data, and the related timing margin calculations are explained in XAPP678C and XAPP688. The data capture scheme outlined in this application note is different from the method used in XAPP678C and XAPP688.

Data Reads

Memory read data is edge-aligned with a source-synchronous clock. For DDR SDRAMs, 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.

Data Capture

During a read transaction, the memory device forwards the clock/strobe (DQS) and associated data to the FPGA. The DQS signal is edge-aligned with the data in the DDR SDRAMs. Data capture is a challenging task in source-synchronous interfaces at higher frequencies because the data changes on every edge of DQS and the strobe is not free-running in DDR SDRAMs.

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 Circuit" section. The data is not registered in the input/output blocks (IOBs). LUT-based dual-port distributed RAM is used for data capture. The LUT RAM is configured as a FIFO. Each data bit is input into two FIFOs. The FIFOs are asynchronous with independent read and write ports. Each FIFO is 16 entries deep. Figure 2 shows a block diagram of the FIFOs. Read data from 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.

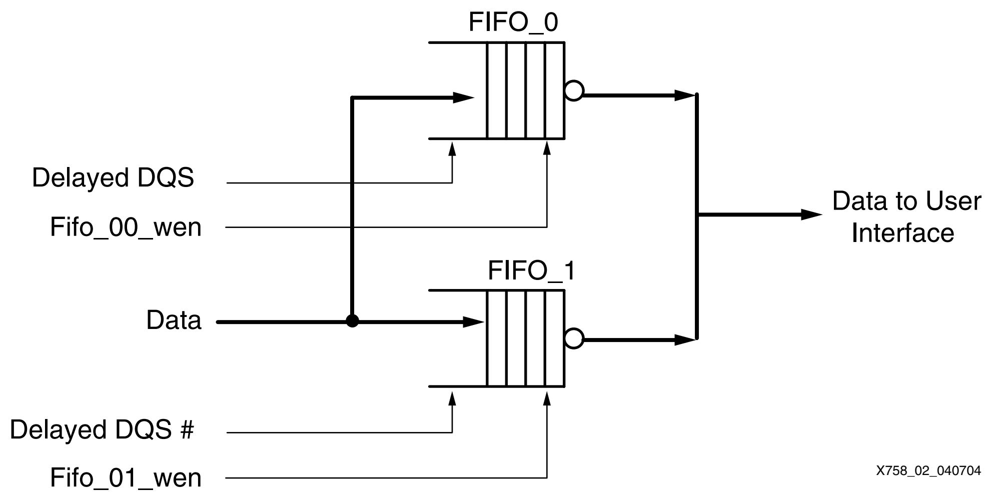


Figure 2: **FIFO Block Diagram**

A write pointer is generated using the delayed DQS. Read pointers are generated in the FPGA internal clock domain. The FIFOs can be written when the FIFO write enable signal (FIFO_we) is asserted. The FIFO_we signal generation is explained later. The rst_dqs_div signal is asserted any time during preamble for DQS. This signal is deasserted any time during the last two DQS phases. Figure 3 illustrates the concept behind rst_dqs_div.

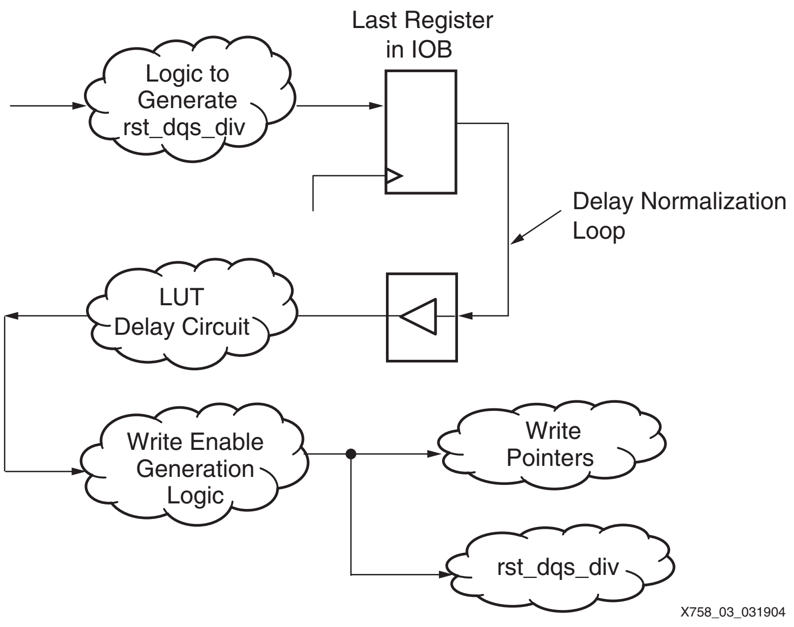


Figure 3: **The rst_dqs_div Circuit**

The DDR 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 rst_dqs_div_out is output from the FPGA, loops back on the PCB, and is input as rst_dqs_div_in. The strobes (DQS), data (DQ) and clock (CK/CK#) signals should be matched in trace length from the FPGA to the memory device. The total trace length of the loop back signal from the rst_dqs_div output pad to the rst_dqs_div input pad should be equal to twice the length of any one of the above set of signals (DQ/DQS/CK). This loop back signal is used to generate write enable signals to the read data capture FIFOs. The rst_dqs_div circuit with the external normalization loop back is shown in Figure 3.

Figure 4 illustrates the logic for write enable generation. The FIFO_0 write enable signal is the logical OR of 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 3-state to Low transition.

The registered output enables FIFOs and FIFO write pointers during the postamble period, where rst_dqs_div is deasserted. The registered rst_dqs_div flag is cleared with the last trailing edge of the delayed DQS, hence the FIFOs and FIFO write pointers are disabled.

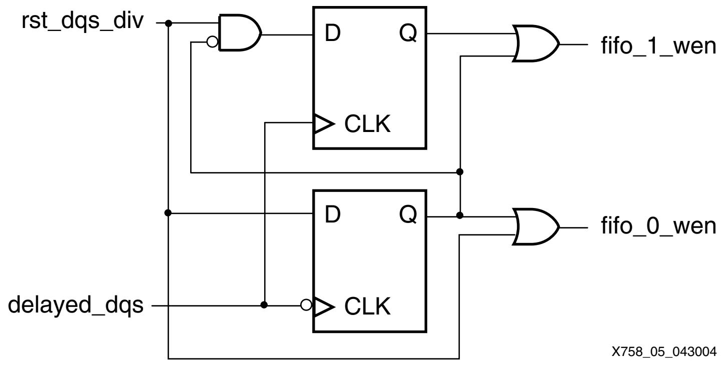


Figure 4: **FIFO Write Enable Generation**

Figure 5 is a timing diagram of rst_dqs_div 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.

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

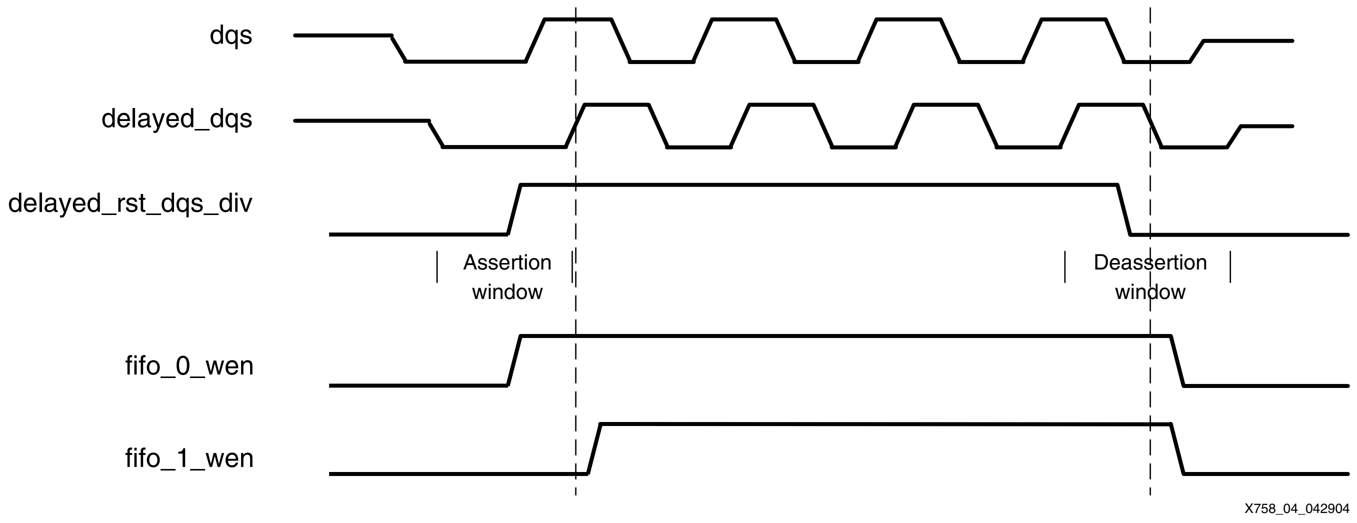


Figure 5: **FIFO Write Enable Timing Diagram**

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

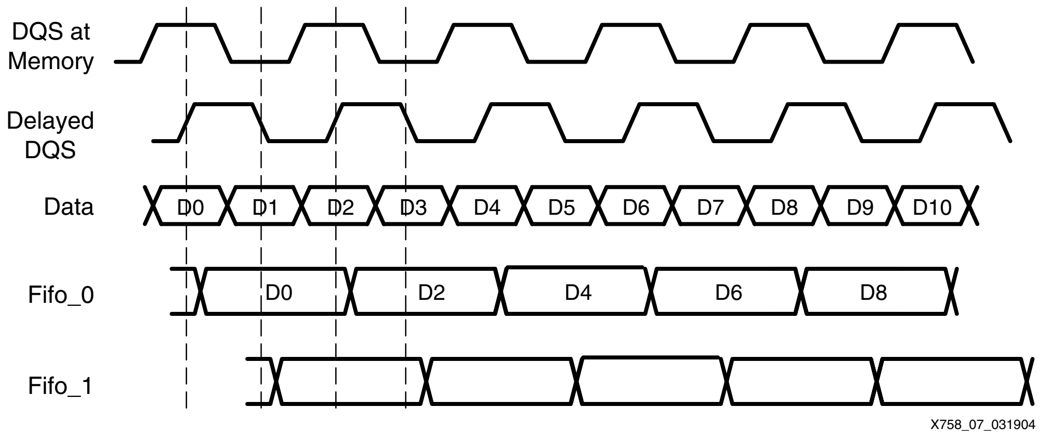


Figure 6: **Read Data Timing**

Delay Circuit

The strobe is delayed using internal delay elements, where the total strobe delay must fall within the valid data window. The delay elements consist of LUTs and other resources in the Xilinx FPGA fabric. Selecting the elements and all routing resources used within the elements defines the delay values precisely. The strobe delay circuit used here is similar to the circuit used in XAPP678C and XAPP688.

Figure 7 shows how the LUTs are configured to create delay elements. The nominal delay of each LUT is 300 ps in a 2VP20, –6 device.

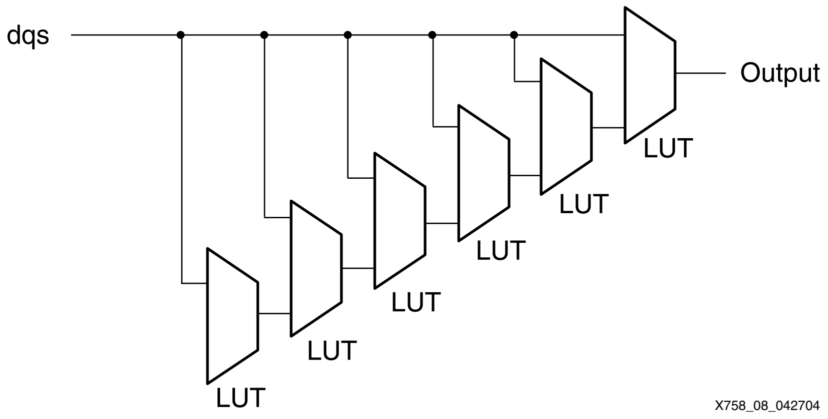


Figure 7: **Building Delay Elements Using LUTs in a Xilinx FPGA**

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 with low skew are located throughout the device. Further details on local clock resources are explained in detail in XAPP609.

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

Timing Analysis

All timing analysis is done using a 2VP20FF1152 device with a –6 speed grade.

Left and Right Banks

Table 1 lists the timing analysis for the left and right banks of the FIFO-based data capture scheme described previously. On the left and right banks, the local clock net delay is relatively small – trace reports delay values between 250 and 400 ps for different parts. A single VFULLHEX line is used for local clock distribution. Hence, the overall strobe delay can be easily controlled to fall within the calculated window with the delay circuit.

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

Parameter	Value (ps)	Leading Edge Uncertainties (ps)	Trailing Edge Uncertainties (ps)	Meaning
Tclock	5988			Clock period at 167 MHz
Tclock_phase	2994			Clock phase, half clock period
Tclock_duty_cycle_dist	299.4	0	0	Duty cycle distortion of clock to memory, 5% of clock period
Tdata_period	2695			Total data period, Tclock_phase – Tdcd
Tdqsq	500	500	0	Strobe-to-data distortion from the memory data sheet
Tpackage_skew	65	65	65	Skew due to package pins and FPGA internal routes
Tds	410	410	0	Setup time from the Virtex-II Pro data sheet
Tdh	-7	0	-7	Hold time
Tjitter	100	0	0	Combined Data and Strobe jitter (they are generated off the same clock)
Tlocal_clock_line	50	50	50	Skew on local clock routes
Tpcb_layout_skew	50	50	50	Skew between data lines and data strobes from memory output to FPGA input
Tqhs	450	0	450	Hold skew factor for data bits from the memory data sheet
Total uncertainties		1075	608	Worst case for leading and trailing edges can never happen simultaneously.
Window for DQS position for nominal case	1012	1075	2087	

Top and Bottom Banks

On the top and bottom banks, the local clock net has a significantly higher delay – trace reports delays from 1.2 ns to 1.4 ns for different parts. The minimum extra delay on the strobe compared to data is approximately 1.6 ns to 1.7 ns because the delay circuit adds an additional 600 ps to the 1.2 ns to 1.4 ns local clock net delay. A timing analysis at 167 MHz for top and bottom banks is shown in Table 2.

Table 2: **Timing Analysis for Top/Bottom Implementations**

Parameter	Value (ps)	Leading Edge Uncertainties (ps)	Trailing Edge Uncertainties (ps)	Meaning
Tclock	5988			Clock period at 167 MHz
Tclock_phase	2994			Clock phase, half clock period
Tclock_duty_cycle_dist	299.4	0	0	Duty cycle distortion of clock to memory, clock duty cycle distortion is 5% of clock period
Tdata_period	2695			Total data period, Tclock_phase – Tdcd
Tdqsq	500	500	0	Strobe-to-data distortion from memory data sheet
Tpackage_skew	65	65	65	Skew due to package pins and board layout (can be reduced with tighter layout)
Tds	410	410	0	Setup time from Virtex-II Pro data sheet
Tdh	-7	0	-7	Hold time
Tjitter	100	0	0	Combined Data and Strobe jitter (they are generated off the same clock)
Tlocal_clock_line	110	110	110	Skew on local clock routes
Tpcb_layout_skew	50	50	50	Skew between data lines and strobes from memory output to FPGA input
Tqhs	450	0	450	Hold skew for data bits from memory data sheets
Total uncertainties		1135	668	Worst case for leading and trailing edges can never happen simultaneously.
Window for DQS position for nominal case	892	1135	2027	Worst-case window

Delay Calibration Circuit

The LUT delays in the delay circuit are measured using a tap delay circuit as shown in Figure 8. Each inverter of the tap delay circuit is created using a LUT. Similarly, each mux in Figure 7 is built using a LUT. Compared to the delay calibration circuit used in XAPP678C and XAPP688, this calibration circuit does not require an additional DCM BUFG.

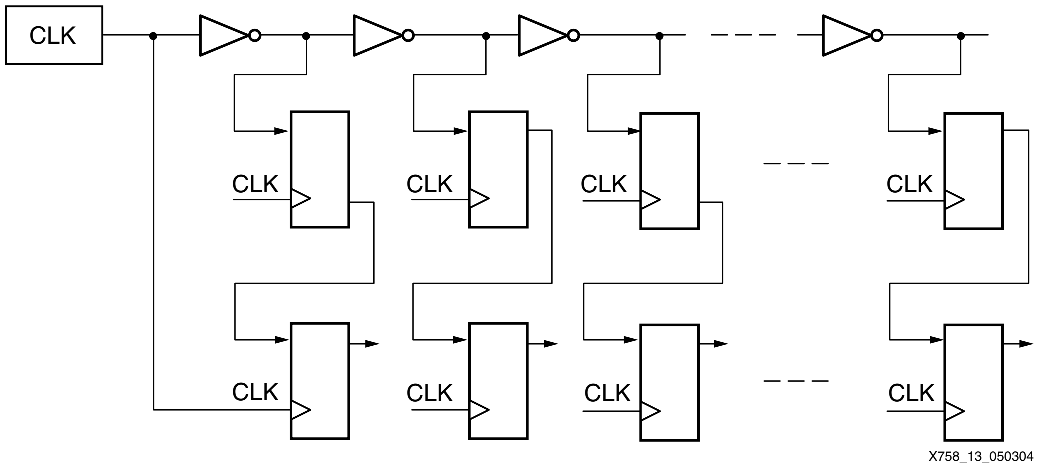


Figure 8: **Tap Delay Circuit Built using LUTs**

Normally, a 167 MHz clock phase (~6 ns period, ~3 ns phase) traverses through 9 to 11 taps of the tap delay circuit. If the chip gets faster due to process variations or other factors, the number of tap delays for the same clock phase increases.

The number of taps required for a clock phase determines if the chip is faster or slower. This is the concept behind the calibration circuit. As the chip operates faster, the number of LUT delays in the strobe path is increased to ensure that the strobe remains within the data valid window.

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

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

Element	Nominal	90%	80%	70%	60%	50%	40%	Units
Data delay	450	405	360	315	270	225	180	ps
LUT delay	300	270	240	210	180	150	120	ps
Inter-LUT route in clock phase circuit	50	45	40	35	30	25	20	ps
Number of LUTs in a clock phase	9	10	11	12	14	17	21	ps
Number of Muxes	1	1	2	3	4	6	6	ps
Total DQS delay	2111	1899.9	1928.8	1897.7	1806.6	1805.5	1444.4	ps
Total extra DQS delay	1661	1494.9	1568.8	1582.7	1536.6	1580.5	1264.4	ps
Data valid window, lower bound	1135	1071.5	1008	944.5	881	817.5	754	ps
Data valid window, upper bound	2027	2048	2070	2092	2114	2136	2157	ps

Notes:

1. In this table, the clock frequency is 167 MHz, and the clock period is 5988 ps.
2. The total DQS delay in the table is calculated based on speed files. DQS delay is the total delay for the route taken by 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 DQS delay minus the data delay. This value should always fall within the data valid window.

Table 4 shows the LUT selection table when left and right banks are used for data and strobe pins. In this table, the clock frequency is 167 MHz, and the clock period is 5988 ps. At nominal temperature, four muxes of delay circuit are selected by the calibration circuit to delay the strobe.

The number of muxes in the DQS path is dependent only on the number of LUTs in a clock phase. This mapping is constant across frequencies (from 100 to 150 MHz). Hence the entire design is frequency-independent. Additional analysis is available upon request.

Table 4: **LUT Selection Table for Left or Right Implementation**

Element	Nominal	90%	80%	70%	60%	50%	40%	Units
Data delay	450	405	360	315	270	225	180	ps
LUT delay	300	270	240	210	180	150	120	ps
Inter-LUT route in clock phase circuit	50	45	40	35	30	25	20	ps
Number of LUTs in a clock phase	9	10	11	12	14	17	21	ps
Number of Muxes	4	4	4	5	6	6	6	ps
Total DQS delay	2050	1845	1640	1645	1590	1325	1060	ps
Total extra DQS delay	1600	1440	1280	1330	1320	1100	880	ps
Data valid window, lower bound	1075	1017.5	960	902.5	845	787.5	730	ps
Data valid window, upper bound	2087	2102	2118	2134	2150	2166	2181	ps

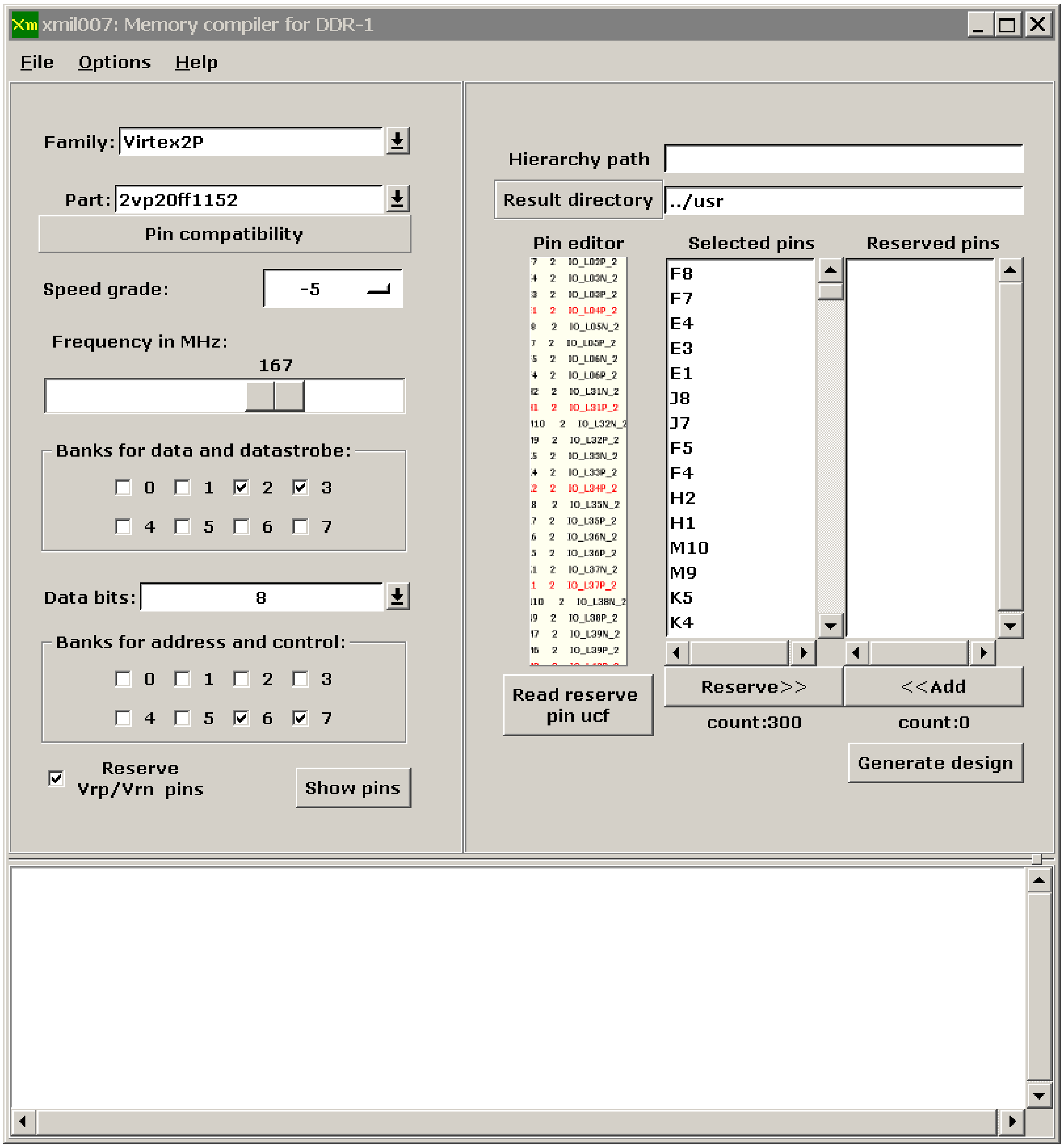
Notes:

1. In this table, the clock frequency is 167 MHz, and the clock period is 5988 ps.
2. The total DQS delay in the table is calculated based on speed files. DQS delay is the total delay for the route taken by 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 DQS delay minus the data delay. This value should always fall within the data valid window.

XMIL007 Tool

The interface described in this application note is quite intricate to implement. Several instances require precise routing delays and placement. To enable customers to effectively design using the methods described here, the *Xilinx Memory Interface Logic* (XMIL007) tool is provided (see Figure 9) to generate the entire interface including RTL and pin constraints. This tool works for most Virtex-II and Virtex-II Pro devices and generates the entire design.

The XMIL007 tool can be invoked from either the command line prompt or through Microsoft Windows. This version is supported only on Windows platforms for a number of Virtex-II and Virtex-II Pro devices.



Family: Virtex2P

Part: 2vp20ff1152

Speed grade: -5

Frequency in MHz: 167

Banks for data and datastrobe: [ ] 0 [ ] 1 [✓ 2] [✓ 3] [ ] 4 [ ] 5 [ ] 6 [ ] 7

Data bits: 8

Banks for address and control: [ ] 0 [ ] 1 [ ] 2 [ ] 3 [ ] 4 [ ] 5 [✓ 6] [✓ 7]

[✓] Reserve Vrp/Vrn pins

Hierarchy path:

Result directory: ../usr

Pin editor

```
7    2   IO_L03P_2
4    2   IO_L03N_2
3    2   IO_L03P_2
1    2   IO_L04P_2
8    2   IO_L05N_2
7    2   IO_L05P_2
5    2   IO_L06N_2
4    2   IO_L06P_2
l2   2   IO_L31N_2
l1   2   IO_L31P_2
110  2   IO_L32N_2
19   2   IO_L32P_2
5    2   IO_L33N_2
4    2   IO_L33P_2
2    2   IO_L34P_2
8    2   IO_L35N_2
7    2   IO_L35P_2
6    2   IO_L36N_2
5    2   IO_L36P_2
1    2   IO_L37N_2
1    2   IO_L37P_2
110  2   IO_L38N_2
19   2   IO_L38P_2
17   2   IO_L39N_2
16   2   IO_L39P_2
```

Selected pins

```
F8
F7
E4
E3
E1
J8
J7
F5
F4
H2
H1
M10
M9
K5
K4
```

count:300

Reserved pins

count:0

X758_09_082404

Figure 9: **Screen Shot of the XMIL007 Tool**

References

The following documents provide supplementary material for this application note:

- Xilinx Application Notes:
  - **XAPP678C**: "Data Capture Technique Using CLB Flip-Flops" (**available under NDA**)
  - XAPP688: "Creating High-Speed Memory Interfaces with Virtex-II and Virtex-II Pro FPGAs"
  - XAPP609: "Local Clocking Resources in Virtex-II Devices"
- Xilinx Virtex-II Pro Platform FPGAs Data Sheet (DS083)
- Micron Double Data Rate (DDR) SDRAM data sheet (MT46V2M32V1LG-5)

Conclusion

The innovative techniques described in this application note make it easy to design any memory interface with Virtex-II and Virtex-II Pro FPGAs.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/03/04	1.0	Initial Xilinx release.
08/24/04	1.1	Updated XMIL007 information and hyperlinks.
01/10/05	1.2	Revised paragraph under Figure 3.