Interfacing the QDR to the XILINX SPARTAN-II FPGA

QDR: An Introduction

The explosive growth of the internet is boosting the demand for high-speed data communications systems. This has led to the evolution of faster processors and is driving the speed of the interfacing peripherals higher. While the processors in these systems have evolved in performance, static memories have been unable to keep up the pace. Newer SRAM architectures have evolved to support the higher throughput requirements of current systems and processors. This application note introduces QDR, which is an SRAM architecture designed to improve the SRAM interface bandwidth by more than four times that of current solutions. This application note is also intended to describe the interface between this high-speed SRAM and the Xilinx Spartan-II FPGA.

QDR is a family of synchronous SRAMs with an innovative architecture. This was designed particularly for high performance networking systems by the QDR Consortium, which consists of Cypress, IDT, and Micron.

QDR stands for Quad Data Rate, which effectively means that four words of data can be transferred through the SRAM in a single clock cycle. The QDR Family includes two devices, which differ by the number of data words that will be burst on every access.

Table 1 shows the current and planned devices in the family.

Table 1. Devices in the QDR Family

Device	SIze	Description
CY7C1302	512K x 18	QDR - Burst of 2
CY7C1304	512K x 18	QDR - Burst of 4

What is QDR?

Most of the existing SRAM solutions are relics from the PC time, with interfaces designed to move data efficiently for PC-type single-I/O applications. In most networking applications continuous movement of data through the SRAM is a necessity. In such applications, there are continuous transitions between read and writes through the memory. Single-I/O devices like standard synchronous pipelined SRAMs do not perform well under these conditions. NoBL/ZBT family of SRAMs have optimized the synchronous SRAM architecture to allow no latency in the read/write transitions and have a 100% utilization of the I/O bus.

For most networking applications, the improvement in throughput provided by the NoBL/ZBT SRAMs is not enough. With the introduction of QDR SRAM devices, networking applications such as ATM switches and routers benefit from the simultaneous Read and Write capabilities that can be performed. With the absence of latency and the increase in data throughput provided in the QDR SRAM, simultaneous access to the same address location is guaranteed.

For more details on the QDR refer to the application note "QDR: The Next Generation High-Performance Networking SRAM."

CY7C1302

Figure 1 shows the block diagram of the CY7C1302 QDR device.

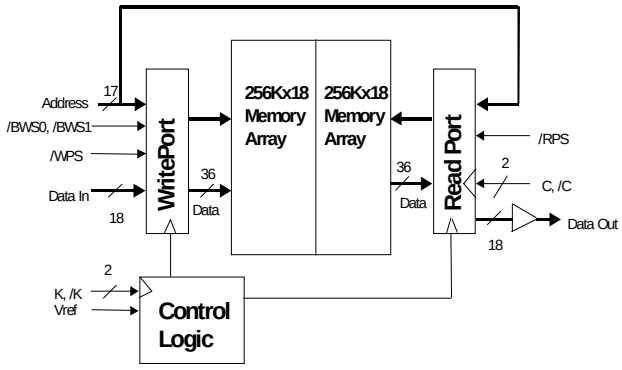

Figure 1. Block Diagram of the CY7C1302

The QDR SRAM family has two members: the two-word burst device (CY7C1302) and the four-word burst device (CY7C1304). The difference between the two is in terms of the number of words of data that can be obtained from the SRAM on a single read or provided to the SRAM on a write.

Figure 1 shows the block diagram for the CY7C1302. While the data ports are separated for the read and write ports, address lines are shared for the read and write ports. Data is transferred in a double data rate manner (DDR) on both input and output ports. This allows four words of data to be transferred on every clock cycle.

Description of Inputs to the QDR SRAM

Clock Inputs

QDR SRAMs have 4 clocks: K, /K, C, and /C. The K and /K clocks are used for sampling the inputs and C and /C clocks are used to clock out the data from the SRAM. All transactions are initiated on the rising edge of K.

Control Inputs

QDR SRAMs have a very simple control structure. Two control signals: /RPS and /WPS are used to control the read and write operations on the SRAM. These signals are sampled on the rising edge of the K clock. More information on these will be provided in later sections.

Address Inputs

The address inputs for the QDR is common for the read and write ports. On the CY7C1304 the addresses are samples on alternate cycles of the clock for a read or a write. On the

CY7C1302 the address inputs are sampled on the rising edge of K for the read and on the rising edge of /K for the write.

Data Inputs/Outputs

The data lines on the QDR are unidirectional. Two words can be transferred from and to the QDR in every cycle. More details on the functionality of the device are explained in the section on timings.

Timing Diagram

Figure 2 shows the timing diagram for the CY7C1302, the two-word burst device on the QDR.

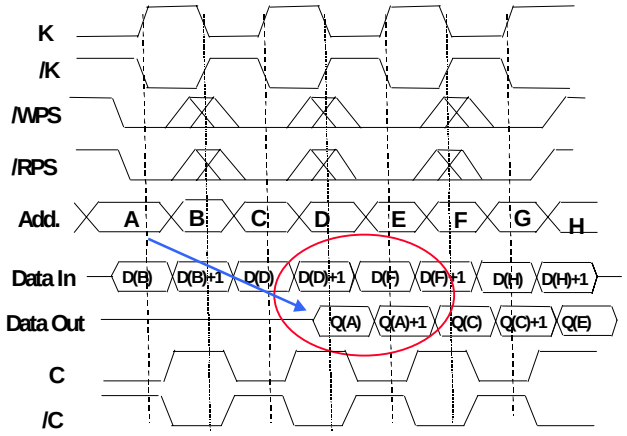

Figure 2. Timing Diagram of the CY7C1302

In the first clock cycle, /RPS and /WPS are both asserted active LOW. The address for the read (A) is latched on the rising edge of the K clock and the address for the write (B) is latched on the rising edge of the /K clock.

The data for the write to address B is also latched on the rising edges of K clock (D(B)) and /K clock (D(B+1). The byte writes are also latched on the same clock as the data,

Read accesses to the QDR_2 are conducted in 2 cycles (a 2-stage pipeline). During the first cycle the address is latched on the rising edge of K clock. The address is then presented to the memory. The next rising edge of K clocks out the first 18-bit data word (Q(A)). The next rising edge of /K clocks out the second 18-bit data word (Q(A+1)).

As seen from the timing diagram, one could start QDR read and write to the same address on the same cycle. When such an operation occurs, the QDR forwards the write data to the read port and ensures that valid data is driven out on the data bus. By doing so, data coherency is guaranteed.

Performance Comparison

Figure 3 shows the performance comparison of the QDR against other SRAMs. The comparison was done assuming that all the interfaces operate at 166 MHz.

The QDR Performance is far ahead of all the other SRAMs and the performance is 4x the closest device in a networking application.

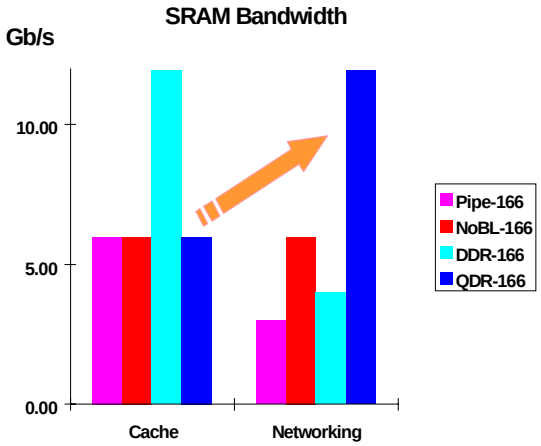

Figure 3. Performance Comparison of the QDR

Spartan-II as a Memory Controller for the QDR SRAMs

To simplify the process of designing an interface to the QDR SRAM, a memory controller was developed using the Spartan II FPGA. A block diagram of the memory controller is shown in *Figure 4*.

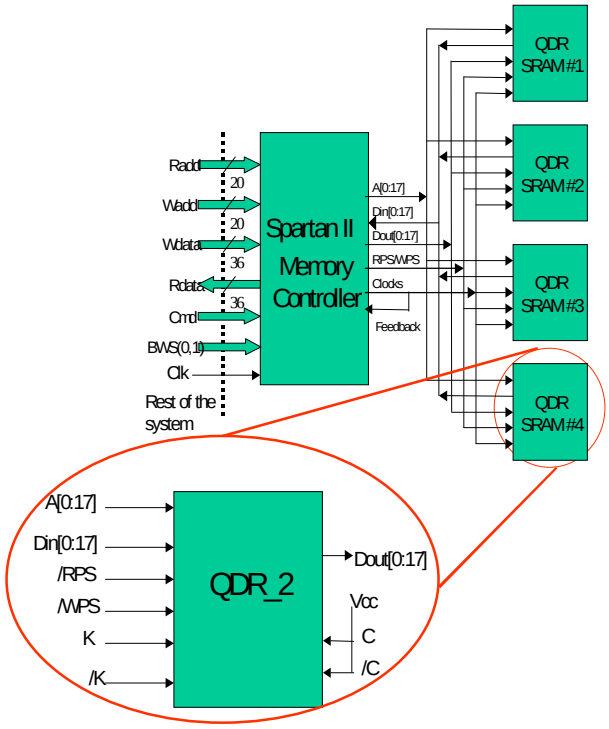

Figure 4. Memory Controller Block Diagram

The memory controller is designed to control four SRAMs in a depth expanded mode. Each QDR SRAM receives separate control signals for the read and write ports, while the address and the data ports are common for all the SRAMs. The SRAMs form a bank of 2M x 18.

The memory controller generates all the control signals for the memory array, The memory controller views the complete SRAM bank like an unified memory array. It supports concurrent double data rate operation on all the inputs and outputs, it also allows byte write operation to the memory bank.

The memory controller assumes that the QDR SRAMs are in the single-clock mode. This can simplify the memory interface. It operates at 100 MHz, allowing a bandwidth of 7.2 Gbits/sec. The memory controller has independent read and write state machines. The memory controller is a command based interface with a two bit command input.

State Machine

Appendix 1 gives the details of the state machine.

Interfacing the QDR SRAM and the Xilinx Spartan-II FPGA

The Spartan-II devices have unique features that simplify the memory controller design. Spartan-II FPGAs offer more than 100,000 system gates at under $10 and are the most cost effective PLD solution ever offered. They build on the capabilities of the very successful Virtex family and support all the associated features including SelectIO, BlockRAM, Distributed RAM, DLLs, and support clock speeds up to 200 MHz. This family of FPGAs support 17 I/O standards and supports VccIO ranging from 1.8V to 5V. The QDR SRAMs with their advanced features and high performance make excellent use of leading-edge programmable logic features present in the Spartan-II family. The DLLs in the design allow for de-skewing of clocks which in normal implementation would need external components. For more information on the Spartan-II family and its advantages as a QDR SRAM Controller, refer to “Spartan-II as a Memory Controller for QDR SRAMs” from Xilinx.

VHDL Implementation

The VHDL implementation of the memory controller can be obtained from

http://www.xilinx.com/products/xaw/qdr/qdrcode.html

When synthesized, the resulting logic diagram of the complete controller shows the memory interface with its three 18-bit buses and the host interface with the dual 36-bit data buses and 18-bit address buses. The Spartan-II FPGA operates internally at 200 MHz. Externally the buses need only operate at 100 MHz, since the DDR interface transfers data on both the leading and trailing edges. The 36-bit read data path from the host is internally split into two 18-bit sections and latched by separate registers. These registers are clocked at 200 MHz, allowing one to send or receive data on both edges of the clock.

The four on-chip DLLs available on the Spartan-II FPGA family can de-skew either the internal global-clock network or clocks fed off-chip to other system components. The two DLLs shown permit the controller to achieve zero clock skew between the FPGAs on-chip clock and the QDR SRAM clock. The Spartan-II DLLs provide additional functions, as well: phase adjustment, frequency division and frequency multiplication.

Figure 5 shows the usage of the DLL in the memory controller design.

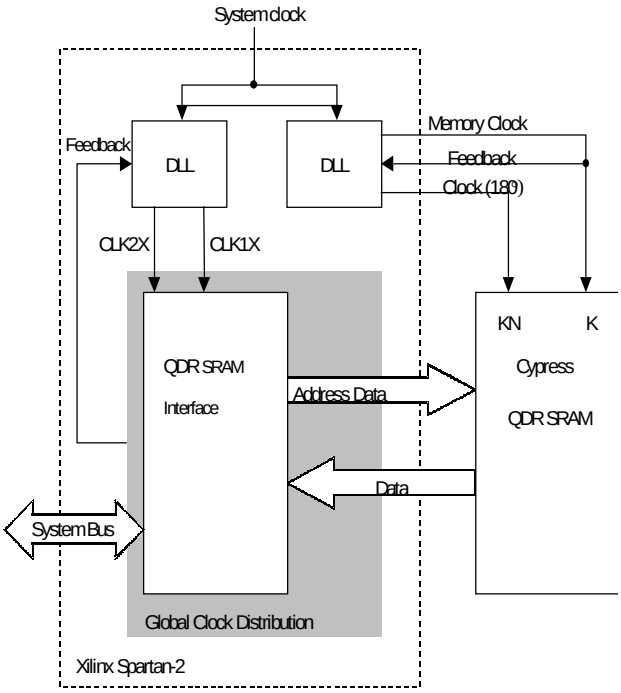

Figure 5. DLL Usage in the QDR Controller Design

Timings

The difficult task for this design is to meet the timing requirements for the CY7C1302. All CY7C1302 signals are registered in the I/O buffers and use HSTL buffers. For the write-cycle timing, all signals must meet those set-up and hold-time requirements. That means dealing with the sum of propagation delays from the Spartan FPGA (Clock to Output), the board wiring delay, and the QDR memory set-up timing. Those delays must total less than the cycle time of the write operation:

Tco(FPGA) + Tpd(Board) + Tsu(QDR SRAM) < Tcyc(write cycle)

2.5 ns + 0.6 ns +0.8 ns = 3.8 ns < Tcyc(5 ns)

The clock-to-out and QDR setup time values are 2.5 ns and 0.8 ns respectively. So there is a good margin for board delay. The QDR memory has a hold-time requirement of 0.5 ns.

During the read cycle, data must meet the set-up and hold time of the FPGA:

Tco(QDR SRAM) + Tpd(board) + Tsu(Spartan-II) < Tcyc(read cycle)

2.5 ns + 0.6 ns + 1.55 ns = 4.65 ns < Tcyc(5 ns)

The set-up time requirement for the Spartan II is 1.55 ns. Along with a clock-to-out timing on the QDR SRAM of 2.5 ns, this demand permits a good margin for operation at 100 MHz.

To implement the controller in the FPGA requires 100 CLB slices, 2 DLLs, 2 global clock buffers and 119 I/O buffers. The design can be verified with a back-annotated simulation at 100 MHz. By using a faster Spartan-II FPGA the interface performance can be improved further.

Appendix 1

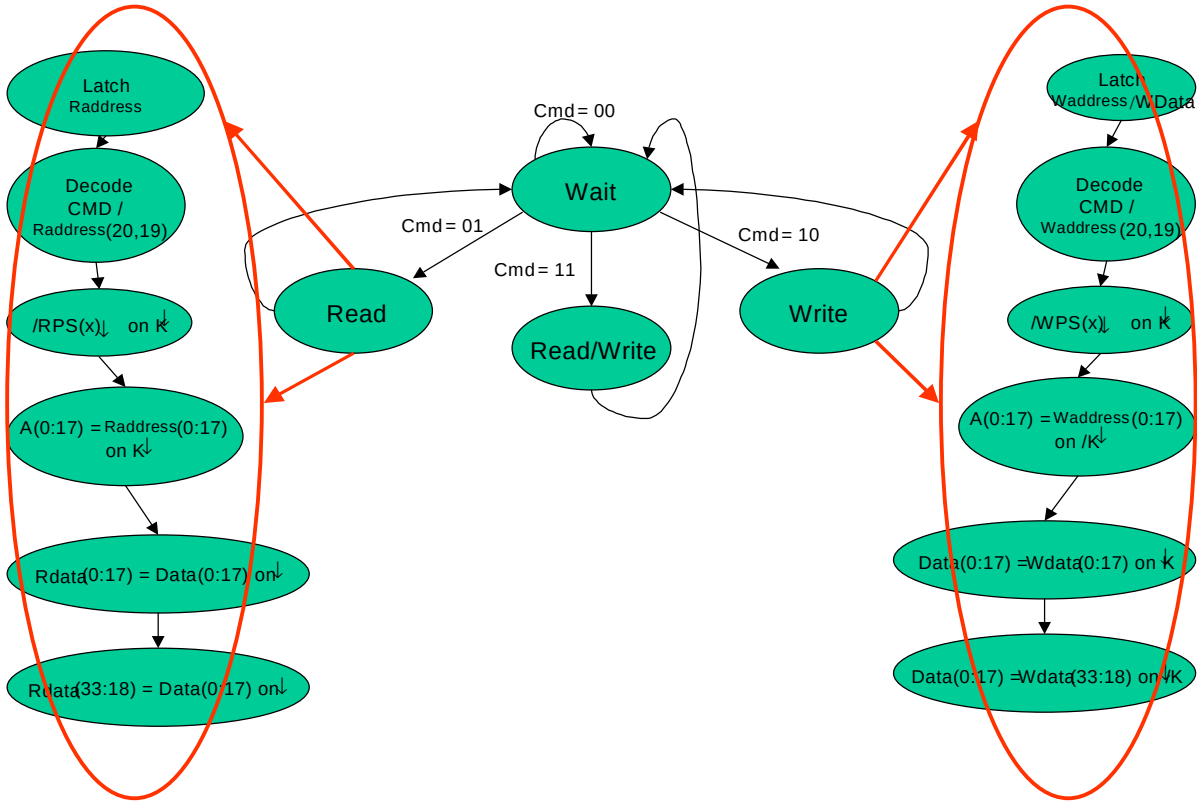

Figure 6. State Machine of the Memory Controller

Appendix 2

More information about the VHDL implementation of the memory controller can be obtained from http://www.xilinx.com/products/xaw/qdr/qdrcode.htm

Information about the customer tutorial for this implementation can be obtained from http://www.xilinx.com/products/xaw/qdr/custtutqdr.pdf