SOLVING HIGH-SPEED MEMORY INTERFACE CHALLENGES WITH LOW-COST FPGAS

A Lattice Semiconductor White Paper

May 2005

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Introduction

Memory devices are ubiquitous in today's communications systems. As system bandwidths continue to increase into the multi-gigabit range, memory technologies have been optimized for higher density and performance. In turn, memory interfaces for these new technologies pose stiff challenges for designers. Traditionally, memory controllers were embedded in processors or as ASIC macrocells in SoCs. With shorter time-to-market requirements, designers are turning to programmable logic devices such as FPGAs to manage memory interfaces. Until recently, only a few FPGAs supported the building blocks to interface reliably to high-speed, next generation devices, and typically these FPGAs were high-end, expensive devices. However, a new generation of low-cost FPGAs has emerged, providing the building blocks, high-speed FPGA fabric, clock management resources and the I/O structures needed to implement next generation DDR2, QDR2 and RLDRAM memory controllers.

Memory Applications

Memory devices are an integral part of a variety of systems. However, different applications have different memory requirements. For networking infrastructure applications, the memory devices required are typically high-density, high-performance, high-bandwidth memory devices with a high degree of reliability. In wireless applications, low-power memory is important, especially for handset and mobile devices, while high-performance is important for base-station applications. Broadband access applications typically require memory devices in which there is a fine balance between cost and performance. Computing and consumer applications require memory solutions such as DRAM modules, Flash cards, and others that are highly cost sensitive while meeting the performance targets for these applications. This article will focus primarily upon memory applications in networking and communications.

Large, fast memory devices are required in networking and communications applications, with tasks ranging from simple address lookups to traffic shaping/policing to buffer management. Inexpensive, legacy Fast-Page mode (FMP) and Extended Data Out (EDO) DRAM used in consumer applications is frequently unsuitable because it is asynchronous (and slower), and requires a precisely-timed series of command signals to initiate data transfers. Networking system architects have traditionally turned to Static RAM (SRAM) to solve latency issues while incurring greater cost. ZBT (Zero Bus Turnaround) SRAM was widely used to improve memory bandwidth by eliminating wait states or idle cycles between read and write cycles.

Recently, system architects have turned to SDRAM for networking architectures, in which reduced latency meets low-cost. Each of these tasks comes with a unique set of requirements. For example,

low- and medium-bandwidth applications require low-latency memory, so ZBT SRAM are ideal. ZBT improves memory bandwidth by eliminating wait states or idle cycles between read and write cycles.

Figure 1 illustrates a typical networking architecture. At 10 Gbps, address lookups with a typical read-write ratio of 1000:1 could easily be handled with Double Data Rate (DDR) SRAM. Link list management, traffic shaping and statistics gathering tasks typically have a balanced 1:1 read-to-write ratio, requiring higher-performance Quad Data Rate (QDR) SRAM. On the other hand, larger buffer memories are typically implemented in DDR SDRAMs (Synchronous DRAMs). A replacement for DRAM, SDRAM synchronizes memory access with a processor clock for faster data transfer. Faster speeds are also achieved because SDRAM allows one block of memory to be accessed while another is being prepared for access. Unlike DRAM, SDRAM uses flowing current rather than a stored charge, eliminating the need for continual refreshing.

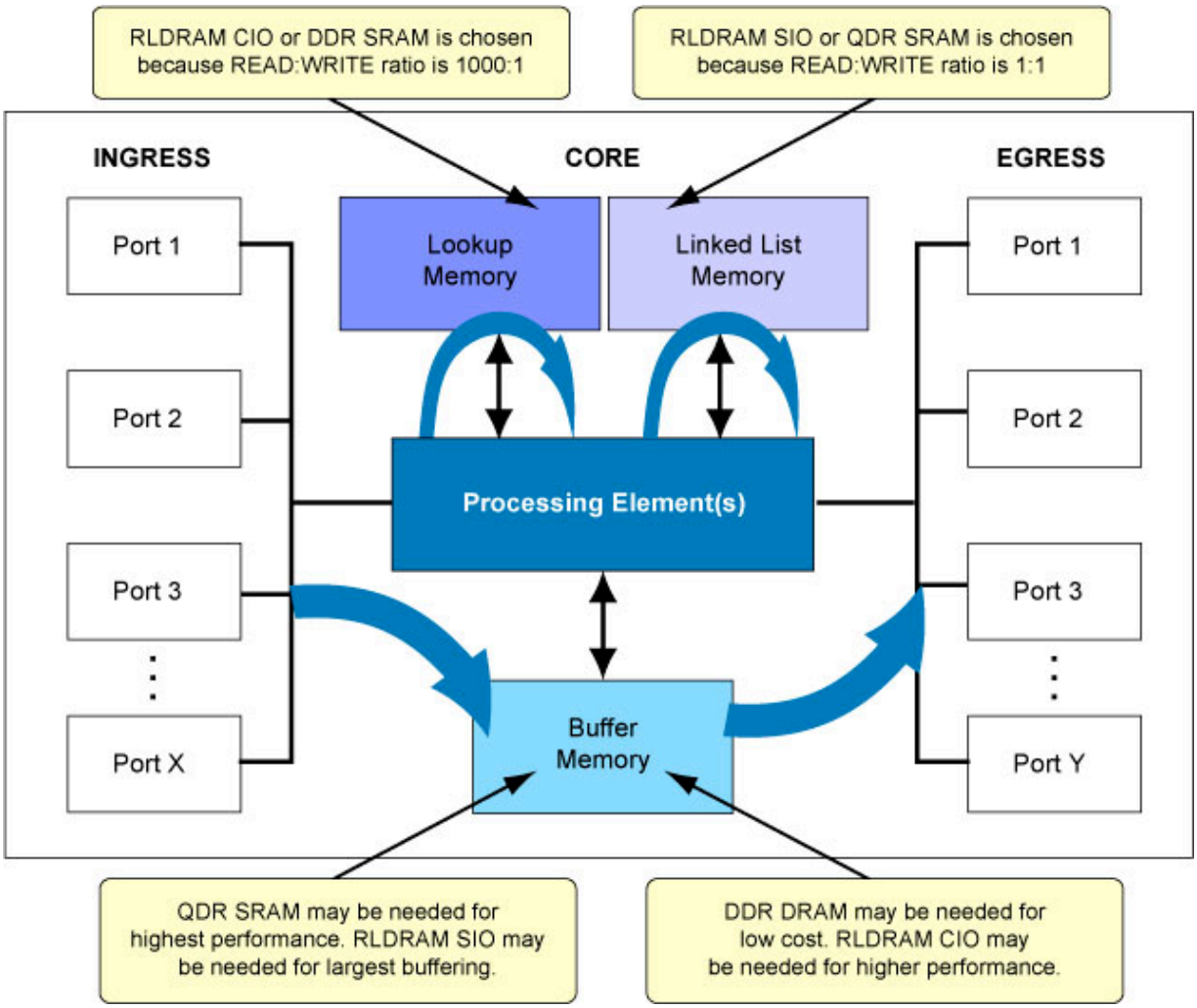

Figure 1 - Memories in Networking: Different Functions Require Different Approaches

Two new contenders have entered the high-performance memory arena in the last few years. Fast Cycle Random Access Memory (FCRAM) improves performance by using a pipeline operation and hidden pre-charge to reduce the random access cycle time, and a highly segmented memory core to

reduce power consumption. The memory core is segmented into smaller arrays such that the data can be accessed much faster and the latency improved. These features make FCRAM ideal for use as buffer memory in high-speed networking applications such as switches and routers, and network servers. Reduced Latency DRAM (RLDRAM) provides SRAM-type interface, non-multiplexed addresses. RLDRAM II technology provides minimized latency and reduced row cycle times that are well suited for applications requiring critical response time and very fast random accesses, such as next generation (10 Gbps and beyond) networking applications.

Table 1 is a comprehensive comparison of the memories employed in high-speed networking applications.

Memory Type	**DDR SDRAM**		**RLDRAM**		**FCRAM**		**QDR SRAM**	
	DDR-I	**DDR-II**	**RL-I**	**RL-II**	**FC-I**	**FC-II**	**QDR-I**	**QDR-II**
Performance (MHz)	100 - 200	200 - 400	200, 250, 300	200, 300, 400	154 - 267	154 - 333	154 -267	154 - 267
Density (Mb)	Components: 64 Mb - 512 Mb DIMMs: 32 Mb - 2 Gb	256 Mb - 2 Gb	256 Mb	288 Mb, 576 Mb	256 Mb, 512 Mb	288 Mb	0.5 – 2Mb	8 - 36 Mb
I/O Standard (V)	SSTL-2.5V, Class I/II	SSTL 1.8V, Class I/II	HSTL 1.8V	HSTL 1.8V	SSTL 2.5, Class I/II	SSTL Class 1.8V, Class I/II	HSTL 2.5 V	HSTL 1.8V
Burst Lengths	2, 4, 8	4, 8	2, 4	2, 4, 8	2, 4	2, 4	2, 4	2, 4
No. of Banks	4	4	8	8	4	4	—	—
Component Data Widths	Components: 8,16,32 DIMMs: 32, 64, 72	Components: 4,8,16 DIMMs: 64, 72	16, 32	9, 18, 36	8, 16	9, 18, 36	8, 9, 18	8, 9, 18, 36,
Read Latency	CAS latency (2, 2.5, 3 clock cycles)	If AL=0 then CAS latency (2, 2.5, 3 clock cycles) + RCD required	CAS latency (5, 6 clock cycles)	CAS latency (5, 6 clock cycles)	CAS latency (3, 4 clock cycles)	CAS latency (4, 5, 6, 7 clock cycles)	1.5 clock cycles	
Read to Write Latency	Burst length/2 + CAS latency cycles	(Burst length/2) + read latency cycles	1 clock cycle while addressing DIFFERENT banks, 8 clock cycles while addressing SAME bank		(Burst length/2) + CAS latency cycles		None	
Write to Read Latency	1 clock cycle AFTER write burst	2 clock cycles	1 clock cycle while addressing DIFFERENT banks, 8 clock cycles while addressing SAME bank		2.5 clock cycles		—	—
Row Cycle Time	12 clock cycles	13 - 15 clock cycles	5 - 8 clock cycles		5 clock cycles		—	—
Data Strobe	Non-free running	Differential strobe - Non-free running	Free running	Differential strobe - Free Running	Non- free running	Unidirectional: Free or Non-Free running	Free running Read & Write clocks	
Data Bits Per Strobe	8, 16, 32	8	8, 16 bit-wide device	9, 18 (for 36 bit-wide device)	8	9, 18 (for 36 bit-wide device)	One differential pair per device	

Read Data Valid	Data valid after CAS latency cycles		Data valid signal from memory		Data valid after CAS latency cycles		—	—
Data Access	Row activation required before column can be accessed		Row and column can be addressed together.		Row activation required before N/A column can be accessed		—	—
External Precharge Required	Yes	Yes	No	No	No	No	No	No
Initialization Sequence	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Auto Refresh Required	Every 15.6 ms or up to 140.6 ms during auto refresh	Every 7.8 ms	Every 8 ms	Every 3.9 ms	Every 7.8 ms	Every 3.9 ms	—	—

Table 1 - Comparison of Memories Employed in High-speed Networking Applications

Current memory interfaces often require clock speeds in excess of 200 MHz to satisfy the throughput requirements of line and switch cards. This is a major challenge in FPGA architectures. PLLs are essential to allow control over the clock data relationship.

Next-generation memory controllers operate at HSTL (High-Speed Transceiver Logic) or SSTL (Stub-Series Transistor Logic) voltage levels. This lower voltage level swing is required to support the high-speed data operation of the inputs and outputs of the memory device (and the memory controller). HSTL is the de-facto I/O standard for high-speed SRAM memory devices, while SSTL is the de facto I/O standard for high-speed DDR SDRAM memories.

This combination of high-speed differential I/O buffers and specialized circuitry (discussed below) to enable seamless read and write operation at high bandwidths has traditionally been the domain of premium FPGAs. Designers no longer wish to pay high prices for utilitarian functions like a memory controller. Several FPGA vendors have responded with low-cost solutions. One solution employs a soft IP to manage the timing relationships and interface between the memory and controller. As with any soft IP solution, this challenges the designer to place & route the IP core within the proprietary design.

In contrast, other vendors have embedded specialized hardware in low-cost FPGAs to manage the memory interface. This approach eliminates design concerns about the need to map, place, route and meet timing requirements on a critical interface.

DDR Memory

In a typical non-DDR system, both the controller and memory in a system transmit or capture data in response to a *single system clock* (Figure 2). Designers became familiar with the timing constraints in these systems, which, over time, have become tighter as clock speeds have increased.

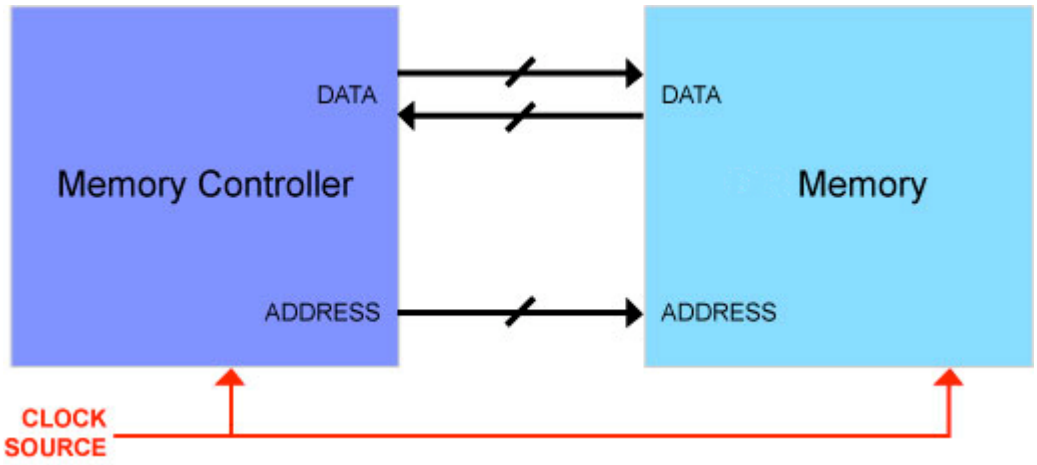

Figure 2 - Typical Non-DDR System

As designers moved to a DDR memory implementation, data rates effectively doubled. The downside (for the designer) was that this effectively cut the data window in half. To accomplish this method of data transfer, DDR SDRAM interfaces rely on the use of a data strobe signal (DQS). DQS is generated from the differential clock fed to the DDR memory and a DLL inside the DDR memory is used to generate and align DQS to outgoing data. Although a DDR memory does not use the differential input clock to launch or capture data, the DQS signal that is used is related to the incoming clock frequency.

The DQS signal has several characteristics:

- DQS is bi-directional
- A DQS line is typically generated for 8 lanes of data from the DDR memory
- The phase of DQS relative to the data depends on the operation being performed (Write or Read)
- DQS is not free running
- In the memory device, DQS is generated by DLL to minimize the skew between it and data
- DQS has a *Preamble* state just after the signal comes out of tristate where DQS goes low
- DQS has a *Postamble* state just before returning to tristate where DQS goes low

Figure 3 depicts a DDR memory and FPGA controller with the associated data and control lanes. Note that the clock signal provided to the memory is differential (CLK/CLKN) to minimize duty cycle variations. It is important to understand that it is *not* the clock signal that is used to capture or launch data to or from the memory. DDR memory also requires a Data Mask (DM) signal, which is used to mask data bits during the write cycles. This allows writes to the memory on only one of the two edges of DQS that occur in a cycle.

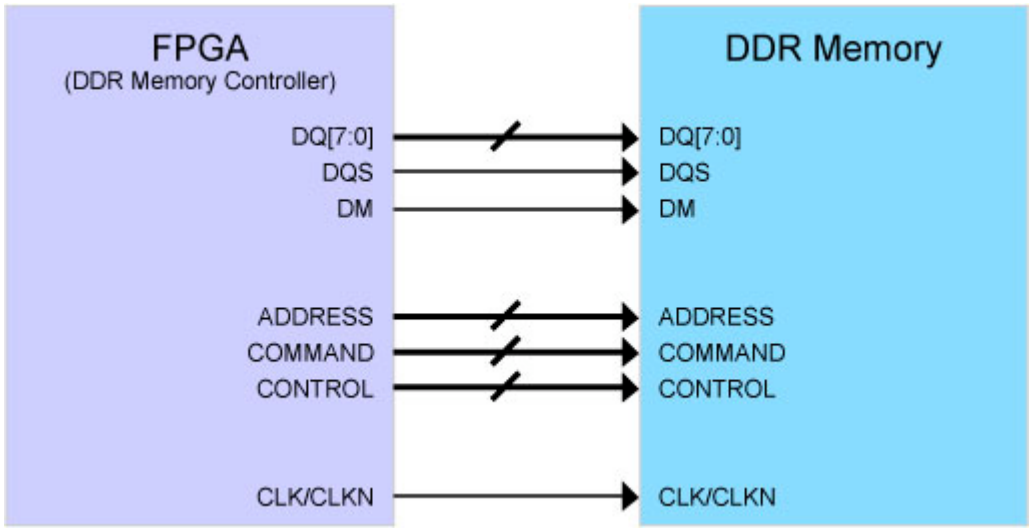

Figure 3 - Typical DDR interface

SDRAM interfaces typically are implemented with x8, x16 or x32 bits for each DQS signal. Note that the ratio of DQS to data bits is independent of the overall width of the memory.

DDR Memory Controller Support in FPGAs

Implementing high performance DDR memory or write interfaces requires dedicated DDR register structures in the inputs (for read operations) and in the outputs (for write operations). Embedded hardware can provide this capability while freeing up FPGA logic for vital user-defined logic. Additional elements are required to simplify the design of input structures for read operations: a DQS delay block and polarity control logic. These blocks are critical for implementing reliable high-speed DDR SDRAM Controllers.

DLL Calibrated DQS Delay Block

Source Synchronous interfaces generally require the input clock to be adjusted in order to correctly capture data at the input register. For most interfaces, a PLL is used for this adjustment; however, in DDR memories the clock (referred to as DQS) is not free running, so this approach cannot be used. In DDR memory interfaces the DQS to Master Clock relationship varies due to factors such as PCB trace length and the memory device being used.

Figure 4 illustrates a method for including automatic clock transfer circuitry that simplifies the memory interface design and ensures robust operation. Additionally, the DQS Delay block provides the required clock alignment for DDR memory interfaces. The DQS signal feeds from the PAD through a DQS delay element to a dedicated DQS routing resource. The DQS signal also feeds polarity control logic, which controls the polarity of the clock to the sync registers in the input register blocks.

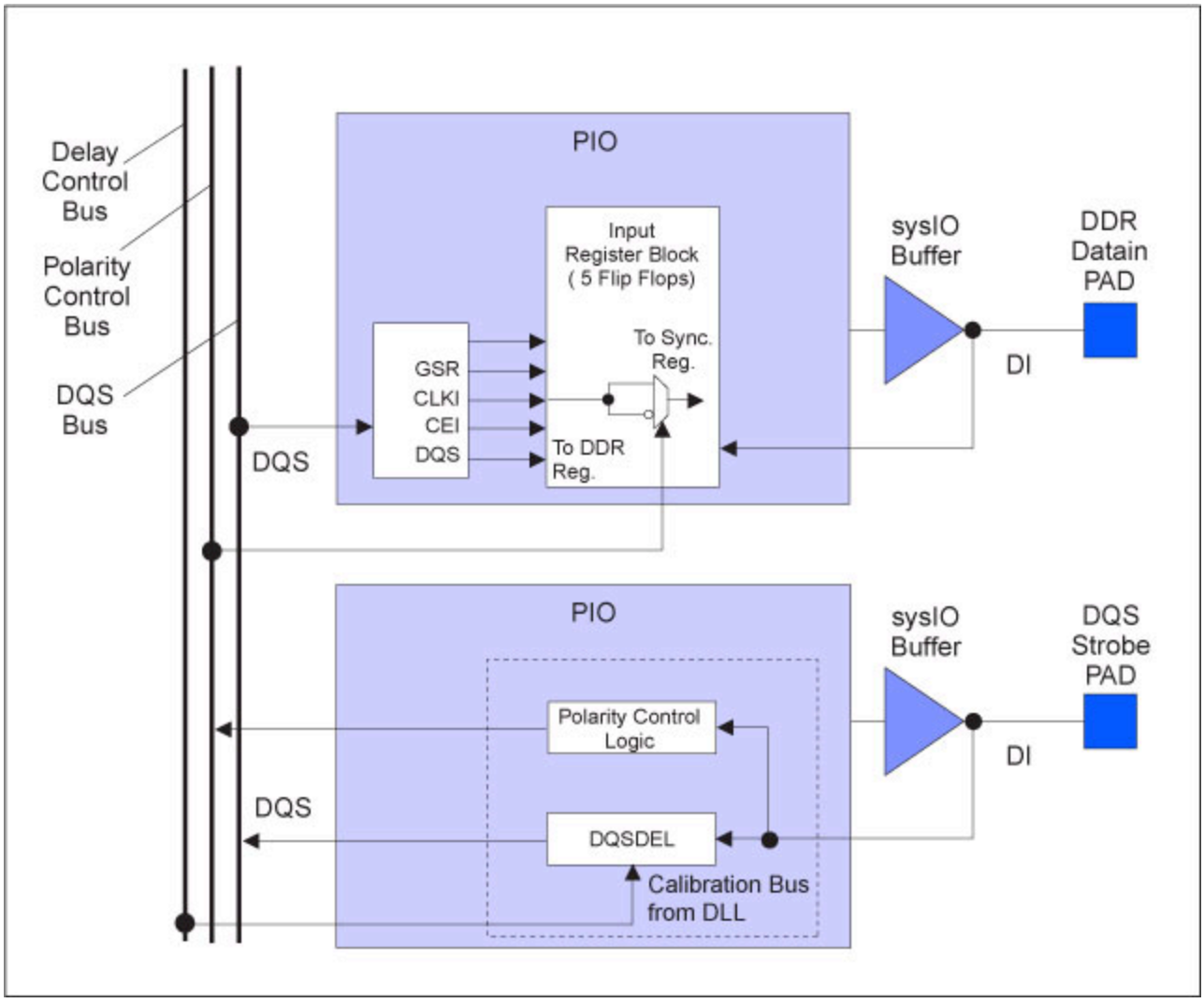

Figure 4 - Block Diagram of Dedicated DQS Circuitry

Temperature, voltage and process variations of the dedicated DQS delay block are compensated for by a set of calibration (6-bit bus) signals from two DLLs on opposite sides of the device. Each DLL (Figure 5) compensates DQS Delays in its half of the device. The DLL loop is compensated for by the system clock and dedicated feedback loop. This is an important architectural feature, because the device is not hampered with the stringent I/O layout requirements common with other FPGAs.

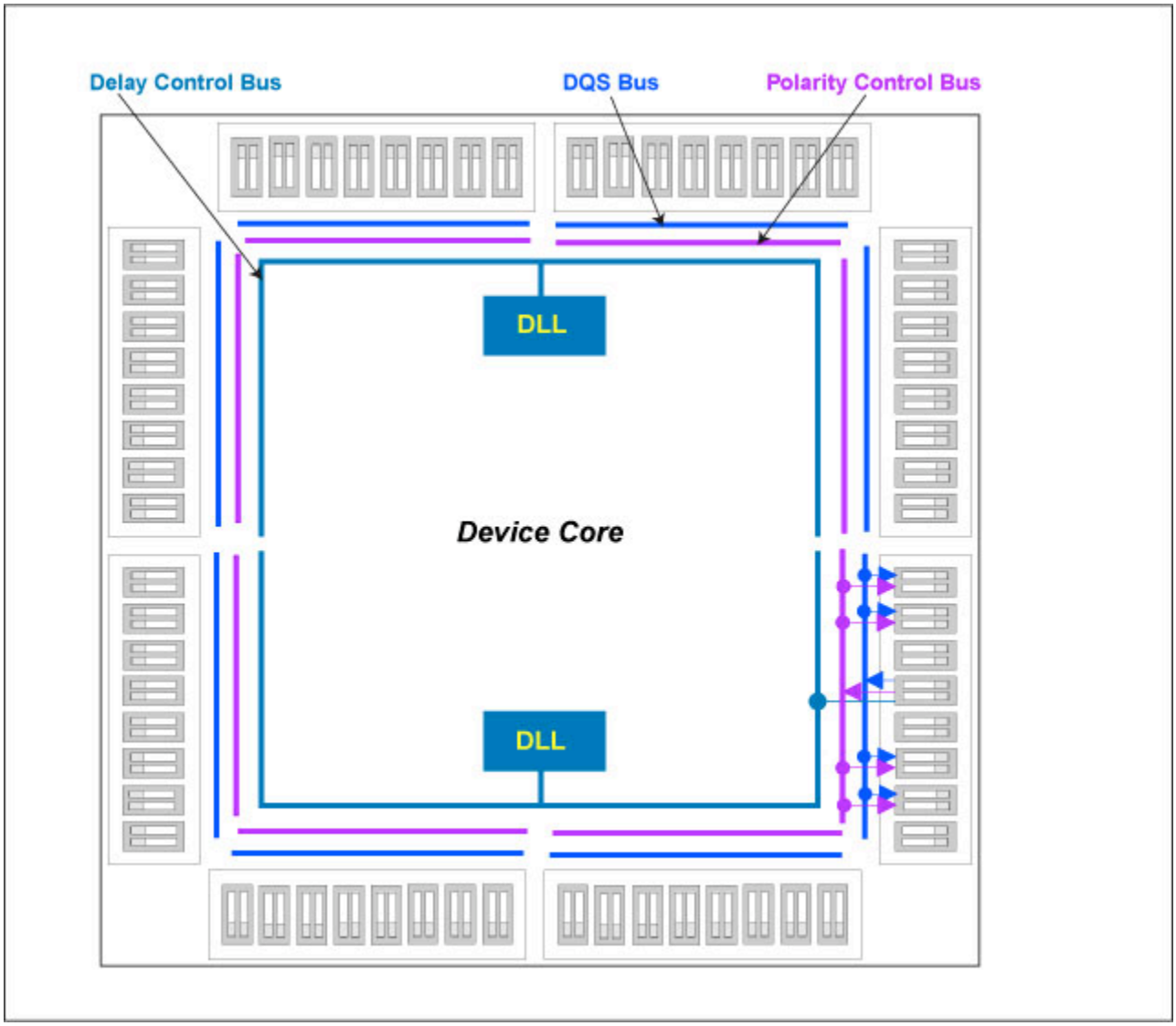

Figure 5 - DLL Calibration Bus and DQS/DQS Transition Distribution

Polarity Control Logic

In a typical DDR Memory interface design, the phase relation between the incoming delayed DQS strobe and the internal system Clock (during the Read cycle) is unknown. For embedded hardware to be used to prevent setup and hold violations at the domain transfer between DQS (delayed) and the system clock, a clock-polarity-selector is used. This changes the edge on which the data is registered in the synchronizing registers in the input register block. This requires evaluation at the start of each Read cycle of the correct clock polarity. Prior to the Read operation in DDR memories, DQS is in tri-state (pulled by termination). The DDR memory device drives DQS low at the start of the preamble state. A dedicated circuit detects this transition. This signal is used to control the polarity of the clock to the synchronizing registers.

Conclusion

As line rates continue to grow, DDR SDRAMs are experiencing wider adoption in networking applications. These increasing system bandwidth requirements are pushing memory interface speeds while costs continue to be driven down. Some FPGA vendors are responding with low-cost devices offering soft-IP solutions. On the other hand, Lattice Semiconductor employs built-in circuitry in its new low-cost LatticeEC and LatticeECP families.

###