High Performance DDR4 interfaces with FPGA Flexibility

Adrian Cosoroaba and Terry Magee
Xilinx, Inc.
AGENDA

- System Requirements for FPGA based systems
  - Higher Bandwidth, Increased Flexibility, Lower Power
- UltraScale FPGA Solution for DDR4 and other parallel memory interfaces
- PHY Solution for higher performance with maximum flexibility
- Multiple Solutions for Different Needs
Bandwidth Driven Systems Demand Extensive Memory Buffering

- Rapid growth in serial I/O bandwidth is enabling next generation systems
- External memory buffering rate must match or exceed traffic rate

Bandwidth-Driven Systems

<table>
<thead>
<tr>
<th>100G/200G Networking</th>
<th>Semiconductor ATE</th>
<th>8x8 Multi-Mode Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super Hi-Vision Camera</td>
<td>Medical Imaging</td>
<td>Radar Processing</td>
</tr>
</tbody>
</table>

External Memory Buffer Example

- Traffic In
- Internal Traffic Processing
- Parallel I/O
- External Memory Buffer (e.g., DDR3, DDR4)
- Traffic Out
FPGA Memory Interface Bandwidth

- Data rate per bit
- Number of Interfaces (improves with better usage of I/O)
- Data bus width
- Data bus efficiency (percentage of time that data bus is utilized)

Total Effective FPGA Bandwidth =

\[ [\text{Data Rate}] \times [\text{Number of Interfaces}] \times [\text{Data Bus Width}] \times [\text{Data Bus Efficiency}] \]
Higher DDR4 Data Rates

➢ 30% higher data rates vs. DDR3
  – 2400 Mb/s (UltraScale devices) vs. 1866 Mb/s (7 series devices)

➢ Available in both mid- and high-speed grade
  – Lower cost benefits of a flexible architecture
UltraScale FPGAs Have Been Architected to Maximize Bandwidth

- Higher data rate support with advanced IO features
- Flexible PHY and I/O architecture for highest interface count and bus width
- Flexible memory controller for greatest data bus efficiency
New PHY Architecture for Higher Performance with Extended Flexibility

**Performance**

- Next generation PLLs, clocking modules, DLLs ⇒ high performance
- Hardened data path ⇒ low latency
- Isolated power supplies ⇒ optimized timing & minimized jitter

**Flexibility**

- 52 pins per I/O bank ⇒ enables 16-bit interfaces in one I/O bank
- 5 ps tap delay ⇒ allows fine tuning of clock-to-data alignment
- Two PLLs per I/O bank ⇒ enables two independent interfaces per bank
Flexible I/O Architecture for Maximum Interface Count and Memory Capacity

Dense packing of memory interfaces
- I/O bank can be split between 2 interfaces & rates
- I/O bank can be split at any byte lane boundary

Higher density DIMM support
- X4 based DDR3/DDR4 support
- Dual and Quad rank DDR3/DDR4 support
Flexible Memory Controller for Greatest Data Bus Efficiency and Flexibility

Bandwidth Efficiency

- Leverages new DDR4 bank grouping ⇒ high performance
- Optimized command queue reorders commands and groups RD/WR’s ⇒ low bus turnaround
- Improved internal clock timing ⇒ shorter command-to-read data latency.

Flexible Customization

- Easily customize page management ⇒ match application specific command patterns
- AXI user interface ⇒ easily interface with other AXI IP or enable a multi-porting capability
Lower Power with DDR4

DDR4 benefits for up to 20% Lower Power

- 1.2V vs. 1.5V I/O
- Pseudo Open Drain (vs. SSTL)
- Data Bus Inversion (DBI)

UltraScale architecture for lower power

Power Savings with DDR4 and UltraScale FPGAs (Using XPE 2014.1)

<table>
<thead>
<tr>
<th>Memory Interface, FPGA</th>
<th>Data Rate, Data Width</th>
<th>I/O and PHY Power</th>
<th>% Power Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3, 7 series FPGA</td>
<td>1866 Mb/s, 32 bits wide</td>
<td>1.41W</td>
<td>-</td>
</tr>
<tr>
<td>DDR4, UltraScale FPGA</td>
<td>1866 Mb/s 32 bits wide</td>
<td>0.94W</td>
<td>36%</td>
</tr>
<tr>
<td>DDR4, UltraScale FPGA</td>
<td>2400 Mb/s 32 bits wide</td>
<td>1.10W</td>
<td>15%</td>
</tr>
</tbody>
</table>
Designed for Flexibility

- Soft Controller and PHY configurability needed for FPGA Applications
- PHY designed for DDR4 but flexible to support other memories
- Networking Applications
  - RLDRAM 3
  - QDR II+ / QDR IV
- Low Power Applications
  - LPDDR3

<table>
<thead>
<tr>
<th>UltraScale Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR4-2400 Mbps</td>
</tr>
<tr>
<td>DDR3-2133 Mbps</td>
</tr>
<tr>
<td>DDR3L-1866 Mbps</td>
</tr>
<tr>
<td>RLDRAM 3</td>
</tr>
<tr>
<td>QDR II+/QDR IV</td>
</tr>
<tr>
<td>LPDDR3</td>
</tr>
</tbody>
</table>
PHY Solution for higher performance with maximum flexibility
Overview

- FPGA Flexibility is a challenge to high-speed PHY design
- Primitive-based PHY structures are reaching their limitation
- Ultrascale turns the problem on its head
- Integrated 13-bit ByteLane is first and foremost a DDR4 PHY
- Re-use for all FPGA’s needs
FPGA Flexibility and High-Speed Memory support

- FPGAs today must provide a high-speed I/O interface
  - to connect its programmable logic with external Memory (DDR3/4 etc)

- I/Os are a scarce resource. I/O interface must be shared with ...
  - General source synchronous interfaces (Video, Networking, etc ...)
  - Simple LED drivers
  - Direct access to the programmable logic
  - and so on ...

- Xilinx FPGAs solve this need with custom low-level Primitives
  - Primitives can be wired and configured in multiple ways
  - They perform a host of I/O access functions
The Memory PHY challenge with FPGA Primitives

- With flexibility comes compromise
  - A selection of primitives that can be configured and wired in multiple ways will ultimately compromise on high-speed performance
  - With care and some dedicated circuits FPGA’s have met this challenge for DDR3

- DDR4 needs a new approach
A crude example of a DDR3 PHY designed based on custom-design primitives today.
Turn the problem on its head …

Rather than build a PHY from FPGA primitives …

Build an **ASIC-quality PHY** that is *first and foremost* a DDR4 PHY

- From hierarchical rtl and custom blocks
- … but a single hardened integrated macro
- Don’t worry about non-memory configuration at first
- Achieve DDR4 performance, power, and area goals
- Only then …
  - Map the customer-visible primitives to the PHY’s hierarchy via Software
  - *Software creates a virtual view of granular bit-level primitives from a single integrated macro*
- Add configurability intelligently *without* compromise to high-speed paths
Integrated PHY for Ultrascale family

- Ultrascale DDR4 PHY is a complete integrated solution
Four 13-bit wide Byte Lanes (PHYs) per I/O Bank
Why a 13-bit wide Byte?

- Best size for most efficient pin-packing

- 2 bytes = 26 bits of Command/Address/CK
  - Good for most DDR3/4 Command/Address
  - Any Overflow can be in a Data Byte

Data Byte:
- DDR: 11 I/Os (8DQ, 2DQS, DM/DBI)
- Free I/Os for CA overflow (>26 bits)
  - Or ext. reference resistor
  - Or random use (reset, etc)
- QDR;RL3: 12 I/Os (9 data, 2 clocks)
Non-memory – Let’s not forget we’re an FPGA

- S/W maps the per-pin primitives to our high-speed DDR4 PHY

- For Non-memory source synchronous interfaces (HDMI, SPI4.2, ADC’s, etc)
  - *For the first time non-memory customers have access to our high performance memory solution*

- Or simple registered or combinatorial programmable logic access
  - *Unbeknownst to them, customer is using a high-performance DDR4 PHY to drive an LED!!*
Proprietary Built-in Self Calibration
- 5ps per tap Data and Clock/Strobe delays are auto-calibrated … 
  … to balance silicon skews and insertion delays on bring-up 
  … to remove process variation on critical data and clock paths 
- Programmable delays are custom-designed for linearity and monotonicity 
- Low-power DLLs track VT to within 5ps

Isolated power and custom clocking
- Extremely low jitter on data and clock outputs and internal Read capture 
- Extremely low DCD on data and clock outputs 
- DCD immune for READ capture
Result …

➢ 72-bit DDR4 @2400Mbps

WRITE (@ DRAM ball)
Multiple Solutions for Different Needs
Multiple Solutions for Different Needs

<table>
<thead>
<tr>
<th>Performance</th>
<th>LPDDR3</th>
<th>DDR4</th>
<th>RLDRAM3</th>
<th>QDR</th>
<th>HMC</th>
<th>MoSys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Data Rate (Bandwidth)</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Latency</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Access rate</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Power-per-device</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Cost</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Xilinx provides broad support for diverse memory interface requirements
Thank You!