Quantifying Effective Memory Bandwidth of Platform FPGAs

Andrew G. Schmidt  
University of Kansas  
2335 Irving Hill Road  
Lawrence, KS 66045–7523  
aschmidt@ittc.ku.edu

Ron Sass  
University of North Carolina, Charlotte  
9201 University City Blvd.  
Charlotte, NC 28223–0001  
rsass@uncc.edu

1. Introduction

The benefits of High Performance Computing (HPC) can be seen in a wide range of applications. From science and medicine to industries as diverse as oil exploration, financial, and entertainment [5], access to cost-effective HPC is becoming a critical part of our national infrastructure. Although exponential semiconductor advances are giving computational scientists faster computing speeds, applications with large amounts of data may not necessarily solve problems faster. More specifically, technology trends are working against system designers: computation rates and memory capacity are both rising faster than the bandwidth between these two components. This so called “Memory Wall” was predicted for general-purpose computing over a decade ago [8], but to date large on-chip caches are able to compensate for the growing disparity. FPGA designers do not have the luxury of large caches, so to be successful, high-performance designs must include custom memory hierarchies and data paths as well as application-specific computations.

Platform FPGAs are now capable of hosting entire systems (and, indeed some have proposed using collections of these systems to build HPC clusters). While tools and a rich catalog of IP cores are available to assemble these systems, many of the key system components — such as buses, bridges, and (off-chip) SDRAM controllers — were intended for embedded systems. While the conventional two-bus (system and peripheral) structures serve embedded systems well, it is not clear how well these components will work for HPC designs. Moreover, even though individual components are well documented, it is not always possible to determine analytically how a system of components will behave.

The focus of this project is to investigate the various on-chip memory organizations to improve off-chip memory performance in HPC designs. Two contributions of this extended abstract are (i) characterizing the behavior of common organizations using off-the-shelf cores and (ii) the development of a testing methodology which simulates different access patterns and accurately measures bandwidth. Ongoing and future work will synthesize a set of memory hierarchy design rules for HPC designs and investigate novel on-chip organizations to improve the efficiency of the off-chip memory interface.

2. Implementation

Modern DRAMs provide a large collection of addressable memory cells. However, the time to access an arbitrary cell is not uniform. For example, based on DDR SDRAM’s operational characteristics [1], sequential access will significantly outperform random accesses. Moreover, many HPC applications are neither sequential nor random, rather something in between. Thus, to accurately assess performance it is necessary to develop a special-purpose core that could be configured to generate memory access patterns commonly associated with HPC applications. These patterns include: sequential, strided, and random access. In sequential access burst mode is also available to moves multiple, adjacent words to or from memory.

In addition to different memory access patterns, a two-bus system architecture allows for a number of different system organizations. The memory controller — an IP core capable of translating bus requests into off-chip memory accesses — can be placed on either the peripheral (OPB) bus or on the system (PLB) bus. Likewise a HPC core can be placed on either bus. This leads to four possible combinations; shown in Figure 1(a). A single HPC core accessing off-chip memory is tested here. Details about the operation of these cores can be found in [7].

The experiments were tested on one node of the Reconfigurable Computing Cluster under construction at the University of North Carolina, Charlotte. The prototype node is a Xilinx ML-310 board with a Virtex-II Pro (XC2VP30) FPGA and Windtec 184-pin DDR400 PC3200 SDRAM DIMM.

3. Results

Although IBM’s CoreConnect documentation indicates that the PLB has a peak bandwidth of 1600 MB/s (533 MB/s typical) and that the OPB has 500 MB/s peak bandwidth (167 MB/s typical) [4], these experiments use a single 100 MHz clock throughout the design. This lowers the peak theoretical bandwidths to 800 and 400 MB/s respectively, and is representative of many target speeds for HPC designs. For purposes of comparing different organizations, this lower frequency is acceptable.

Perhaps the most surprising result, which is summarized in Figure 1(b), is the order in which memory is accessed is at best
a secondary factor of the overall performance. That is, the bandwidth limitations of using a bus and its protocol are so substantial that an intelligent HPC core can only make modest improvements in RAM bandwidth by scheduling its memory accesses if a typical bus structure is employed. Indeed, the effective bandwidth of a core performing non-burst, strided, or random access patterns is at best a mere 24.85 MB/s or 3.11% of the theoretical bandwidth of the bus. When considering the theoretical bandwidth of a single DDR SDRAM (nominally between 1600 MB/s and 3200 MB/s according to the data sheet), a single core’s effective bandwidth is between 0.78% and 1.55%!

Burst mode transactions make a substantial improvement, providing up to 199.71 MB/s or 24.96% of the bus’s theoretical bandwidth. This result is as expected since a single burst request can deliver up to 128 bytes of data and each word retrieved is on average twice as fast as a non-burst transaction. However, burst mode transactions only work for sequential access and are limited to lengths of 128 bytes. In addition, burst mode transactions increase HPC core design complexity which burdens hardware designers. If the application does not have a great deal of sequential accesses, then burst transactions will probably not benefit the application.

The OPB and PLB are limited in the number of cores that may master the bus. To support a larger number of cores, typical designs use a two-bus approach with bridges to connect the two. In the case of the off-the-shelf embedded systems cores, the bridge limits the bandwidth to 9.52 MB/s, illustrating the impact of the core’s location on bandwidth.

4. Related Work

Many research projects are investigating memory subsystems and in particular, HPC memory bandwidth. The Merrimac project at Stanford has investigated computer architectures and novel languages in attempts to use bandwidth more efficiently [2]. With the advent of multi-core chips a number of industrial efforts are looking at interconnection schemes such as dual-channel, fully-buffered, and direct access. Memory buffering, re-timing, and various ways to access data in sliding windows for FPGAs have also been extensively covered in the literature.

Research more closely related to this work have looked at the efficiency of communications between an embedded processor and reconfigurable logic [6] and providing more efficient communications between the embedded processor and off-chip memory [3]. The focus of this work is the performance between the reconfigurable logic and off-chip memory since a specific application or process in platform FPGAs is not given exclusive access to off-chip memory — which is why a bus is often used.

References