Performance Analysis of SGI RASC RC100 Blade on 1-D DWT

Abhishek Mitra, Ge Yao
Dept of Computer Sc, UC Riverside
amitra@cs.ucr.edu, gyao@cs.ucr.edu

Walid Najjar
Dept of Computer Sc, UC Riverside
najjar@cs.ucr.edu

Abstract

We analyze the performance of a high performance Reconfigurable Computing system viz. the SGI RASC RC100 Blade. We calculate the interface throughput, as well as its effect on the performance of 1-D 5-3 DWT algorithm compiled using ROCCC. We also demonstrate that with large image sizes, it is possible to achieve a speedup of 16.5 times when compared to the software implementation running on Intel Core2Extreme CPU.

1. Introduction

FPGA based reconfigurable computing systems are being commonly used to achieve speedups ranging between 10x to 1000x for various applications. FPGAs allow speedup of slow sequential software by efficient hardware execution. The typical application scenario includes profiling the software code to ascertain the slowest executing components and compiling the software code to equivalent HDL. Thereafter the HDL code is synthesized and implemented on FPGA. Simulation after Place and Route can provide an accurate representation of the theoretical speedup, provided by the FPGA. However when the actual design is actually implemented on a Reconfigurable Computing system, the end to end speedup may be quite less due to the issues such as data transfer overhead. This paper tries to highlight the performance of 5-3 1D DWT (Discrete Wavelet Transform) in practice.

2. System Overview

We target the SGI RASC RC100 Blade, consisting of two Virtex-4 LX 200 FPGAs, with 40 MB of SRAM logically organized as two 16MB blocks (as shown in Fig. 1) and an 8MB block. The SRAM are 36Bit QDR devices with 4 bit parity, thus transferring 128bit data every clock cycle. The RC100 Blade is connected using the low latency NUMALink interconnect to the SGI ALtix 4700 Host System, for a rated peak bandwidth of 6.4GB per second. Around one-tenth of the area on each FPGA is

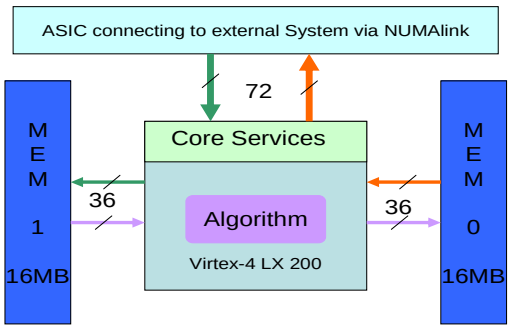


Figure 1: System Overview of an FPGA on the SGI RC100 Blade

used by the Core Services which is the logic used to interface the FPGA to the NUMALink and SRAM memories. The HDL code representing the algorithm is synthesized along with the Core Services, and the resulting bitstream is programmed on the FPGA.

2.1 System Interfaces

The RASC Abstraction layer provides various DMA based I/O interfaces to the FPGA on the RC100 Blade [1].

2.1.1. Direct I/O

Direct I/O involves allocating memory for input and output data on the hugetlb space using appropriate API calls. The data is transferred from the host memory in the hugetlb to the SRAM on the RC100 Blade (Fig. 1 Green Lines). Once the input data is transferred to the SRAM, the FPGA starts processing and stores the output onto the on-board SRAM (Fig. 1 Purple Lines). After conclusion of processing the data by the FPGA, output data is transferred back to the host memory in the hugetlb (Fig 1. Orange Lines). On the RASC blade it is possible to achieve 2.2GByte/s throughput (Fig. 2) when more than 1MByte is transferred in one go.

2.1.2. Buffered I/O

Buffered I/O works similar to Direct I/O except that it

involves allocating memory for input and output data in the user space on the host. Unfortunately transferring the data to / from the SRAM on RC100 blade using buffered I/O involves an additional copy in the kernel memory space, thus lowering the maximum throughput to only 1.4GByte/s (Fig. 2) on the RC100 blade. Buffered I/O is best avoided, unless hugetlb system is unavailable on the host.

2.1.3. Streaming DMA I/O

Streaming I/O does away with using the SRAMs on the RC100 blades for transferring data, though the SRAMs may be still be used by the algorithm hardware for local storage. On the RASC blade, it is possible to achieve a throughput of 2.4GByte/s using streaming.

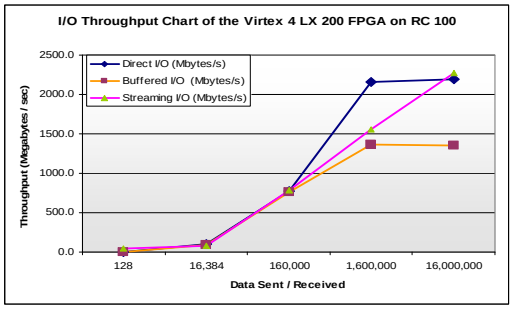


Figure 2: Read / Write Throughput using three I/O interfaces to RC100 on SGI Altix 4700 platform

3. ROCCC and 1D DWT

ROCCC (RIVERSIDE OPTIMIZING COMPILER FOR CONFIGURABLE COMPUTING) is built on the SUIF2 and Machine-SUIF platforms. It compiles C code into VHDL for mapping onto the FPGA fabric. ROCCC performs unrolling and also a very extensive set of loop analysis and transformations, aiming at maximizing parallelism and minimizing area.

```
for(i=0;  i<LEN ; i=i+4)
{
    Lift1_2 = a[i+1] - (a[i]+a[i+2])/2;
    Lift1_3 = a[i+3] - (a[i+2]+a[i+4])/2;

    Lift2_1 = a[i] + (Lift1_2 + Lift1_1 + 2)/4;
    Lift2_2 = a[i+2] + (Lift1_2 + Lift1_3 + 2)/4;
    Lift1_1 = Lift1_3;

    b[i] = Lift1_2;
    b[i+1] = Lift1_3;
    b[i+ROW/2] = Lift2_1;
    b[i+1+ROW/2] = Lift2_2;
}
```

Figure 3: The C code of the lifting implementation of 5-3 DWT, used for accelerating on RC100

We have utilized the 5-3 DWT [2] algorithm based on the lifting scheme. The C code used to generate VHDL using ROCCC is illustrated in Fig. 3. We prepare the 9-bit input image data from OpenJPEG using 16-bit representation in C. The algorithm in Fig. 3 processes 5 pixels and generates 4 output pixels each clock cycle. Since the hardware transfers 128bit data (8 pixels) every clock cycle, we unroll the loop [3] once to maximize the interface throughput. The implemented pipeline runs at 200MHz on the FPGA, with a peak theoretical throughput of 1.6Giga Pixels/s.

4. Experimental Results

We run the 5-3 DWT algorithm on five image components of sizes 0.1, 1.0, 2.0, 4.0 and 8.0 Megapixels respectively. The software implementation uses the OpenJPEG library running on a Core2Extreme X6800 (2.93GHz) CPU. It can be seen that the FPGA implementation performs better (upto 16X) with increasing image sizes, when compared to the software implementation (Fig. 4). Moreover for all but the 0.1 Megapixel data, the FPGA provides speedup.

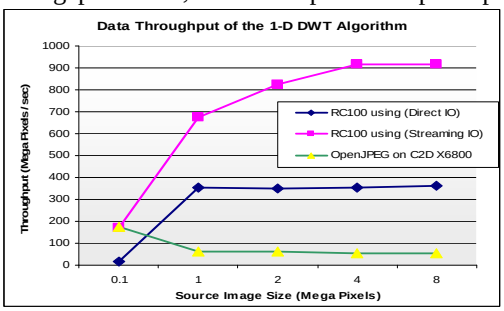


Figure 4: Throughput of 1-D DWT on FPGA and CPU

5. Conclusions

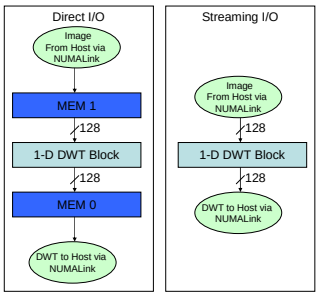
In order to obtain useful speedup from the RC100 Blade, it is necessary to run the algorithm on large data sizes (of the order of megabytes) to effectively amortize the I/O overhead. As a result, more complex algorithms would provide better speedup, since more processing is done per byte of data transferred from the host to the FPGA.

6. References

[1] SGI, "Reconfigurable Application-Specific Computing User's Guide", pp 121-122, pp 161-180.

[2] Majid Rabbani, "The JPEG2000 Still Image Compression Standard", pp77, pp83.

[3] B. A. Buyukkurt, Z. Guo, W. Najjar. "Impact of Loop Unrolling on Throughput, Area and Clock Frequency in ROCCC: C to VHDL Compiler for FPGAs" (ARC 2006) Delft, The Netherlands.



We analyze the performance of the SGI RASC RC100 BLADE and highlight the actual performance attained vis-à-vis 5-3 1D DWT for varying data size, and compare the speedups when compared to an Intel Core 2 Duo Extreme Edition Processor.