Modeling and exploration of a reconfigurable architecture for digital holographic imaging

Lenart, Thomas; Svensson, Henrik; Öwall, Viktor

Published in:
[Host publication title missing]

DOI:
10.1109/ISCAS.2008.4541401

2008

Modeling and Exploration of a Reconfigurable Architecture for Digital Holographic Imaging

Thomas Lenart, Henrik Svensson and Viktor Öwall
CCCD, Department of Electrical and Information Technology, Lund University
Box 118, SE-221 00 Lund, Sweden
Email: {thomas.lenart,henrik.svensson,viktor.owall}@eit.lth.se

***Abstract*— The use of coarse-grain reconfigurable architectures (CGRA) is a suitable alternative for hardware acceleration in many application areas, including digital holographic imaging. In this paper, we propose a CGRA-based system with an array of processing and memory cells, which communicate using a local and a global communication network, and a stream memory controller to manage data transfers to external memory. We present our SystemC-based exploration environment (SCENIC) and methodology used to construct and evaluate systems containing reconfigurable architectures. A case study illustrates the advantages with rapid system level exploration to find and solve bottlenecks in complex designs prior to RTL description.**

I. INTRODUCTION

Digital holography can be used as an alternative to conventional microscopy and has several interesting features, e.g. improved depth focus and possibility to generate three-dimensional and phase contrast images [1]. While a conventional microscope uses lenses, the holographic system uses an image reconstruction algorithm to convert digitally recorded images into visible images. Since both phase and amplitude information of the object is recorded, post-processing algorithms can be used to extract characteristics such as variations in the optical thickness or in the refractive index, which is useful in many biomedical applications [2]. An overview of the holographic setup and image reconstruction is shown in Fig. 1 [3].

An application specific hardware accelerator has previously been developed to speed up the computationally demanding reconstruction algorithms used in digital holographic imaging [4]. The hardware accelerator is tailored for one specific application, resulting in a high speed-up over conventional software-based solutions. However, the reconstruction phase is often followed by custom post-processing steps to extract or present data in intuitive ways, and a hardware accelerator enabling a more flexible hardware mapping is required to support these custom processing steps.

In this paper, a coarse-grain reconfigurable architecture (CGRA) for hardware acceleration in digital holographic imaging is proposed.

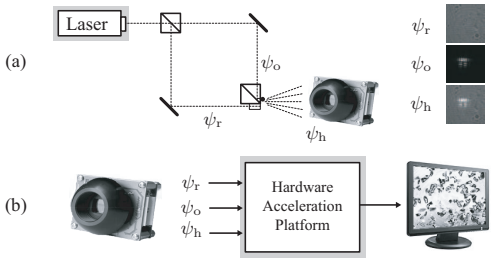

Fig. 1. (a) Holographic setup capturing the light from a laser source on a digital sensor. By blocking the reference or the object beam, totally three images are captured representing the reference ψ_r, the object ψ_o, and the hologram ψ_h. (b) The three captured images are processed by a hardware accelerator to reconstruct visible images.

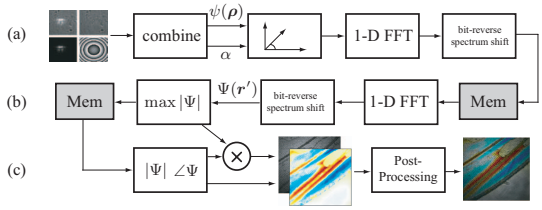

Fig. 2. Dataflow for reconstruction algorithm. Grey boxes indicate memory transfers for data reordering and white boxes indicate computation. (a) Image combine, pixel rotation, and one-dimensional FFT over rows. (b) One-dimensional FFT over columns and store maximum value. (c) Calculate amplitude or phase and scale result into pixels. Post-processing is performed on the resulting images to extract useful material properties.

By using a run-time reconfigurable CGRA, more functionality can be supported at a lower hardware cost compared to mapping the algorithms to application specific hardware accelerators. Hence, sharing the physical resources result in hardware savings, and flexibility is increased by using an array of generic compute and storage elements.

Our design exploration framework SCENIC is used to construct and evaluate systems containing CGRA accelerators. SCENIC enables functionality to map applications to the CGRA and to evaluate performance and other characteristics of the system, as will be demonstrated in a case study in Section V.

This paper is organized as follows: In Section II, our previous work and related design issues are described. The proposed CGRA architecture is presented in Section III together with the exploration environment for constructing generic simulations of CGRA-based systems. In Section V, a case study is presented to demonstrate algorithm mapping, in this case of a post-processing algorithm for digital holographic imaging, to the proposed reconfigurable accelerator. Finally, conclusions are given in Section VI.

II. APPLICATION SPECIFIC ARCHITECTURE

A hardware acceleration platform for digital holographic imaging has been developed in our previous work [4]. XSTREAM accelerates the reconstruction algorithm that transforms the captured images of the object, reference, and hologram, into visible images. The processing data flow of the reconstruction algorithm is shown in Fig. 2(a-c), partitioned into a number of iteration steps. Since the reconstruction algorithm is based on a large size two-dimensional FFT, an intermediate transpose step is required when separating the transformation into one-dimensional FFTs. Due to the large transform size and limited on-chip memory, data must be stored and reordered in main memory and reiterated multiple times over the XSTREAM accelerator for processing. Each iteration requires different functional units to be activated and with different configurations, which means that some functional units are idle during one or more

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

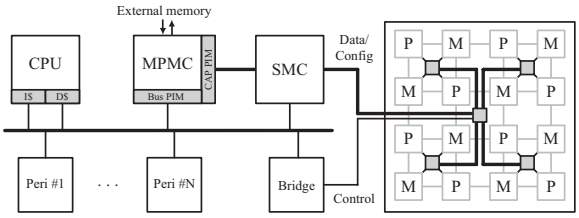

Fig. 3. A system containing an CGRA, which communicate with the multi-port memory controller (MPMC) through a stream memory controller (SMC).

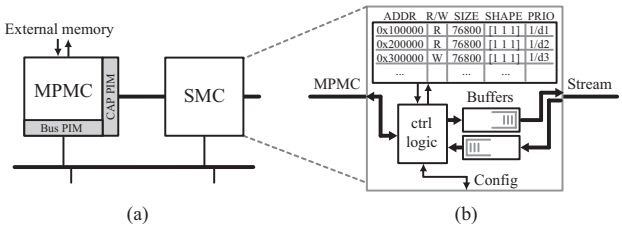

Fig. 5. (a) Stream memory controller (SMC) and multi-port memory controller (MPMC) connected to the bus system and the reconfigurable accelerator. (b) The stream memory controller (SMC) handles data transfers to and from the reconfigurable accelerator. Data is transferred in blocks and the SMC also handles memory addressing and double-buffering.

of the iterations, and the system is hence not fully utilized. Although the functional units in XSTREAM have been optimized for this kind of multi-pass processing [4], a more flexible approach would be to use a CGRA for acceleration.

The possibility to parallelize an algorithm is often limited by the available memory bandwidth, which can create an unavoidable bottleneck even for an application specific solution. The data dependencies in the reconstruction algorithm, especially in the large two-dimensional FFT, requires a high amount of bandwidth for data reordering in external memory. This data dependency can be explored in reconfigurable computing to distribute the computations over multiple configurations executing sequentially. This enables the possibility to reconfigure the accelerator in run-time to execute both the reconstruction algorithm and additional post-processing steps using the same hardware resources. Hence, the system can be extended with future functionality without modifications to the hardware. Exploration to find and solve memory bottlenecks will be further discussed in Section V.

III. PROPOSED RECONFIGURABLE ARCHITECTURE

The proposed CGRA-based system is based on an array of resource cells, which represent the functional units. The resource cells are connected locally to each neighboring cell, and communicate globally using network routers. The proposed system containing a CGRA is shown in Fig. 3, and also includes a processor, a multi-port memory controller (MPMC), and a stream memory controller (SMC) to supply the CGRA with data.

A. Resource cells

Resource cell is the common name for all types of functional modules, and is divided into processing cells and memory cells. The 32-bit processing cell (P) can execute instructions downloaded to an internal program memory (PGM), and supports arithmetic and logical operations. The memory cells (M) are used to separate processing from communication, emulating FIFO functionality as well as supporting random memory access. Memory cells contain a configurable number of banks that each can be configured to forward incoming data to any arbitrary resource cell (global communication), or to supply the neighboring resource cell with data (local communication). The internal building blocks inside processing and memory cells are shown in Fig. 4.

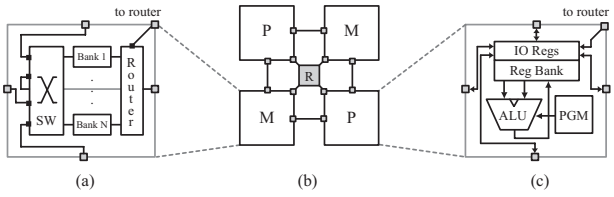

Fig. 4. (a) A memory cell (M) containing a configurable number of memory banks. (b) One tile from the reconfigurable array. (c) Example of a processing cell (P) containing an ALU and a microprogram.

B. Local and global communication

The local interconnects handle communication between neighboring resource cells. It uses a simple handshake protocol to assure safe delivery and self-synchronization of data. The main part of the communication between resource cells is assumed to be in the local interconnects. However, a global network adds routing flexibility and is also connected to external memory. The communication is based on simple routers, which allow only a single path through the global network. The reason is to simplify router implementation and the routing scheme. The buffering in each router is hence minimal to save hardware cost, compared to related work which often use virtual channels which are known to be resource consuming [5]. Routing is based on unique identification numbers (ID) assigned to each resource cell. In this way, data can travel globally over the CGRA and to external interfaces.

C. External memory interface

Memory cells can store data during computations, but data to be processed is usually located in caches or external memory. In this work, stream memory controllers (SMC) are proposed for transferring data efficiently between main memory and the resource cells. A SMC can be connected to any router in the network and is uniquely identified in the same way as resource cells. The SMC contains a stream table that handle stream descriptors, which provide information about how to transfer data between the resource cell and the external memory, as shown in Fig. 5. The SMC can also operate as an intermediate storage space or double-buffer between two processing elements to allow emulation of large-size buffers.

The SMC is connected to the MPMC, which controls the external memory. The MPMC can be configured to emulate different memories by changing operating frequency, wordlength, data-rate, and internal parameters for controlling the memory timing. This is useful when exploring systems to find optimal memory configuration.

IV. MODELING AND EXPLORATION USING SCENIC

This section presents our SystemC environment with interactive control (SCENIC), which extends the open-source SystemC simulator [6] with support for constructing, controlling, and observing simulation models interactively. SCENIC is used to create, simulate, and explore the proposed CGRA system in Section III. SystemC has been used in many academic and industrial research projects, mainly for exploring complex architectures such as Network-on-Chip (NoC) [7] or multi-processor systems [8]. The advantages with SystemC include

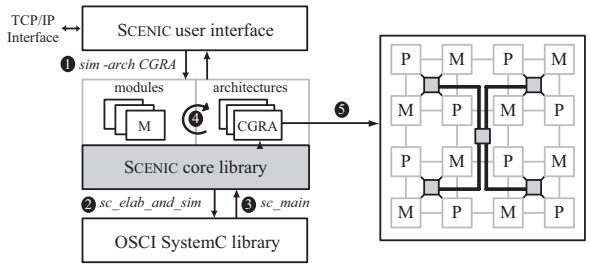

Fig. 6. (a) The user interface communicate with the SCENIC core library. A simulation is initiated from the user interface, which triggers the SystemC library to initialize. When ready, SCENIC calls the requested *architectural template* which construct a simulation from user-defined *modules*. (b) The SystemC simulation constructed with SCENIC has run-time control and observability from the user interface.

modeling at different abstraction levels, simplified hardware/software co-simulation, and high simulation speed.

Simulations in SCENIC are constructed using *architectural templates*, which describe the design architecture, and *modules* that describe functionality. Modules are user-defined SystemC simulation objects that inherit controllability and observability from a SCENIC. Modules can register member variables to be visible, modified, or logged from the SCENIC user interface, or to make callbacks when the user requests more complex information from the module. Architectural templates are executable code which takes user parameters to dynamically build a simulation, resulting in flexible and powerful system description. The instantiated modules, registered member variables, and the design hierarchy are visible from the user interface. The internal flow for constructing a simulation is shown in Fig. 6.

SCENIC handle user interaction from a command line interface, script files, or to a graphical user interface using a TCP/IP socket (Matlab support). SCENIC supports user commands for starting and stopping the simulation, setting environment variables, and evaluating basic arithmetic or logical operations. If the hierarchical name of a SystemC simulation module is given as a command, the request is forwarded to the module which can evaluate custom user functionality. In this way, SCENIC enables fast exploration to setup different scenarios in order to verify and evaluate system architectures.

All functional units in Fig. 3 have been implemented as modules in SCENIC, and an architectural template has been created to dynamically build CGRA simulations based on these modules. The architectural template contains the following parameterizable functionality:

1) Tile generator: The tile generator use a tile description file to create an array of resource cells. When constructing larger arrays, the tile is used as the basic building block, as shown in Fig. 7(a). The resource cells are configured with the functional unit specified in the tile description file, but can also be configured during simulation to support partial dynamic reconfiguration.

2) Topology generator: The topology generator creates the local interconnects between resource cells, and supports *mesh*, *torus*, *ring*, and user-defined topologies. Fig. 7(b) shows the resource cells connected with a mesh topology.

3) Network generator: The network generator inserts the global network which connects to the resource cells, as shown in Fig. 7(c). Stream memory controllers are connected directly to the network routers. The network generator also updates the routing tables to allow global communication.

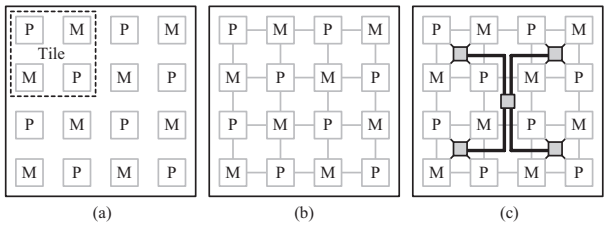

Fig. 7. (a) Constructing the array from a tile description. (b) Building local communication topology, which can be mesh, ring, torus or custom topology. (c) Building global network with routers (in grey).

V. CASE STUDY

The importance of system level exploration will be illustrated in this case study, which presents the implementation of a post-processing algorithm for visual enhancement by combining amplitude and phase images (CAP). A run-time reconfigurable platform enables the possibility to change algorithm mapping, ranging from sequential execution using only a single processing cell to partly or fully parallelized execution. Selected mapping depends on the algorithm and available hardware resources in the CGRA, but also on system parameters such as available bandwidth to external memory.

SCENIC is used in this case study to evaluate system performance in terms of data throughput and to find potential bottlenecks when mapping algorithms to the CGRA. The SCENIC scripting environment is used to configure modules to define a scenario, reconfigure modules during simulation (controllability), and to extract performance data during and after simulation (observability) to evaluate the system. Two architectural mappings will be evaluated, totally consisting of five scenarios and 50 test cases. The first mapping is when the algorithm execute on a single processing element, and the second is when dividing the CAP data processing onto two processing elements, as shown in Fig. 8. This illustrates that throughput is not only a matter of simple parallelism, but also a balancing of system resources to achieve maximum performance.

The following SCENIC script downloads two bitmap images, shown in Fig. 9(a-b), through the MPMC and inserts stream descriptors in the SMC to transfer data to and from the processing cell:

```
%% create and configure 4x4 CGRA
sim -arch CGRA:ra0 -tile 2x2p2m2.tile -width 2 -height 2
    -topology mesh -router_fanout 4 -insert_top_router 1
sim -arch BusSystem   %% bus-system with SMC and MPMC

%% download bitmap images to memory
MPMC wr -addr 0x100000 -file "amplitude.bmp"
MPMC wr -addr 0x200000 -file "phase.bmp"

%% config processor and setup links to smc
reconfig_cell(ra0.rc1x1, "cap.asm")
reconfig_route(ra0.rc1x1, SMC)

%% insert stream descriptors
SMC insert -addr 0x100000 -rnw 1 -size $IMAGE_SIZE
SMC insert -addr 0x200000 -rnw 1 -size $IMAGE_SIZE
SMC insert -addr 0x300000 -rnw 0 -size $IMAGE_SIZE

runb 4 ms             %% run blocking simulation

%% extract performance data from modules
MPMC rd -addr 0x300000 -size $IMAGE_SIZE -file "cap.bmp"
set TRANSFERS   [ra0.rc*x* get -var transfers];
set SIM_TIME    [eval [SMC get -var last_transfer] @ 0];
set UTILIZATION [eval [eval sum $TRANSFERS] / $SIM_TIME]
```

The program memory in the processing cell is configured with a three-instruction assembly program to multiply the RGB-colors in

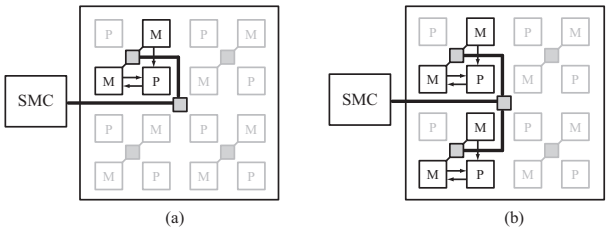

Fig. 8. (a) Mapping of CAP to a single processing element. (b) Dividing the computations onto two processing elements.

the input streams byte-wise, and write the result to the output port. Hence, four clock cycles are required to process one pixel. The result after processing is shown in Fig. 9(c).

The script is executed in batch mode with different transfer lengths, which is the size in words of a transaction between the memory and the processing cell. The transfer length is important for a realistic case study, since external memory is burst oriented with high initial latency to access the first word in a transfer. Consequently, the simulation in Fig. 10 plot (a) shows improved throughput when the transfer length increases.

To show the advantage with a flexible and reconfigurable architecture, the SCENIC script is modified to map onto two parallel processing cells, sharing the computational workload to increase throughput. However, the simulation in Fig. 10 plot (b) shows reduced throughput for the second mapping when the transfer length increases. With further analysis by inspecting the MPMC using SCENIC, it can be seen in Fig. 11 that the memory access pattern change abruptly, caused by interleaving of the result streams from the two processing cells. Interleaving results in shorter burst transfers to external memory and decreased throughput. However, the interleaving problem can be solved by reordering the received data in the SMC.

The result after reordering is shown in Fig. 10 plot (c). Reordering solves the burst length issue, but the throughput has still not increased as expected. Since the utilization is only 50% in each processing element, more memory bandwidth is required to supply the processing elements with data. To address the bandwidth issue, the MPMC is reconfigured to evaluate two new scenarios, first evaluating a double data-rate (DDR) memory and then increasing the memory wordlength to 64 bits. The simulations are shown in Fig. 10 plot (d-e), presenting dramatically increased data throughput.

This case study has illustrated how SCENIC can be used to guide the designer when tuning the system for optimal performance. Our current work involves mapping both reconstruction algorithm and custom algorithms to our CGRA, where each algorithm mapping needs to be optimized as shown above. Automatic algorithm mapping and structural VHDL generation is also part of our ongoing work.

VI. CONCLUSION

This paper has presented a coarse-grain reconfigurable architecture intended to accelerate algorithms in digital holographic imaging. The CGRA is an array of processing and memory cells, which communicate using a local and a global communication network, and a stream memory controller to manage data transfers to external memory. The system is modeled and explored using our presented SystemC-based environment SCENIC, used to evaluate the performance and to identify bottlenecks in the system. A case study demonstrates the mapping flexibility of our CGRA and how to tune the system for optimal throughput using SCENIC.

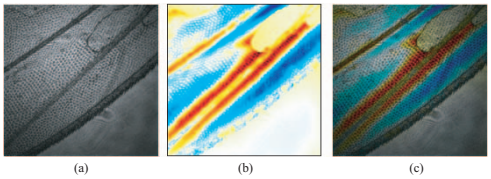

Fig. 9. (a) The reconstructed amplitude image showing the wing of a fruit fly. (b) The reconstructed and unwrapped phase image. (c) The combined amplitude and phase image.

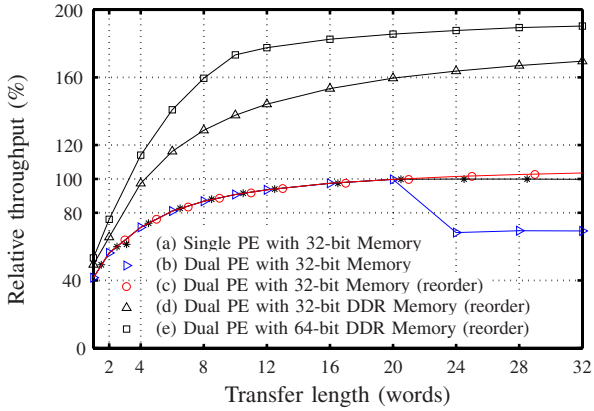

Fig. 10. (plots a-e) Throughput relative to execution on single processing cell. Simulation shows mapping to single and dual PEs for different memory configurations. Throughput is plotted against transfer length (burst length).

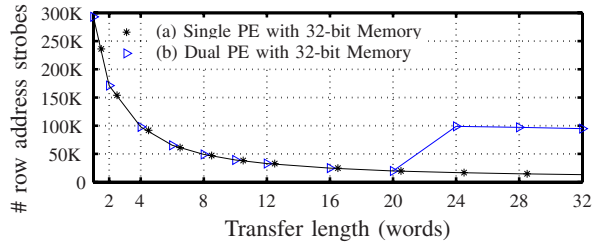

Fig. 11. (plots a and b) Number of row address strobes to external memory. A high number of row address strobes indicate an irregular access pattern, which has negative impact on throughput.

REFERENCES

[1] U. Schnars and W. Jueptner, *Digital Holography*. Springer-Verlag, Berlin, Heidelberg: Springer, 2005.

[2] Myung K. Kim and Lingfeng Yu and Christopher J. Mann, "Interference Techniques in Digital Holography," *Journal of Optics A: Pure and Applied Optics*, no. 7, pp. 518–523, July 2006.

[3] M. Gustafsson *et al.*, "High resolution digital transmission microscopy - a Fourier holography approach," *Optics and Lasers in Engineering*, vol. 41(3), pp. 553–563, Mar. 2004.

[4] T. Lenart *et al.*, "Accelerating signal processing algorithms in digital holography using an FPGA platform," in *Proc. of ICFPT*, Tokyo, Japan, Dec. 15-17 2003, pp. 387–390.

[5] J. Chan and S. Parameswaran, "NoCGen: A Template Based Reuse Methodology for Networks on Chip Architectures," in *Proc. of the 17th International Conference on VLSI Design*, 2004, pp. 717–720.

[6] Open SystemC Initiative (OSCI), "OSCI SystemC 2.2 Open-source Library," http://www.systemc.org.

[7] A. Portero, R. Pla, and J. Carrabina, "SystemC Implementation of a NoC," in *Proc. of IEEE International Conference on Industrial Technology*, 2005, pp. 1132–1135.

[8] L. Yu, S. Abdi, and D. D. Gajski, "Transaction level platform modeling in systemc for multi-processor designs," UC Irvine, Tech. Rep., Jan. 2007. [Online]. Available: http://www.gigascale.org/pubs/988.html