CODACS Project: A Demand-Data Driven Reconfigurable Architecture

Lorenzo Verdoscia

Research Center on Parallel Computing and Supercomputers – CNR, Via Castellino, 111 80131 Napoli, Italy,
verdoscia.l@cps.na.cnr.it

Abstract. This paper presents CODACS (COnfigurable DAtaflow Computing System) architecture, a high performance reconfigurable computing system prototype with a highly scalable degree able to directly execute in hardware dataflow processes (dataflow graphs). The reconfigurable environment consists of a set of FPGA based *platform-processors* created by a set of identical Multi Purpose Functional Units (MPFUs) and a reconfigurable interconnect to allow a straightforward one-to-one mapping between dataflow actors and MPFUs. Since CODACS does not support the conventional processor cycle, the platform-processor computation is completely asynchronous according to the dataflow graph execution paradigm proposed in [8].

1 Introduction

The advent of in-circuit (re)programmable FPGAs(Field Programmable Gate Arrays) has enabled a new form of computing prompted by the (re)configurable computing paradigm [1] where a function calculation is computed configuring and interconnecting a number of cells. In particular, given its fine grain nature, the dataflow execution model is promising when applied to this platform because, as (re)configurable computing, it computes a dataflow graph configuring and interconnecting actors instead of carrying out in sequence a set of operations as a standard processor does.

However, despite the general scepticism regarding dataflow architectures due to its disappointing results, we believe they are still a valid proposal to increase performance and seriously attack, at least at a chip level, the von Neumann model.

There are at least four reasons that have motivated this proposal and consequently driven the design process [2]: the first is the demand to directly map in hardware dataflow graphs in dataflow mode; the second is the need to dispose of a straightforward data flow control and actor firing mechanism at a minimal hardware cost; the third is the requirement to reduce the continual LOAD and STORE operations and augment performance; the last is the possibility to adopt primitive functions of a functional language as assembler language for a processor.

2 CODACS Architecture

CODACS general architecture, shown in Fig. 1, is constituted by a set of identical nodes connected in a WK-Recursive topology [3] where each node is constituted by a *Smart Router Subsystem* (SRS), devoted to provide some kernel functions and all communication and routing activities, and a *Platform-Processor Subsystem* (PPS), devoted to execute dataflow graphs.

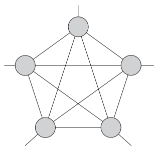

Fig. 1. CODACS Architecture connected as WK-recursive with $N_d = 5$ and $Level = 1$

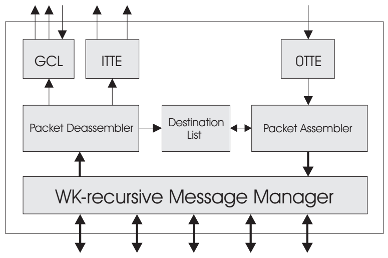

Fig. 2. Smart Router Subsystem Architecture

Smart Router Subsystem. When a message reaches a node, the SRS (Fig. 2) evaluates it. If that node is not the destination, the WK-recursive Message Manager (WKMM) routes the message through the appropriate output link according to the routing strategy described in [4]. If it is the destination node, the WKMM transfers the message to the Packet Disassembler (PD) for the processing. The PD unpacks it, evaluates its content, and transfers information to a) the Graph Configuration List (GCL), that contains the graph configuration table list assigned to the platform-processor; b) the Destination List (DL), that contains the list of the result destination node set (one set for each configuration); c) the Input Transfer Token Environment (ITTE), to transfer data tokens to the PPE.

In the ITTE, data token storage occurs in separate but associate buffers to transfer right and left tokens to MPFUs. When results coming from the PPE are ready inside the Output Token Transfer Environment (OTTE), the Packet Assembler (PA) scans the destination node list, associates nodes to results, prepares new messages (one for each receiver), and transfers them to the WKMM for delivering.

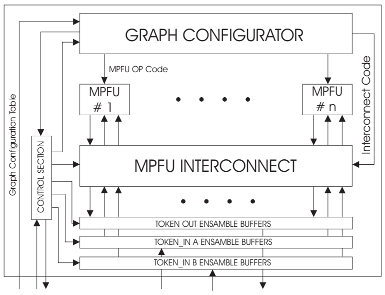
GRAPH CONFIGURATOR
MPFU OP Code
MPFU # 1
MPFU # n
Interconnect Code
Graph Configuration Table
CONTROL SECTION
MPFU INTERCONNECT
TOKEN OUT ENSAMBLE BUFFERS
TOKEN_IN A ENSAMBLE BUFFERS
TOKEN_IN B ENSAMBLE BUFFERS

Fig. 3. Platform-Processor Subsystem Architecture

Platform-Processor Subsystem. This subsystem (Fig. 3) executes the dataflow graph assigned to that node. After receiving the graph configuration table from GCL, the Graph Configurator (GC) executes two operations: it sets the MPFU interconnect and assigns the operation code to each MPFU, thus carrying out the one-to-one correspondence (mapping) between graph nodes and computing units. Once the configuration phase terminates, it sends a signal to the control that enables the two Token_In buffers to start the computation. When a graph computation ends, results are stored in the Token_Out buffer and then transferred to the OTTE. If the same graph must process different input tokens (e.g. matrix inner product), the GC only checks for the input token availability. We point out that, thanks to the I/O Ensemble Buffers and Token Transfer Environments, the platform-processor and smart router environment become local to each subsystem allowing to overlap data load, message transfer, and computation activities.

3 Performance

To implement the proposed architecture, we have used ALTERA Quartus II [5] development software and a PCI board with 5 APEX20K15-C FPGA components. As a result [2], we obtained a platform-processor with 105 interconnected MPFUs that execute operations on 32 bit integer operands at time $t_{MPFU} = 1$ μsec while the measured transfer time from/to the SRS $t_b = 8$ nsec. To evaluate CODACS, Jacobi and Gauss-Seidel iterative algorithms have been used because they face the same problem in a parallel and sequential mode. Table 1 shows the two performance indices CP (communication penalty) and Sp (speedup) for some values of n (number of equations). Due to fine grain dataflow operations, Jacobi performs better than Gauss-Seidel. However, most of the time is spent in communication.

4 Concluding Remarks

Principal features that distinguish this machine from similar ones ([6], [7]) are: the platform-processor executes dataflow graphs, including loops, without con-

Table 1. Performance

	Gauss-Seidel		Jacobi	
n	CP	Sp	CP	Sp
96	6.94	0.90	7.92	5.70
320	5.38	3.03	6.15	9.86
992	4.66	9.85	5.78	17.42

trol tokens but using only actors with homogeneous I/O conditions; MPFU assembly language is a high level programming language; graph execution and communication environments are separated to overlap data transfer and computation; no memory usage is required during a graph execution to reduce latency penalty; finally, it is characterized by a highly scalable general architecture. At the moment we are realizing a prototype employing 5 ALTERA APEX20K15-3C components.

Acknowledgments

This work was supported in part by Regione Campania under contract (POP-FESR Azione 5.4.2) “Sviluppo di Metodologie e Tecniche per la Progettazione e la Realizzazione dei Sistemi Avanzati di Supporto alle Decisioni” and by CNR-Agenzia2000 Program under Grant CNRC0014B3_008.

References

1. Gray, J.P., Kean, T.A.: Configurable Hardware: A New Paradigm for Computation. In *Proc. Decennial CalTech Conf. VLSI*, pages 277–293, Pasadena, CA, March 1989.
2. Verdoscia, L., Licciardo, G.: CODACS Project: The General Architecture and its Motivation. Technical report, CNR Research Center on Parallel Computing and Supercomputers, Via Castellino, 111 - 80131 Napoli - Italy, January 2002.
3. Chen, G.H., Du, D.R.: Topological Properties, Communication, and Computing on WK-Recursive Networks. *Networks*, 24:303–317, 1994.
4. Verdoscia, L., Vaccaro, R.: An Adaptive Routing Algorithm for WK-Recursive Topologies. *Computing*, 63(2):171–184, 1999.
5. ALTERA Corporation.: Quartus programmable logic development system and software. San Jose, CA, May 1999.
6. Singh, H., alii.: Morphosys: An Integrated Reconfigurable System for Data-Parallel and Computation Intensive Applications. *IEEE Trans. Computers*, 49(5):465–480, May 2000.
7. Murakawa, M., alii.: The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing. *IEEE Trans. on Computers*, 48(6):628–639, June 1999.
8. Verdoscia, L., Vaccaro, R.: A High-Level Dataflow System. *Computing*, 60(4):285–305, 1998.