The MANGO Process for Designing and Programming Multi-Accelerator Multi-FPGA Systems

Rafael Tornero, José Flich, José María Martínez, Tomás Picornell
Universitat Politècnica de València
Valencia, Spain
rtornero@disca.upv.es,jflich@disca.upv.es
jomarm10@gap.upv.es,tompic@gap.upv.es

Vincenzo Scotti
Università degli Studi di Napoli Federico II
Naples, Italy
vincen.scotti@studenti.unina.it

ABSTRACT

This paper describes the approach followed in the European FETHPC MANGO project to design and program systems made of multiple FPGAs interconnected. The MANGO approach relies on the instantiation and management of multiple generic and custom-made accelerators which can be programmed to communicate each other via non-coherent shared memory and through synchronization registers. The paper introduces the low level architecture including the multi-FPGA interconnect deployed, the communication protocol and the architectural template-based approach to simplify the design process.

ACM Reference Format:
Rafael Tornero, José Flich, José María Martínez, Tomás Picornell and Vincenzo Scotti. 2018. The MANGO Process for Designing and Programming Multi-Accelerator Multi-FPGA Systems. In *Proceedings of Fourth International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC'18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

High-Performance Computing (HPC) is slowly but steadily introducing FPGA-based devices as energy efficient components helping in improving the performance-per-watt relationship [1, 4]. FPGAs are power efficient for specific applications which suit best on reconfigurable devices with massive number of computing units which can be used in parallel. As HPC is also being adopted to sustain and improve BigData applications, specially those related to Artificial Intelligence and Deep Learning, FPGA devices get more interest as fine-grained parallelism is exploited more efficiently.

However, the use of FPGAs for HPC purposes brings its own limitations and complexities. One major issue is scaling beyond a single node or FPGA which is still not a mainstream capability, except for few examples such as the Catapult system [4]. Another well-known complexity is programmability. As we approach commodity in the use of FPGAs there is need to develop new methods and tools that help reducing complexity in handling and programming those devices. Enabling multi-FPGA systems for implementing large designs is also a challenge that needs to be overcome.

The MANGO project [2] is a European project which has as one of its main objectives the development of complex multi-FPGA systems for HPC architecture exploration purposes. In the context of MANGO, FPGAs from different models and vendors can be plugged together and interconnected in a pin-to-pin fashion. Memory and IO modules are also plugged on top of FPGAs in order to create a complete system. The final system configuration is decided by the system engineer and depends on physical constraints but mainly on application requirements. Given the large flexibility offered by the underlying hardware produced in MANGO, there is a need to provide a flexible and simplified methodology for proper design and implementation of the complete system, enabling the instantiation of multiple accelerator devices within the FPGAs adapted to the specific algorithms that need to be run on the FPGAs.

In this paper we describe the methodology followed in the MANGO project for the simplification process of design, test and management of any generic cluster configuration made of multiple FPGA devices.

2 MANGO ARCHITECTURE

Figure 1 shows the general view of the MANGO approach. The final MANGO prototype will be made of 16 clusters, each with two different components: an standard HPC server, and the Heterogeneous Node (HN) cluster implemented on a set of interconnected FPGAs, memory and IO modules. For the sake of simplification we focus on the description of a single HN cluster.

The HN cluster can be configured by the system engineer by putting together a set of FPGAs, potentially from different vendors and models, using the deployed composable infrastructure provided in MANGO by ProDesign Company [3]. Motherboards are provided to plug FPGA modules from different vendors and models. Those FPGAs can be connected pin-to-pin using specific cables. Large systems up to 20 FPGAs can be deployed guaranteeing a synchronous phased-aligned clock is achieved. Memory modules (DDR3, DDR4, ...) and IO modules (PCIe, Ethernet, ...) can be plugged on top of FPGAs, building a complete system. Physical construction of the cluster does not require any specific expertise and resembles a lego-like exercise.

The physical construction of the cluster is followed by the definition of a set of accelerators (complex IP blocks) to be instantiated

H2RC'18, Nov 2018, Dallas, Texas USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

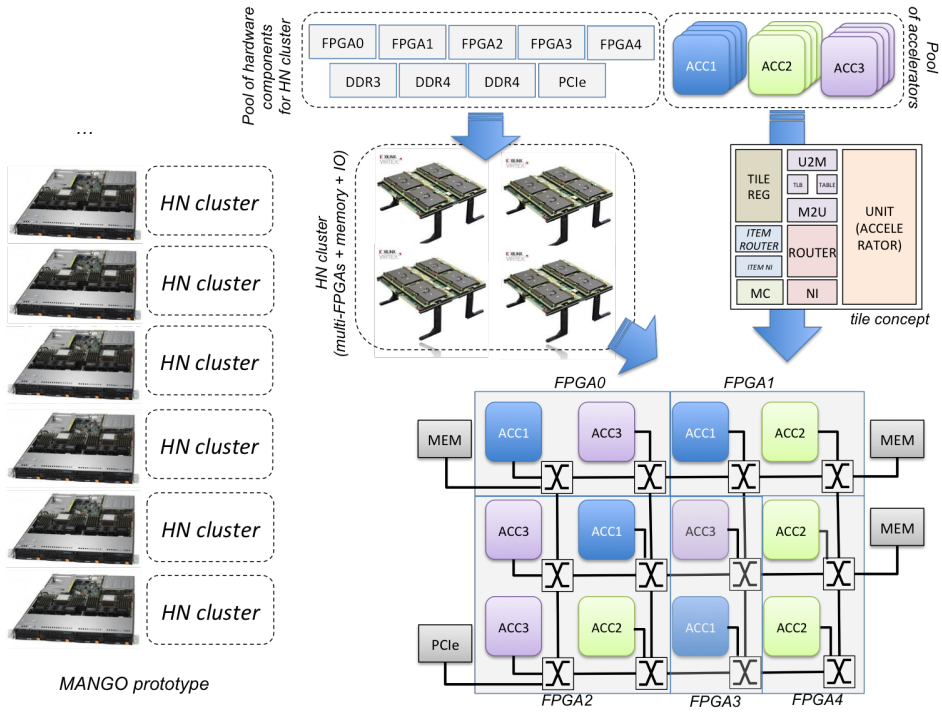

Figure 1: Global view of the multi-FPGA architecture design.

and deployed all over the FPGA cluster. Accelerators will be of different types and functionality, custom-made or generic ones, being able to adapt to the specificities of the target applications on the HPC infrastructure. Within MANGO different generic accelerators (generic many-core system and generic GPU-like accelerator) and custom-made (coarse grained reconfigurable accelerator) have been deployed. All of them follow the same defined interface.

The MANGO project aims at simplifying the development of a tool for rapid architecture exploration, enabling the system engineer to practice and experiment with different suitable configurations. This, in terms of functionality, requires the following properties that must be guaranteed by the designed architecture: *flexibility* in the configuration of the system, effective *communication* with the deployed system, and *monitoring* of the deployed system.

Flexibility in the configuration means that the engineer should be able to rapidly and conveniently change the system configuration with the following options as example: Number and type of accelerators to be instantiated, Number of DDR memory modules to be instantiated, or mapping of accelerators and memory modules on FPGAs. The system engineer describes also the physical configuration of the cluster (which FPGAs are attached to which ones, where memory modules and IO modules are attached to, ...).

Flexibility must be achieved at design time and with a proper method that by performing the smallest changes possible to the source code being able to achieve radical changes in the configuration. In this sense, we refer to an architecture ID to each possible configuration that the engineer can conceive. Therefore, the system must provide a simplified way to allow the engineer to change and define an architecture ID.

Effective communication means that the accelerators within the multi-FPGA system must properly communicate between them and to/from memory modules and to/from the server. In addition, the system must be provisioned with mechanisms to guarantee QoS requirements from applications running their kernels in the accelerators. All this reduces to two basic components of the many-core architecture. On one hand, the design of a network connecting accelerators and memories, and on the other hand, the design of a proper communication infrastructure between the server and the manycore.

Finally, in terms of monitoring, the system must provision proper and effective methods to let the monitoring infrastructure to get access to critical data from the system. This will be provided by an additional network.

2.1 Heterogeneity Support

The previous requirements are the driving goals set for the design of the MANGO architecture. Adding different accelerators of different types to a cluster of connected FPGAs makes the system highly heterogeneous. This leads to additional complexity that needs to be handled. In order to simplify such task, we have opted for a tile-based design where accelerators and memory modules are mapped to. In this approach, the HN cluster reverts to an homogeneous set of tiles laid out in a 2D mesh configuration. Each tile will offer the same functionality for communication but within each tile a different accelerator and memory configuration may be placed. This means heterogeneity is exercised at tile level. Figure 1 shows the tile concept. Every accelerator is embedded into a tile. An homogeneous

grid of tiles is span over the grid of physical FPGAs deployed within the cluster. The number of tiles mapped to each FPGA could even be different, but taking into account that the 2D mesh configuration has to be kept.

2.2 Network Architecture

The MANGO network is implemented as a set of modules embedded and distributed over the tiles. The MANGO network is made of two major components, the ROUTER module and the NI module, both present on every tile. The ROUTER provides connectivity between modules located at neighbor tiles, while the NI connects tile components to the ROUTER in the local tile. The network is made of ROUTERs interconnected, one on each tile, and NI modules connected to one local ROUTER module. The final layout (topology) of the network is a 2D-mesh network. One important aspect of the network is the different link bandwidth attainable between routers. The pin bandwidth between FPGAs will bound the router's communication bandwidth. In order to minimize the impact on communication bandwidth, we define different phit sizes between connected routers, depending on their location (either on the same FPGA or at different FPGA modules). With this, intra-FPGA communication is not affected by the inter-FPGA communication limited bandwidth.

VNs provide traffic isolation and bandwidth reservation opportunities. Indeed, the router bandwidth can be reserved among different traffic streams, one on each VN, and so among different memory transfers. This is a key factor to allow the system to meet QoS requirements regarding memory access bandwidth, as potentially requested by applications. In order to do that, VNs can be dynamically partitioned with respect to the bandwidth available on the link. In addition, packets traveling in the network through routers have a label in their headers, enabling the router to establish priorities for resource assignment, thereby enabling timing guarantees from the application perspective.

Related with the communication between the server and the HN cluster, currently, every architecture supports communication by Ethernet or PCIe, by means of ProDesign's MMI64 proprietary communication protocol. A set of communication channels can be defined, which, in liaison with VNs can offer independent guaranteed bandwidth communication flows.

2.3 Accelerator Interface

In order to provide compatibility and scalability, the MANGO project has defined a clear and unique set of interfaces to connect any type of accelerator within the MANGO architecture. Two interfaces (data and control interfaces) are mandatory to be used by each accelerator while the third one (synchronization interface) is optional. The data interface is related with the ability of the accelerator to access memory and MANGO registers. A TLB strategy is implemented on each accelerator, which enables mapping virtual addresses of the accelerator on physical addresses. A flat physical memory address space is supported by the concatenation of all DDR memories implemented on the HN cluster.

The control interface enables configuration and control of the accelerators. However, accelerators will have different configurability and behavior. Thus, the control interface must be generic enough and flexible to let accelerators be properly configured based on their complexities. With this goal in mind, the control interface is a minimal one which lets the accelerators define their protocols and specificities.

With the previous two interfaces the accelerators can easily synchronize with other accelerators or with the main application running on the server side. Accelerators can synchronize by reading/writing on register-mapped memory locations.

The optional interface allows accelerators to use a more direct and efficient generic interrupt-based synchronization method. The interrupt system developed in MANGO for synchronization between host applications and accelerators consists of a 16-bit interrupt vector that can be masked in order to ignore the interrupts the accelerator/host application is not interested in. The method is generic in the sense that every accelerator can define their own specific interrupts, so they can be commanded from the host to the accelerator and vice versa.

2.4 Architecture Template Definition

The basic building blocks of the architecture (both hardware and IP blocks) can be compounded in many different ways and configurations allowing a deep exploration of heterogeneous architectures, which is the goal of the MANGO project. The designer of a MANGO architecture must deal with all the complexities of the MANGO components in a structured and simplified way in order to make the design process smooth and simple. Indeed, one background and significant effort performed within the project is the design of a system platform able to be easily configured and adapted to multiple and different configurations of accelerators within the MANGO architecture. In this section we describe such effort performed within MANGO.

The MANGO platform allows the designer to easily define, configure, and implement a supported architecture or a new one. An architecture configuration is identified by a unique ID and each architecture defines, among others, the following parameters: I) number of FPGAs to use in the cluster, II) type of each FPGA to use in the cluster, III) connectivity between FPGAs within the cluster, IV) number of tiles on each FPGA defined as the number of rows and columns of tiles, V) number and exact location of memory modules in the cluster, VI) number and exact location of I/O devices connections in the cluster, and VII) types of accelerators (and versions of accelerators) on each tile defined in the cluster.

The complete MANGO architecture has been coded in Verilog Hardware Description Language (VHDL) with the aim of addressing every detail of the highly parametrized tile-based template developed in MANGO. Nevertheless, accelerators can be designed using different methodologies, e.g. High Level Synthesis (HLS), given that they absolutely accomplish with the MANGO accelerator interface defined above. The Vivado Design Suite [6] has been used for behavioral simulation, synthesis and implementation purposes. For a proper management and understanding of the code implementing the MANGO architecture, a set of hierarchical template modules have been enforced. Each accelerator (or even new ones) can be easily added to the project. The tile based design is exploited in the module hierarchy. A tile includes all the components shown previously.

Section	Description
FPGA description	Gets the FPGA models
FPGA identification	Provides the IDs given to every FPGA in the architecture
Multi FPGA	Contains all the common parameters required
Clock	Allow us to define the clocks used for some architecture subsystems
Topology	Contains the description of the MANGO network topology
Memory	Collects information about memory technologies and how they are connected to
Input/Output	Collects the information on the input/output system devices as how to connect to
Network	Describes the type of the data, control and debug network
Constraint	Allow us to specify a set of constraints for the architecture

Table 1: Sections on architecture template file.

The architecture enables the definition of multi-accelerators platforms where each accelerator can be of different type or the same type but with different configurations (e.g. with different number of cores or cache capacities). To enable such reconfigurability and flexibility, the template modules has been highly parametrized. The use of parameters makes the same accelerator module can be instantiated several times (one on each tile) with same or different configuration parameters, leading to different versions of the same code but coexisting at the same time on the same MANGO architecture instance.

The MANGO architecture is defined completely from a single ASCII template file. The engineer can deploy a brand new architecture by only modifying such file. Every architecture is composed of a set of sections that groups related architecture definition parameters. Some sections are common to all the architectures, but also there exist other sections that are architecture dependent. For the latter case, a macro or define keeps control on enabling or disabling that section for the architecture. Table 1 describes the main sections that build an architecture.

2.5 Monitoring, Control and Debug Support

In order to simplify the test and validation process, the MANGO approach supports an additional network connecting tiles which handles control, monitoring, and debug information from the accelerators. This network is a lightweight implementation of the main data network with only one VN and with a shorter flit size (performance is not a requirement). A complete software infrastructure has been deployed to handle all the generated control traffic on the server node.

Control and debug information is accelerator specific. A 32-bit control port is defined for each unit (accelerator) to send control and debug information to the server node. All this traffic is tunneled through the control network and handled by the daemon process implemented on the server node. A new accelerator will need to define its own control commands and debug format and enhance the daemon for proper support.

2.6 FPGA Resource Utilization

Table 2 shows the FPGA resource utilization of the MANGO tile when there is not any accelerator implemented inside it and when there is a generic MIPS-based cache coherent 2-core accelerator implemented in. The results have been obtained for the Virtex 7 V2000T speedgrade -1 Xilinx FPGA [5]. The resource utilization

Resource	Tile (without accelerator)	Tile (with accelerator)
LUT	3.29	10.72
LUTRAM	2.47	4.94
FF	1.34	5.23

Table 2: FPGA resource percentage utilization of a tile.

shown for a tile without any accelerator implemented in it provides us the overhead that introduces the MANGO infrastructure with regards to the complete system. As it can be seen, it is kept below 5% for every type of FPGA resource.

3 CONCLUSIONS

In this paper we have presented the MANGO approach to support the implementation of multiple accelerators on a multi-FPGA system. The approach let's the system engineer to customize a complete cluster by mixing FPGAs of different vendors and models, together with DDR memories and IO modules, and then exposing the configuration of accelerators (IP cores) throughout the multi-FPGA configuration. The paper has hinted also the interconnection infrastructure and the communication protocol with support for multiple flows of communication and the support of bandwidth reservation policies.

ACKNOWLEDGMENTS

This work is supported by the European Commission through MANGO project, under the Horizon 2020 FET-HPC program, grant number 671668.

REFERENCES

[1] Texas Advanced Computing Center. [n. d.]. Retrieved August 15, 2018 from https://www.tacc.utexas.edu/systems/chameleon
[2] MANGO Consortium. [n. d.]. Retrieved August 15, 2018 from https://www.mango-project.eu
[3] Prodesign Gmbh. [n. d.]. Retrieved August 15, 2018 from https://www.prodesign-europe.com
[4] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James R. Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2016. A reconfigurable fabric for accelerating large-scale datacenter services. *Commun. ACM* 59, 11 (2016), 114–122. https://doi.org/10.1145/2996868
[5] Xilinx. [n. d.]. Virtex 7 Series. Retrieved August 15, 2018 from https://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html#productTable
[6] Xilinx. [n. d.]. Vivado Design Suite. Retrieved August 15, 2018 from https://www.xilinx.com/products/design-tools/vivado.html