Network on Chip Architecture for Multi-agent Systems in FPGA

EDUARDO A. GERLEIN, Departamento de Electrónica, Pontificia Universidad Javeriana;

T.M. MCGINNITY, Intelligent Systems Research Centre, University of Ulster; School of Science and Technology, Nottingham Trent University

AMAR BELATRECHE, Faculty of Engineering and Environment, Northumbria University

SONYA COLEMAN, Intelligent Systems Research Centre, Ulster University

A system of interacting agents is, by definition, very demanding in terms of computational resources. Although multi-agent systems have been used to solve complex problems in many areas, it is usually very difficult to perform large-scale simulations in their targeted serial computing platforms. Reconfigurable hardware, in particular Field Programmable Gate Arrays (FPGA) devices, have been successfully used in High Performance Computing applications due to their inherent flexibility, data parallelism and algorithm acceleration capabilities. Indeed, reconfigurable hardware seems to be the next logical step in the agency paradigm, but only a few attempts have been successful in implementing multi-agent systems in these platforms. This paper discusses the problem of inter-agent communications in Field Programmable Gate Arrays. It proposes a Network-on-Chip in a hierarchical star topology to enable agents' transactions through message broadcasting using the Open Core Protocol, as an interface between hardware modules. A customizable router microarchitecture is described and a multi-agent system is created to simulate and analyse message exchanges in a generic heavy traffic load agent-based application. Experiments have shown a throughput of 1.6Gbps per port at 100 MHz without packet loss and seamless scalability characteristics.

Categories and Subject Descriptors: C.1.4 [Computer Systems Organization]: Reconfigurable computing; B.3.6 [Hardware]: Reconfigurable Logic and Fpgas

Additional Key Words and Phrases: Multi-agent systems, Network-on-Chip, Open Core Protocol, agent-based simulation

ACM Reference Format:

Eduardo A. Gerlein, T.M.. McGinnity, Ammar Belatreche, and Sonya Coleman. 2015. Network on Chip Architecture for Multiagent Systems in FPGA. *ACM Trans. on Reconfigurable Technology and Systems (TRETS)*. *X*, *X*, Article XX (Xxxxx 2017), 24 pages.

DOI:http://dx.doi.org/10.1145/0000000.0000000

1 INTRODUCTION

A multi-agent system (MAS) is formed by a set of agents that coordinate and conjugate their abilities and resources to solve complex problems through concurrent interactions that lead to emergent effects at system level [Ferber 1999]. Numerous commercial and open-source - generally java-based - packages are available for the implementation of a software-bas MAS. Even though platforms such as JADE [Bellifemine et al. 2007],

Eduardo Gerlein is supported by a Vice-Chancellor Research Scholarship (VCRS) from the University of Ulster, as part of the Capital Markets Engineering project.

Author's addresses: Eduardo Gerlein, Departamento de Electrónica, Pontificia Universidad Javeriana, Bogotá, Colombia. e-mail: *egerlein@javeriana.edu.co*. T.M. MCGINNITY, College of Science and Technology, Nottingham Trent University, Nottingham, U.K., e-mail: *martin.mcginnity@ntu.ac.uk*; Ammar Belatreche, Faculty of Engineering and Environment, Northumbria University, Newcastle, U.K., e-mail: *ammar.belatreche@northumbria.ac.uk*; Sonya Coleman, Intelligent Systems Research Centre, Ulster University, Londonderry, U.K., e-mails: sa.coleman@ulster.ac.uk

DOI:http://dx.doi.org/10.1145/0000000.0000000

Cougaar [Anon 2012], MASON [Luke et al. 2005], SWARM [Minar et al. 1996] and BESA [González et al. 2003] are well engineered in terms of software design, the target computing architectures, i.e. serial processors, in which they are expected to be deployed limit their parallel computing. A high impact on performance is observed when a platform host a large scale MAS that includes several thousands of agents [Allan 2010], or scaling up to large simulations [Pawlaszczyk and Strassburger 2009].

Field Programmable Gate Arrays (FPGA) are a very promising technology for high performance computing with a highly parallel and flexible architecture [Gokhale et al. 2008]. However, although FPGA technology seems to be the next logical step in the development of multi-agent technology, only a very limited number of projects have reported multi-agent implementations in reconfigurable hardware. The current agent oriented programming (AOP) methodologies are not entirely appropriate to design and deploy MAS at microchip level [Bosse 2014]. Agents in hardware are difficult to engineer since there is no clear methodology for their design that incorporates a similar level of conceptualization to software implementations, while at the same time takes into account the specific requirements for FPGAs [O'Sullivan and Studdert 2005], i.e. limited memory resources, availability of special purpose hardware blocks and general purpose logic elements, routing and fitting inside the target device.

In this paper, agents deployed in a FPGA will be referred to as *hardware agents* as opposed to the traditional *software agents* that reside in a processor's program memory. There is a small number of reports exploring design techniques in order to deploy a hybrid system, where agents can be immersed in both software and embedded environments. Even fewer works report successful implementations on FPGA. The architecture and methodology presented in this paper are entirely agent-based, extending the agency concepts to generate a feasible communication model based on Network on Chip (NoC). The main objective is to merge naturally an agent oriented methodology and the hardware design flow to model and deploy MAS in FPGA. NoCs borrow concepts from large scale computer networks, providing a modular, flexible and scalable communication infrastructure for cores' interconnection at microchip level. A NoC is conformed by a set of interconnected routers and network interfaces, which allow to separate the problem of core´s functionality from the communications.

The proposed concept discussed in this paper is the result of a combination of a Star-NoC topology, scaling in a hierarchical fashion by means of the integration of lower level clusters, in conjunction with a message broadcast mechanism through standardized module interfaces. These interfaces implement the Open Core Protocol (OCP) [(OCP International Partnership) 2009]. The OCP is a socket-based interface that attempts to define point-to-point links between processing elements, giving freedom for choice and implementation of the final communication architecture. The OCP defines a configurable set of I/O signals and the handshaking protocol between two communicating entities using master/slave interfaces. Those interfaces are configured through a set of OCP parameters, most of them optional, to form different OCP profiles defined in the document compliant. While other projects have shown the feasibility of interconnecting agents using customized NoCs [Ebrahimi et al. 2011], to the best of the authors' knowledge, this paper is the first attempt to discuss a generic systematic approach as a general solution to address the particular problem of agent communications in FPGA using the NoC paradigm while at the same time taking into account both agency requirements and digital design practices. Large-scale NoCs have been used to enable bio-inspired neuro-computing platforms [Carrillo et al. 2013][Merolla et al. 2014], nevertheless the nature of such applications does not allow effective comparison of the achieved performance and the NoC metrics used to

describe them with the results discussed in this paper. To validate the proposed concept, a MAS comprised successively of 5, 30, 150 and 375 agents has been deployed in an Altera's Stratix IV FPGA to simulate intensive interactions in a generic agent-based application. For the experiments, a router microarchitecture is designed, as well as the OCP interfaces at the nodes to integrate the Event Driven Reactive Architecture (EDRA) [Gerlein et al. 2014a] [Gerlein et al. 2014b] with the NoC's interconnection fabric. Results have shown a zero load latency of seven clock cycles per hop and a high throughput of 1.6 Gbps represented by the injection of messages to the nodes every three clock cycles at full-load capacity. The NoC simulation traffic was run at a frequency of 100 MHz.

The remainder of the paper is organized as follows: Section 2 discusses earlier research into implementing agents in FPGA and the lessons learned in this endeavor. In Section 3, the EDRA model used to design agents in hardware is discussed. Section 4 presents the proposed hierarchical Star-NoC for MAS in FPGA and detailed design of the routers, network interfaces and their implementation results. Section 5 presents the implementation of heavy load communication simulation using a generic agent-based model. Section 6 concludes the paper.

2 RELATED WORK

The design of MAS for hardware and/or software/hardware hybrid platforms cannot be driven by a pure Agent Oriented Methodology because these methods address the problem from a purely software perspective rather than taking into account key hardware aspects. These aspects includes the amount of memory resources, special purpose hardware blocks, routing and fitting limitations, low level abstraction of hardware design, and major physical constraints at the deployment stage [Bosse 2014]. Chen *et al.* [2011] have indicated that regardless of FPGAs benefits, design of MAS in reconfigurable hardware requires the addressing of certain issues such as *system control complexity*, *degree of modifiability* and *universal communication abstractions*. System control complexity attempts to arbitrate opposing resource requests in a true concurrent context. *Degree of modifiability* addresses the changes in the design, that may be reflected in partially or even totally different hardware implementations that may lead to the appearance of new placement, fitting or timing issues in the newer version. In addition, universal communication abstractions are also desirable, since software implementations have been using them for many years. FPGA-based MAS must include structures to enable inter-agent communication. This communication modules must implement custom standard interfaces depending on the application and the particular requirements. The authors proposed a general architecture for hardware agents which includes wrapping circuits for internal control and communication management and a user logic area that encapsulates the agent functionality, isolating it from the communication infrastructure. The communications between the agents are proposed using standardized network-on-chip techniques.

Meng [Meng 2005] proposed a reconfigurable agent-based architecture taking advantage of FPGA technology to instantiate processors and logic units, which leads to the implementation of a reconfigurable multi-core system-on-chip (RMCSoC). A co-design approach is used, in which structured agents with repetitive and time consuming tasks are deployed using hardware modules. The agents that present more complex and irregular structure, are programmed in software using an embedded soft-processor. To evaluate the proposed model, an agent-based navigation system for a Pioneer 3DX robot is designed, where a set of concurrent agents in charge of sensor management and

acquisition of information are deployed on hardware. An approximate decrease of 50% in navigation time is observed in a dynamic environment with the FPGA-based MAS compared with the software approach.

An important application of hardware agents is fault tolerance for large scale computing systems as seen in [Lukovic and Christianos 2010] where the authors addressed the problem of multi-agent communication architecture with the development of a NoC. The particular application is a security agent-based network called the *secure-NoC*, to monitor and ensure the integrity of communications in a Multi-Processor System-on-Chip (MPSoC) composed of processing cores and communication elements (routers and network interfaces) interconnected through what the authors call a *data-NoC*. The security agents are organized in a hierarchical fat-tree architecture. A similar approach was proposed in [Ebrahimi et al. 2011], where a MAS is in charge of monitoring the congestion and status of the routers of a main *data*-NoC. The monitoring agents collect the congestion information from their attached routers and distribute it to all the agents in a broadcast fashion to optimize adaptive routing based on the local and global congestion to balance work load across the network. This work discusses an interesting approach which implements broadcasting the information to every agent in the system through a NoC using a mesh topology. Although the application fulfills the purpose of reducing the *data*-NoC congestion, a mesh topology requires the incorporation of a routing scheme which impacts the overall latency in a router. In addition, as discussed in [Carrillo et al. 2013], a single mesh architecture is no naturally scalable, and a hierarchical approach is needed to reach scalability. The approach used in this paper, scales through a hierarchical Star-NoC and avoids routing stages inside the network switches using message broadcasting to reduce router latency.

In [Bosse 2014], the author attempts to integrate the agents' behavior , interaction, and mobility with what is called an Agent-On-Chip processing architecture (AoC). This approach uses a reconfigurable pipelined communicating architecture implemented with finite-state machines. Bosse develops an intelligent sensor network to monitor the structural fitness of the actuators and joints in a robotic manipulator. A set of autonomous nodes are connected to nine gauge sensors and interconnected to each other in a 10 by 10 mesh network. The nodes are synthetized in a Xilinx XC3S1000 FPGA. This project presents an interesting insight into the importance of selecting simplified architectures such as finite state machines (FSM) to design agents at hardware level and the implementation of communication modules separated from the agent functionality to provide social interactions. Furthermore, in [Bosse 2015], the author extends the Agent-orientated Programming Language (AAPL) to construct an activity-based model for the agent behavior at programming level, which results appropriate for hardware implementations. In the activity-based model the agent´s behavior is determined by the internal state, which in turn is modified by activities. The activity-based model can be mapped to hardware structures through FSM, as well as software implementations, and simulation using a unified methodology which in turn favors a seamless integration of a MAS hosted by a hybrid platform. Nevertheless, specific issues regarding a scalable integration of agents are not addressed directly. The approach presented in this paper goes further in that direction, discussing NoC considerations from an agency perspective suitable for deployment in FPGA devices. A router implementation is discussed for a hierarchical Star-NoC, using standard interfaces that enable the interconnection of agents to the network regardless the selected architecture to implement the agents´ functionalities.

Naji, Wells and Etzkorn in [Naji, Wells, et al. 2004] directly address the problem of transferring multi-agent paradigms over reconfigurable hardware, proposing an architecture at agent-level that takes as a basis the traditional Believe-Desire-Intention model. The authors propose a general architecture defining a set of ports and signals to enable data flow. The agents are connected in a peer-to-peer fashion, providing high-speed interactions but also limiting scalability as the agents in the system must be modified if new entities are included in later versions or if new functionalities are added.

Communications in the agency paradigm considers every agent as a *master* in a *master*/*slave* scheme, who is capable of engaging in communicative acts as necessary. This capability may generate an overload in bus arbitration modules present in bus-based communications. Bus-based systems usually follow a *master*/*slave* approach where the channel is stalled during any transaction. While bus-based communication strategies were sufficient for years to fulfil the communication requirements in complex chips, the current size, level of integration scale and speed of modern systems have required the design of innovative solutions to manage data exchange between the cores. Therefore the use of buses as a primary communication mechanism for MAS in FPGA should be discarded if large scale multi-agent applications are intended. NoCs appeared early in the 21st century [Benini and De Micheli 2002], using analogous concepts taken from large scale communications networks, and applying them to the embedded SoC domain. In contrast to bus-based architectures, NoCs route packets of data from the source to the destination components, via a network fabric that consists of switches (routers) and interconnection links (wires). The model proposed in this paper uses the Event Driven Reactive Architecture (EDRA) introduced in [Gerlein et al. 2014a] and posteriorly in [Gerlein et al. 2014b], to design and implement the agents at individual level, while also implementing a hierarchical Star-NoC at social level to enable agents to communicate, cooperate and scale.

3 EVENT DRIVEN REACTIVE ARCHITECTURE (EDRA)

The EDRA model is based on the assumption that intelligence in agents can be achieved by the aggregation of interacting reactive modules called *behaviors* as discussed by Brooks in his subsumption architecture [Brooks 1986]. A reactive approach naturally encompasses the agency paradigm with a digital design implementation. EDRA describes the internal agent's micro-architecture using the Organizational Approach for Agent Oriented Programming (AOPOA) methodology introduced in [González and Torres 2006], establishing a structured fine-grained task decomposition inside agents to generate reactive *behaviors* which are triggered by *events*. The *behaviors* are linked with consistent hardware interfaces to enable internal flow of information.

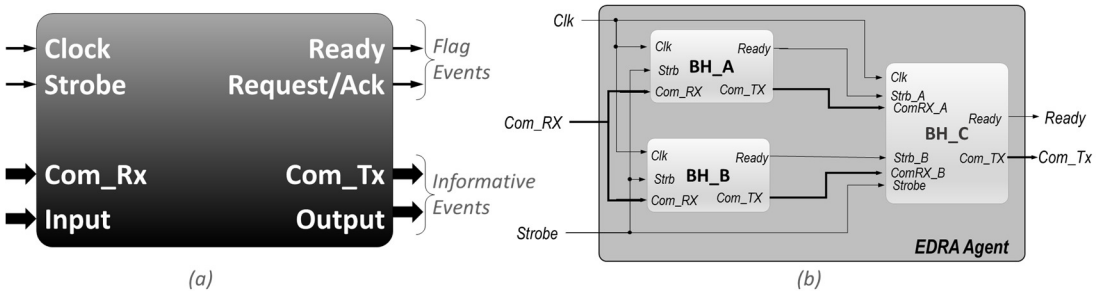

Figure 1. (a) Architecture of an EDRA Behavior extended from the model proposed in [Naji, Etzkorn, et al. 2004]. (b) EDRA agent with 3 behaviors.

To define the agents, the AOPOA methodology describes a system as an iterative tree of *goals*, in which the root represents the system's main *goals* and the child nodes represent the agent's *sub-goals* named *roles*. The final leaf nodes represent the agents that will be deployed in the system. To complete the EDRA model, the task decomposition must continue inside the agents seeking the simpler of these *roles* or *tasks*, until reaching the lowest level of complexity. The complexity of such simple *tasks* must be evaluated by a heuristic assessment, due to the fact that they must be implemented in hardware modules. These basic *task* structures are called *behaviors*. Inside an agent, the *behaviors* will interact using *events* that represent a set of predefined signals in a hardware communication interface. The defined *behaviors* are modelled and implemented as independent hardware modules. These hardware modules are further instantiated in a single hardware description file to construct each individual agent. At conceptual level, the *events* are grouped into two types: *informative* and *flag events*. *Informative* e*vents* involve the passing of information, i.e., processed results or information from and to the environment. *Flag events* are used to make announcements, requests, or acknowledgements. At implementation level, the *events* are modelled as set of signals in a port interface. Figure 1-a presents a generic architecture of a *behavior*, in which *flag events* are modelled as bit-related signals and *informative events* are represented as vector-related signals depending on the needs and characteristics of the information. Figure 1-b represents an EDRA agent conformed by three interacting behaviors. More than one interface port is possible as shown in the number of strobe ports present in the Behaviour C in the Figure 1-b.

The ports used for managing *flag events* are: (a) *Clock* – mandatory signal in digital systems to synchronize general execution; (b) *Strobe* – listening port used for activation of a particular *behavior* to execute the programmed *task* or to indicate that particular information is available to be processed; (c) *Ready* – signal used to indicate its programmed task is finished; (d) *Request*/*Ack* – used to call for the execution of a desired *task* or associated to a *request-data-acknowledge* protocol. For *informative events,* four types of ports are defined: (a) *Communication Reception Port (ComRx)* –handles incoming information from inside the system; (b) *External input* – handles incoming information from an external source; (c) *Communication Transmission Port (ComTx)* –handles outgoing information inside the system; (d) *External output* –used to send data or information to an external destination. For a more detailed description of the EDRA we reffer the autor to [Gerlein et al. 2014a] and [Gerlein et al. 2014b].

4 HIERARCHICAL STAR-NOC FOR MAS COMMUNICATIONS

The following sections will discuss a novel approach intended to address the problem of large scale multi-agent interactions in FPGA using a hierarchical Star-NoC model. The basic star topology connects several nodes in a point-to-point basis, with a central node or switch in charge of distributing messages. One switch and its associated nodes will be called a *cluster*. The communication strategy followed by the NoC will be *broadcasting*. Every message generated by a node will be concentrated and retransmitted to all the nodes in the *cluster*. Despite the fact that a *broadcast* strategy might increment the network traffic, it benefits the implementation of MAS cooperation strategies such as bid/ask auctions, partial global planning and blackboard-based cooperation where messages are intended for more than one agent at a time. The discrimination of messages will occur at the node network interfaces, programmed to receive certain type or types of messages (inserted in the message header) instead of using subscription-based communications that tend to serialize the transmission of *multicast* and *broadcast* messages. The use of *broadcasting* with filtering at nodes also simplifies the design of the routers since no

routing schemes are required, though the challenge in this case is the implementation of virtual channels to guarantee message preservation and quality of service.

Other NoC topologies, such as mesh-based, present a one-to-one correspondence between nodes and routers, which proportionally increments the use of resources and increases the power consumption. A star architecture uses a lower number of routers than other topologies which is advantageous in the sense that the majority of power consumption in a SoC is due to the interconnection fabric. Unlike other topologies, the star structure presents the lowest *hop count* as well as the lowest *diameter* (equal to 1). The degree of the nodes is 1 (connected to just one switch) and the degree of the switch will be n+1 (connected to n nodes plus one switch in a higher hierarchy). Latency associated with each hop is reduced by the avoidance of routing algorithms since the *broadcast* scheme is selected by default, although *one-to-many* and *one-to-one* routing strategies can be also implemented. Additionally, in general a 2D topological layout, such as the one given by the Star topology, can be easily mapped using standard fabrication architectures, which include FPGAs, therefore reducing synthesis time [Jerger and Peh 2009].

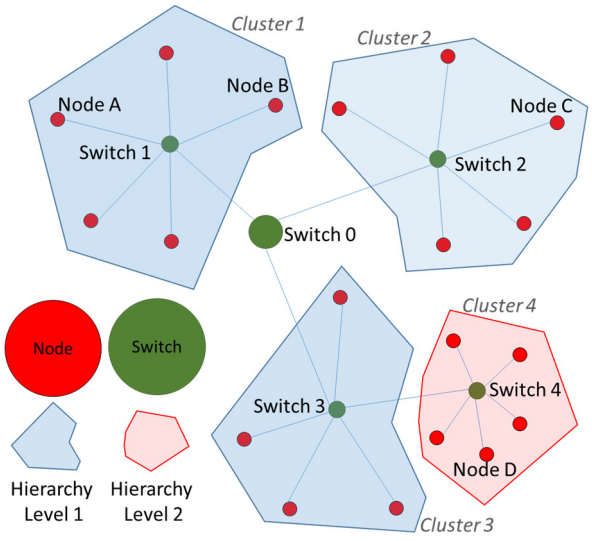

Figure 2. Hierarchical Star Topology

4.1 Hierarchical Scalability

Several benefits have been identified for hierarchical NoCs [Das et al. 2009]: a hierarchical approach allows a modular implementation providing a low-cost topology framework which presents short physical paths between interconnecting nodes which in turn favors low power consumption and enhanced signal integrity. At the same time, a hierarchical approach offers the possibility of optimizing traffic locality, favoring in turn, the creation of micro-societies in multi-agent applications. The Star-NoC scalability will be achieved using a hierarchical structure extending to deeper levels of clusters as needed according to the number of agents required in a particular application. A top-level hierarchy switch interconnects individual clusters that in turn are attached to nodes-agents as depicted in Figure 2. The clusters will be connected to a higher hierarchical switch connected in turn only to other switches. A lower level hierarchy is possible by replacing a node by a switch as shown in Figure 2, where the *Switch 4* is the core of the lowest hierarchy cluster. The routers implement standard interfaces, therefore the inclusion of lower level cluster

hierarchies is performed in a seamless fashion. The degree of each of the cluster's switches shown in Figure 2 is d=6 (five ports used for nodes, one port used for a higher hierarchy switch) and d=3 for the top-level switch (*Switch 0*), although similar architecture for all the routers can be used leaving open the remaining ports.

4.2 Latency

Latency is given by the number of hops that a message has to travel to reach a target destination node. This number of *hops* increments with the inclusion of lower clusters. Therefore the total latency is a function of the individual *hop* latency plus the delays produced by the traffic at each switch. The network traffic might degrade the bandwidth if the number of generated messages is close to the saturation threshold. The worst case scenario is reached when all the nodes inject messages at the same time and a particular message is the latest in the queue to be attended by the scheduler module within the router. In Figure 2, if a communication from the agent placed in Node A is targeted at an agent located in Node B, the corresponding message will be routed using only one *hop* through their respective cluster switch (*Switch 1-Node B*), since both are placed in the same cluster; therefore latency will be determined by one *hop* assuming zero traffic load. Additional latency must be taken into account if more than one agent in Cluster 1 injects messages at the same time, and the message sent by Node A is not the first one in the channel assignment. If the same message is targeted at the agent located at Node C in Cluster 2, three more hops must be taken into account, i.e. *Switch 1-Switch 0*, *Switch 0-Switch 2*, *Switch 0-Node C*. If the message is targeted to Node D at Cluster 4, the route will be given by *Switch 1-Switch 0*, *Switch 0-Switch 3*, *Switch 3-Switch 4*, *Switch 4-Node D*, with a total of 4 hops. Therefore, the network presents a deterministic number of *hops* given by the level of depth in the hierarchy, hence there is a fixed path delay between a pair of nodes as opposed to the situation observed in a bus-shared scheme where adding processing elements decreases the communication capacity of the system.

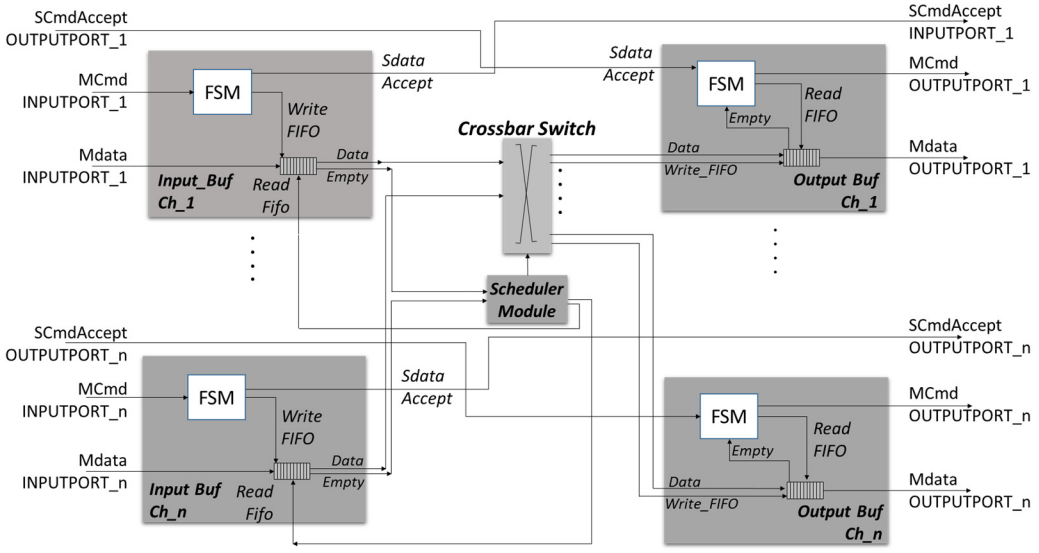

Figure 3. Router Microarchitecture for a Star NoC. Input and Output buffers communicate using OCP interfaces. The use of broadcast avoids a routing engine, only a scheduler module that executes a round robin assignment policy only between those ports that required transmission. Input and output buffers are controlled by FSMs

4.3 Router Microarchitecture

This section describes the design of the router microarchitecture implementing the necessary hardware modules to transfer and receive packets to/from any of the attached nodes in a broadcast manner. The router also provides communication through the different NoC levels when the NoC grows hierarchically. The router microarchitecture is depicted in Figure 3. The following sections will discuss the particular implementation for each one of the constitutive modules in the router.

4.3.1 Input and Output Buffers.

The main components inside the switches are the input and output buffers. The buffers are implemented as a FSM and a FIFO data structure. The FSM is in charge of reading and issuing OCP signals and controlling the writing and reading cycles of the FIFO, which in turn is in charge of buffering incoming packets. For the input buffers the FSM implements an OCP simplified slave *peripheral* profile with a subset of three basic signals: *MCmd*, *MData* and *SCmdAccept*. For the output buffers (also called pushback buffers), the FSM implements a master *peripheral* profile issuing the same three signals as in the input buffers. Timing of the signals follow the OCP´s specification. The FSMs in both input and output buffers, define three states: *idle*, *reading* and *running*. The *idle* state is the default state and will be triggered by the presence of data in the FIFO. The *reading* state is a temporary transition that allows the module to fetch data from the output port of the FIFO and register the content for transmission. The *running* state will issue the corresponding data at the output of the buffer module. This action will be coordinated by the scheduler module in the case of the input buffers, or the signal *SCmdAccept* issued by an OCP slave interface located at the nodes for the output buffers.

4.3.2 Scheduler Module and Crossbar Switch

The scheduler module is in charge of allocating the router resources coming from multiple incoming requests from the input ports. Typically, multi-agent communications present irregular traffic patterns that must be attended as soon as possible, ensuring packet preservation and avoiding starvation of ports. This characteristic was exploited in [Carrillo et al. 2013] and a similar approach was implemented in this case. The scheduler module ignores those ports that do not require attention and performs a round robin assignation only on those ports that indicated the presence of packets to be transmitted. The functionality is implemented using a sequential circuit and takes one clock cycle to generate the corresponding assignment of the channel to the winning port. This approach is well suited for a scenario where the agents exhibit a high communication rate, allocating equal priorities to all incoming ports that require the transmission of data while avoiding the spending time attending to idle ports.

The crossbar switch is implemented using a multiplexer/demultiplexer architecture that simultaneously places the data coming from the particular port that has granted the use of the channel into the output buffers for its corresponding packet broadcast to the related nodes in the NoC, as depicted in Figure 4. The delay in the crossbar switch is determined by the technology used in the FPGA, since its implementation is entirely combinational driven strictly by the transition time of the gates.

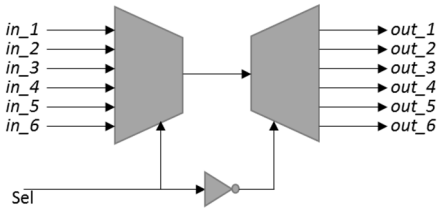

Figure 4. Crossbar Switch implementation. Multiplexer/demultiplexer architecture that places simultaneously the winner port data into the output ports

4.3.3 Network Interface

The network interface enables the transfers of information between the EDRA-agents and the NoC switches using OCP master and slave interfaces. As shown in Figure 5, a NoC node is composed by the agent and the network interface. In this case, the agents are constructed using the EDRA model, which provides a set of signals to trigger the agent's internal behaviors. The network interface is composed of a pair of master and slave OCP interfaces, similar to the input and output buffers in the NoC switches.

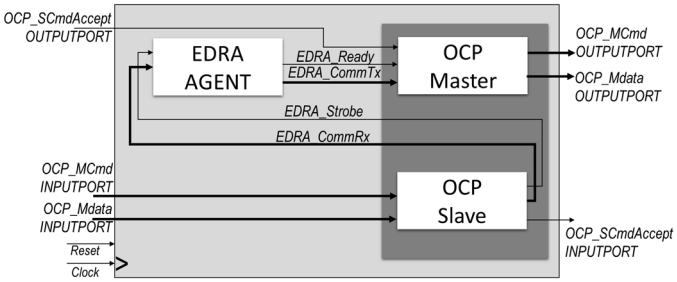

Figure 5. NoC Node Architecture. The network interface bridges the agents with the NoC switches using OCP master and slave interfaces and generate the signals in the EDRA model required by the agents

4.4 Router Implementation Results

This section discusses an analysis of resource utilization, latency and throughput obtained for the router implementation. The router was implemented in Very High Speed Integrated Circuits Hardware Description Language (VHDL) using Quartus II. The system was deployed in the Altera's Stratix IV FPGA running at 100 MHz. The buffers at the input and output ports were implemented using memory blocks in the FPGA. For the traffic simulation application discussed in this paper each buffer has a word length of 48-bits and a depth of 32, 64 and 128 words, although those parameters are customizable. The routers are implemented with six input ports and six output ports which allow connection of six nodes, or five nodes and a higher level router. The number of ports is also a parameter that can be modified according to the requirements of a particular application. For this set of parameters, the compilation process reported the results shown in Table 1. The resource utilization in all cases represents less than 1% of the resources available in the FPGA, which in turn benefits the scalability of the NoC given the lower area footprint achieved by a single router. It is important to note that the information presented in Table 1 only takes into account the router logic, and does not include the resource utilization incurred by the network interfaces at the nodes. The FIFOs are implemented using the memory blocks of the FPGA, therefore only the registers that are part of the OCP finite state machines and the scheduler module are part of the reported memory logic elements in Table 1.

Table 1. Router implementation results in an Altera's Stratix IV FPGA

	32 words-depth	64 words-depth	128 words-depth
Combinational LE	1670	1790	2611
Memory LE	684	684	1368
Dedicated logic registers	2077	2692	3433
Total Thermal Power Dissipation	1573.38 mW	1580.37 mW	1593.46 mW
Dynamic Thermal Power Dissipation	35.49 mW	43.07 mW	53.64 mW

Table 2 presents the memory utilization incurred by a single router attributed to the input and output OCP interfaces which include FIFOs. The total memory requirements for a system depends on the FIFO size at the routers and the number of levels implemented in the network. The network depth in turn determines the maximum number of routers for a particular configuration and therefore the maximum number of agents. For instance, a one level NoC only requires one switch and can accommodate as many agents as ports in the router. For the experiments discussed in this paper, a six-port router was implemented, therefore five agents at the nodes of the NoC. For six-port switches in a two-level hierarchical NoC, seven switches are required and it can accommodate a maximum of 30 agents. A three-level NoC requires a maximum of 37 switches and can implement up to 150 agents, and so on. The size of the FIFOs depends on the attempted traffic load for an application and trade-offs between the size of buffers, word length and NoC's operating frequency can be reached to optimize the use of resources.

Table 2. Memory requirements for a single router with respect to the FIFO's size.

Fifo Size	Word Size (bits)	Memory Required (bits)	Memory Required (KBytes)	Fifo Size	Word Size (bits)	Memory Required (bits)	Memory Required (KBytes)	Fifo Size	Word Size (bits)	Memory Required (bits)	Memory Required (KBytes)
16	32	6,144	0.75	32	32	12,288	1.5	64	32	24,576	3
	64	12,288	1.5		64	24,576	3		64	49,152	6
	128	24,576	3		128	49,152	6		128	98,304	12
	256	49,152	6		256	98,304	12		256	196,608	24
	512	98,304	12		512	196,608	24		512	393,216	48

The latency obtained for the router is constrained by the state transitions in the OCP interfaces in both the input and output ports. Each interface contributes a latency of three clock cycles, plus one more clock cycle used by the scheduler module to grant the use of the channel. Therefore, the latency at zero traffic load for a single router is 7 clock cycles, which represents 70ns at an operating frequency of 100 MHz. In contrast, at full traffic load, when more than one message is injected into the switch, the output ports are able to inject messages to the nodes every three clock cycles, i.e. 30ns at 100 Mhz. Table 3 presents different values of throughput at different operating frequencies observed for a router measured in a single output channel when the clock frequency and packet width are varied. Generally, the design of routers for NoCs implies fixing internal parameters such as the number of ports, FIFO size and packet size. For a specific application the packet size is determined by the availability of resources but mainly by the communication protocol and routing schemes. In packet-switched networks, large pieces of information are divided in smaller units in a process called packetization. Such smaller pieces of information are routed through the network based on the destination address contained within each packet's header. Breaking down information into packets allows the use smaller port sizes and even different routes. This strategy also implies extra latency in the network interfaces that must spend time in the packetization process, and also the implementation of techniques to re-build messages that are received out-of-order when multiples routes are used. The broadcasting mechanism proposed in this paper avoids the

implementation of techniques to re-build messages that are received out-of-order when multiples routes are used. The broadcasting mechanism proposed in this paper avoids the routing schemes, and therefore overhead bits that must be included in the packets' headers are not necessary, allowing a full payload in the messages.

Table 3. Throughput observed by a single router measured in one output channel

Clock Freq (MHz)	Packet Width (Bits)	Router Troughput /channel (Gbps)
10	32	0.107
	64	0.213
	128	0.427
	256	0.853
100	32	1.067
	64	2.133
	128	4.267
	256	8.533
200	32	2.133
	64	4.267
	128	8.533
	256	17.067

The synthetic traffic example discussed in further sections assumes a packet size of 48-bits which is enough to encode the information attempted (see Section 6.2). Table 4 presents the impact of increasing the port size in the router implementation in terms of logic elements used, necessary memory in each OCP interface at the input and output ports and power consumption. The FIFO depth for the OCP interfaces is fixed at 64-words. The columns labeled as "Increment" represent the relative incremental growth with the previous step.

Table 4. Impact of Port-Size in the Router implementation with a FIFO depth of 64-words

Port Size (bits)	Resources (LEs)	Resources Increment	Memory per Port (bits)	Memory Increment	Power Consumption (mW)	Power Consumption Increment
32	1241	--	2048	--	1545.43	--
48	1589	28.04%	3072	50.00%	1580.37	2.3%
64	1886	18.69%	4096	33.33%	1604.19	1.5%
128	3074	63.00%	8192	100.00%	1699.47	5.9%
256	8475	175.66%	16384	100.00%	1890.03	11.2%

As seen in Table 4, the effect in power consumption is the lowest as the port size increments. As expected, regardless the fixed FIFO depth, incrementing the word length in the packets will impact greatly in the memory usage. Resource utilization is affected proportionally as the word-length increases, since internal paths and registers are also proportionally increased.

For these experiments, a six-port router was implemented, therefore six agents at the nodes of the NoC for single cluster. A hierarchal Star-NoC implies that lower level cluster routers will be attached to five agents and an upper level router. The reason for implementing six ports obeys mainly to take advantage of the architecture inside the Stratix IV family used for these experiments. The Stratix IV internal Logic Elements or Adaptive Logic Modules are the basic building blocks of logic, which are able to implement any logic function with up to six inputs. Logic functions inside the routers such as the scheduler module and the crossbar switch will relate six control signals. Implementing

more than six or less ports might result in a waste of resources leaving partial LEs available for construction of the arbitrage functions.

Table 5. Maximum number of nodes and routers according to the number of ports in a full populated four-level hierarchical Star-NoC

No. Ports	No. Nodes	No. Routers	Nodes-Router Ratio
3	24	16	0.667
4	108	41	0.380
5	320	86	0.269
6	750	157	0.209
7	1512	260	0.172
8	2744	401	0.146
9	4608	586	0.127
10	7290	821	0.113

On the other hand, using more ports allow to accommodate more nodes in the network, but that is not necessarily the most efficient solution. Table 5 presents the maximum number of ports and routers needed to populate a four-level hierarchical Star-NoC. In addition, the Nodes-Router ratio is included. The node-router ratio in a 2D-mesh topology corresponds to 1, due to the correspondence between nodes and routers in the NoC. The advantage of using a Star-NoC is that this 1-to-1 correspondence decreases, favoring the use of resources and decreasing the total power consumption of the network since less switches are needed to accommodate the same number of nodes. Figure 6 presents the node-router ratio for a four-level hierarchical Star-NoC. As it can be seen, the exponential trend reaches the flatten area around six ports, which means that even though it is possible to accommodate a higher number of nodes with a 10-port switch, increasing the number of ports in the routers does not represent a significant gain given by the node-router ratio after reaching 6 ports.

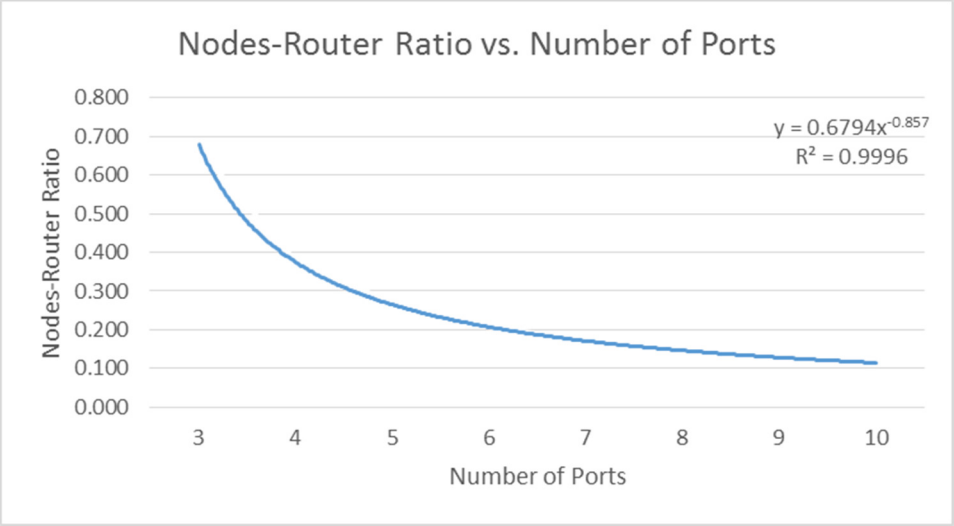

Figure 6. Nodes-Router ratio vs. number of ports in a full populated four-level Star-NoC

The queuing theory described in [Kiasari et al. 2013] is used to evaluate the router performance. The queuing theory allows to model the switch as a queuing system similar to a population of customers that request attention from a determined service facility. In this case only one server is present although the queuing theory allows the presence of more than one server. When a new customer arrives to the facility, it waits in a queue until the server is available to address the service request. The queuing theory requires to characterize the arrival of requests and the internal process of the router, in the form $A/B/m/K$-S, where A describes the distribution of the arrival time, B represents the distribution of the service time, m is the number of available servers and K the maximum depth of the queue. S is optional and it describes the service policy, which in this case is a *round robin* (*RR*) with *priority* (*PR*) as described in *section 4.3.2*. The arrival time in this experiment was a deterministic (*D*) synchronized process where all the agents in the nodes issue messages at the same time. The service time is also *deterministic* (*D*). The router implementation exhibits seven clock cycles of latency at zero load which corresponds to the first issuing round of messages. Subsequently, the router presents a message issuing latency of three clock cycles which correspond to the full load latency when more than one input port request attention simultaneously. In this manner, according to the queuing theory, the router is modelled as a $D/D/1/32$-*RR*-*PR* process. The scenario depicted in the Figure 7-a is used in a set of experiments where the agents´ issuing time is varied at every round. The communication rate started with one clock cycle until saturation and subsequently increasing the issuing intervals until finding the saturation threshold when the router is able to operate without saturating the input buffers. The minimum injection rate obtained for each one of the nodes is one message every 18 clock cycles, with an average latency of 14.49 clock cycles. The *hop* latency is measured observed for each message starting at the input buffer of a certain port and observing the time needed to be issued by the output port. Injecting messages at lower rate will cause input buffer saturation that in turn, can be mitigated by augmenting the size of the FIFOs.

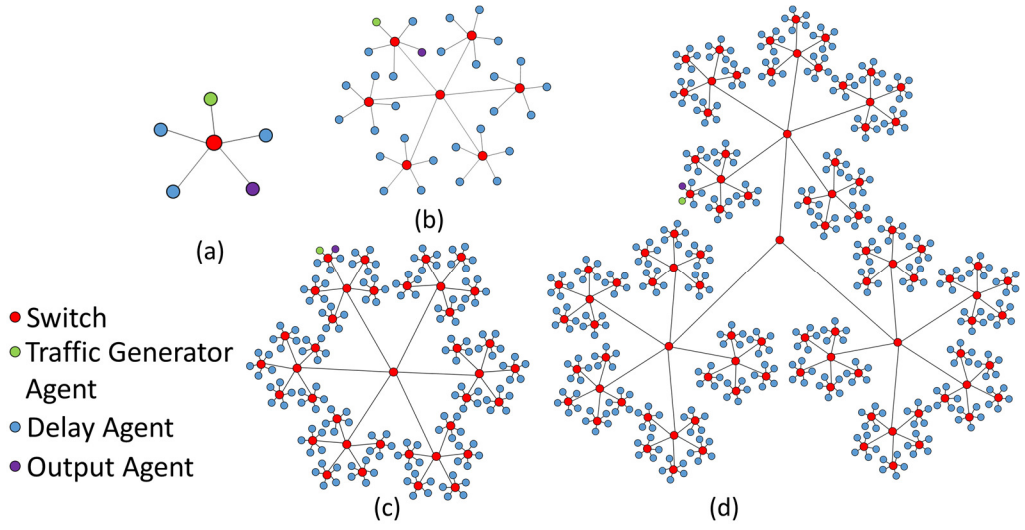

Figure 7. NoC Scenarios. (a) Single Cluster; (b) Two-level 6 cluster; (c) Three-level 30 cluster; (d) Four-Level 75 clusters

5 MULTI-AGENT NOC TRAFFIC SIMULATION

A synthetic MAS was created with the purpose of analyzing communication traffic using the hierarchical Star-NoC while at the same time verifying the performance of the proposed model using common metrics. Four scenarios were constructed as depicted in Figure 7: (a) single cluster with a total of 5 agents, (b) two-level NoC with a total of 30 agents, (c) three-level NoC with a total of 150 agents, and (d) four-level NoC with 375 agents. The size of the MAS in Figure 7-d is limited by the FPGA device, which is unable to accommodate more clusters. The generated traffic in the NoC does not correspond to a specific multi-agent application. The traffic simulation among a set of communicating agents allows the verification of the requirements in the routers and network interfaces. A *broadcasting* approach was used to obtain a hierarchal distribution of messages. The traffic simulation represents a generic heavy traffic load application where any message issued by an agent in a lower level of the NoC can reach any other node in the system.

5.1 Agent Description

Three types of agents were created using VHDL to run a network traffic simulation following the generic architecture EDRA/OCP (see Figure 5). The different agents' roles are:

- *Traffic Generator Agent (*Figure 8-a*)*: plays the role of communication initiator programmed to issue a certain number of messages at a predefined time intervals. The traffic generator agent is composed by one EDRA behavior in charge of the message generation implemented using a FSM and a network interface module.
- *Delay Agent (*Figure 8-*b)*: simulates a generic information processing core, receiving a particular input message type and issuing different output types after a predefined time delay, adding the agent's ID to the message route. The delay agent is composed by the network interface and two EDRA behaviors, one in charge of the input message filtering and one in charge of the simulated processing core.
- *Output Agent (*Figure 8-c*)*: receives as input a particular message type and stores it in a data structure to be sent to a host PC using a serial protocol. The output agent is composed of the network interface, the input filter behavior and the serial communication behavior to interface with the host PC.

5.2 Network Messages

Each message issued and transmitted is composed of 48-bits. The message is distributed in the following fields:

- *Message ID*: 4-bit field issued by the traffic generator agent with the purpose of being used for message tracking analysis.
- *Message Type*: identifies the type of message issued by the agents, and used to filter the incoming messages. Three types of messages were used labelled with the identifiers: "01" (MsgType1), "10" (MsgType2) and "11" (MsgType 3). The traffic generator agent always issues messages MsgType1. The delays agents are configured in two versions: [input: MsgType1 / output:MsgType2] and [input: MsgType2 / output:MsgType3]. The output agent only receives messages MsgType3 as presented in Table 6.

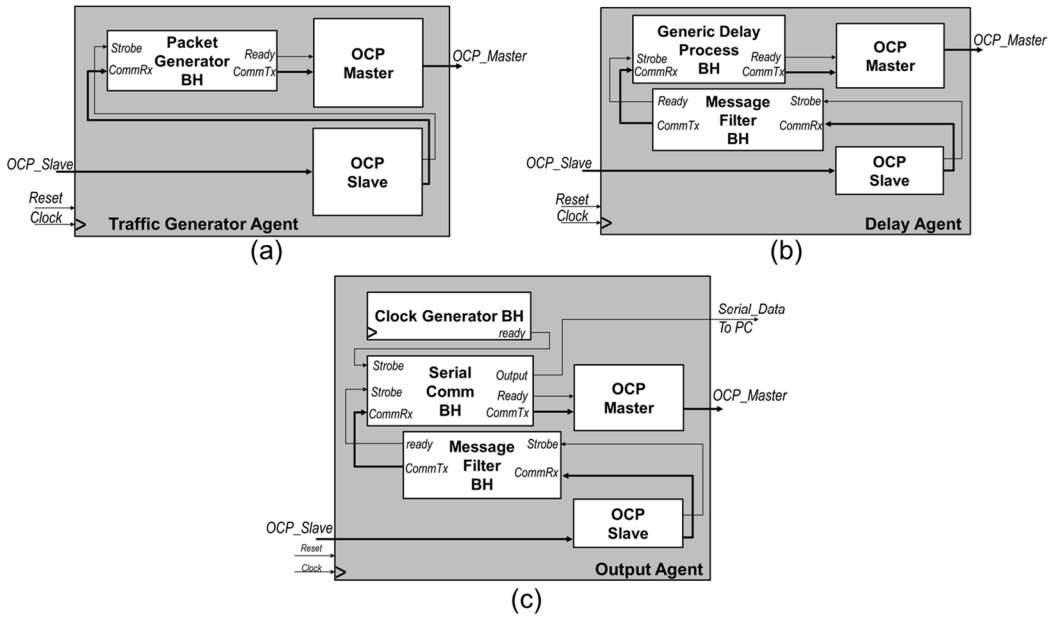

Figure 8. Block Diagram of the agents in the communication traffic experiments. (a) Traffic Generator Agent; (b) Delay Agent; (c) Output Agent

- *Node Count*: 2-bit field used to keep track of how many nodes a particular packet has reached. The node count is updated by every agent.
- *Route*: this field is incrementally constructed at each node agent which adds its *agent ID* (9 bits) once a message is received and the incoming message type is verified. In this manner, the messages are labelled with the route that they have travelled to arrive at the target destination. Hence, each message can only be processed by four agents, i.e. the traffic generator agent, one delay agent type 1, one delay agent type 2 and the output agent.

The traffic simulation terminates when the output agent receives the exact number of expected messages determined by the number of messages issued by the traffic generator agent, and the number of agents of each type deployed in the system. A text file is generated at the end of the simulation including the message log to verify the different routes that messages have followed.

5.3 Experimental Results

Figure 9 depicts the traffic observed at a single output port of the central router when operating at full capacity, showing the issuing of commands and data every three clock cycles, which corroborates the router throughput values in Table 3.

Table 6. Type of agents according the input and output type of messages

Agent Type	Input Messages	Output Messages
Traffic Generator Agent	---	MsgType1
Delay Agent type1	MsgType1	MsgType 2
Delay Agent type2	MsgType1	MsgType 2
Output Agent	MsgType 3	Serial Port to PC

The signals in Figure 9 correspond to the OCP master interface, clock and reset for an output port in a single router respectively. The routers have shown a consistent

throughput per channel of 1.6 Gbps with 48-bit messages running at 100 MHz. It is important to highlight that the performance here reported corresponds an experimental observation while running the network at full capacity, i.e. all Delay agents injecting messages in a concurrent fashion. It is worth noting that in the case of using a clock frequency of 156.25 MHz such as in 10 Gigabit Ethernet applications, this particular router implementation is capable of managing a throughput of 3.33 Gbps using packets of 64-bits and 13.33 Gbps using packets of 256 bits. Given the real payload present in each packet for 10 Gigabit Ethernet, which appears after filtering the network stack protocols, the implementation of the Star-Router theoretically would be able to be used in these kind of applications.

Injecting messages at lower rate than 18 clock cycles will lead to reach the saturation threshold, making the network to enter in deadlock states. Deadlocks imply that the buffers become rapidly saturated with undelivered messages in both the network interfaces and the routers. Due to the fact that the routers work in a pipelined fashion, they are able to inject messages every 3 clock cycles, and therefore not being saturated by the injection rate of nodes if this rate does not exceed 18 clock cycles. For the larger scenario (Figure 7-d) of 4-level hierarchy, the average delay for a message is 28 clock cycles at zero traffic load, which is represented for seven clock cycles for agents in the same cluster, and 49 clock cycles for a pair of agents that must complete the longest path to communicate. Table 7 presents the agent distribution for each scenario, showing the expected number of messages by the output agent when the traffic generator agent issues only one message. The message log shows the completion of all possible routes without packet loss for every one of the scenarios implemented (see Figure 7), which shows that the approach of broadcasting messages through a hierarchical Star NoC is a feasible solution to enable inter-agent communication in FPGA. Table 7 also presents the maximum number of *hops* that a message should traverse to reach any other agent in the system for each scenario which allows a deterministic analysis of message latency depending on the number of hierarchy levels implemented. The message filtering at the nodes has also shown advantages in reducing the latency in the switches by avoiding routing mechanisms without incurring timing penalties for the agents thanks to the combinational circuit implemented for this task.

In this application the buffers will be saturated stalling the traffic in the network before completing the expected number of messages producing deadlock effect if the FIFO depth at the routers and network adapters at the nodes is not enough to handle the generated traffic. A heuristic approach was followed in these experiments to determine the FIFO depth required for each scenario to avoid deadlock, incrementing to the power of two the number of registers until the traffic simulation was able to be successfully completed. For these experiments, a large amount of storage space in the FIFOs is needed to ensure the packet preservation and to avoid the network saturation, given the low latency selected for the delay agents and the fact that the simulation is designed to reach almost full capacity since each agent is generating packets in a concurrent fashion. Each FIFO at the OCP interfaces are set to 128-words depth. Nevertheless, if the processing agents present a latency large enough compared to the router latency, the amount of registers utilized by the buffers can be reduced in size to optimize of the use of resources.

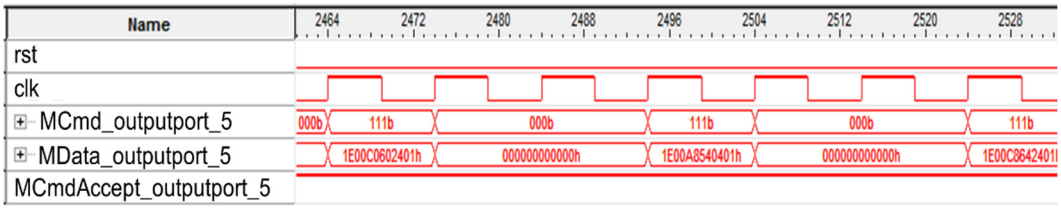

Figure 9. Traffic observed at single output port in the router with the router operating at full capacity. As observed, messages are sent by an output port every three clock cycles.

As seen in the Table 7, the total amount of expected messages is 33,522 in the four-level simulation. Once the first message is issued, it reaches its targeted agents (*Delay Agent type1*) in the worst scenario in 49 clock cycles, i.e. latency of seven cycles per hop at zero traffic load for the first message and seven maximum hops for the longest possible path. The first message is processed by 151 of those *Delay Agent type1*, which in turn will re-issue the message after a random delay varying from two clock cycles and a 50 clock cycles. The maximum interval programmed a *Delay Agent* is comparable with the router latency, and therefore the traffic generated by the *Delay Agent*s will grow rapidly along the network. Increasing the operation frequency of the routers is an alternative approach to increasing FIFO depth used to avoid deadlock and network saturation. According to the synthesis report, the routers can operate at a theoretical frequency of 192.23 MHz, which represents a latency of 36.4 ns (7 clock cycles), almost half of the latency obtained at 100 MHz. Increasing the operating frequency represents additional challenges for the designer while working with multiple clock domains which is far from a trivial task. Nevertheless the overclocked network would be able to handle a heavy traffic load without the necessity of using large FIFOs. A trade-off between frequency of operation and use of resources must be found for a particular application if there is a strong limitation in the use of resources.

Table 7. Agent distribution by type and number of messages expected by the output agent.

	Single Cluster	**Two-level**	**Three-level**	**Four-level**
Number of Agents	5	30	150	375
Number of clusters	1	6	30	75
Number of Switches	1	7	37	93
Max Hop Count	1	3	5	7
Number of Traffic Generator Agent	1	1	1	1
Number of Delay Agent type1	2	13	61	151
Number of Delay Agent type2	1	15	87	222
Number of Output Agent	1	1	1	1
Number of output msgs	2	195	5307	33522

The use of a hierarchical Star-NoC presents some advantages when compared with a mesh network in a similar application as discussed in this paper. Table 8 presents a comparison between the proposed hierarchical Star-NoC and a mesh network. In this case, a 20-by-20 network is required in order to include the 375 agents implemented in the four-level model which represents the larger scenario discussed in this paper. A mesh network presents a one-to-one correspondence between routers and processing elements, therefore a total of 400 routers must be instantiated to populate the entire network. Assuming that a broadcast scheme is implemented, which means that no routing modules need to be included in the router, and thus no increment in the router area footprint, the resource increment needed to implement a mesh network will corresponds to 330%. The network diameter which corresponds to the maximum hop count is also increased from seven in the Star-NoC to 38 in a 20-by-20 mesh network. The average hop count in a k-by-k mesh network operating in a broadcast scheme corresponds to $(3k\text{-}1)/2$ (for k even) [Park et al. 2012]. Consequently, at zero traffic load the average hop count will be increased to 29.5,

which in turn corresponds to 206.5 clock cycles assuming the same latency of 7 clock cycles in the router. Finally, a software benchmark was implemented in a Xeon quad-core CPU running a 2.8GHz with 12 GB of RAM, using the java-based multi-agent platform called BESA [González et al. 2003], which manages the agent communication concurrence using a multithreading scheme. The Star-NoC implementation discussed in this paper has achieved an acceleration factor of 275x when deployed in an Altera Stratix IV running at 100 MHz.

Table 8. Star-NoC and Mesh Network comparison

Metric / Topology	Number of Agents	Number of Routers	Resource Increment	Max Hop Count (Diameter)	Average latency @zero traffic load (clock cycles
Hirarchical Star	375	93	0%	7	28
20-by-20 Mesh	375	400	330%	38	206.5

6 DISCUSSION

This paper has discussed a solution for agent communications in FPGA. The proposed approach consists of a hierarchical Star-NoC topology that exhibits modularity, data parallelism, scalability, low latency and high throughput. The communication protocol selected is broadcasting, which reduces latency at the network nodes by eliminating the routing mechanism. The messages are filtered at the nodes by a low latency combinational circuit using the message type field located in the packet header, enabling the messages to be targeted to one or several agents in the system. The OCP interface, a standardized socket-based communication architecture, was used to enable the transactions between ports in the routers and the nodes in the NoC, which allows seamless integration with different agents or processing elements through a well-defined set of signals and timing rules. The proposed NoC for the MAS concept was verified by the construction of an agent-based traffic simulation and deployed in an Altera Stratix IV FPGA to assess the main NoC metrics and performance according to the degree of scalability implemented. The OCP master-slave interfaces are able to handle effectively the packet load in a broadcasting increased traffic scenario. In addition, a broadcasting mechanism privileges the implementation of agent strategies for cooperation such as blackboard, partial global planning and market-oriented bid/ask auctions in which messages are generally intended for more than one agent in the system. Local communications to create agent micro-societies can be achieved using the OCP command signal to limit the transmission of messages to local clusters enabling multicasting. Although the implementation in this paper assumes the use of large data ports to enable the transactions of larger messages in one cycle, further resource optimizations can be reached using smaller port sizes and implementing burst approaches, also supported by the OCP interface. Additionally, it is possible to improve latency in the routers by the elimination of the pushback FIFOs at the output ports which contributes to the overall latency.

The total number of message types will be determined by the specific application. A large number of types will be required if the quantity of point-to-point messages increases. Even though the size of the field within the packet used by the message type can be customized, the increase of the point-to-point messages will produce unnecessary traffic all over the network. To overcome this drawback, hierarchical star networks can take advantage of locality, in the sense that agents that must share larger amount of point-to-point messages

can be physically connected within the same cluster or across low level nodes that share a common higher level switch. A similar circuit such as the one implemented at the agents to filter incoming messages, can be added to the input buffers inside the routers to manage local traffic. In this manner it is possible to limit messages to pass through regions of the network outside the local clusters where the communication is intended. The communication strategy for hardware agents proposed in this paper, opens a door to develop agent-based applications in many fields that requires the speed-up capabilities of hardware implementations. Large-scale agent-based simulations such as market behavior, disease spreading and natural disaster simulations, can be achieved investigating the connection capabilities between FPGA boards to create large clusters. The current FPGA boards present different connectivity interfaces such as PCI, Ethernet, proprietary interconnection modules, etc., that provide high level of versatility in this regard. Efficient interfaces must be created between the particular communication ports in the boards with EDRA and the Star-NoC. Additional research must be conducted to generate hardware-oriented agent-based strategies to manage diverse clock-domains, synchronization, and resource management and information consistency. Additionally, the use of the common agent-based communication protocols in software such as the FIPA-compliant [IEEE 2012] can also be explored to generate hybrid solutions that include PC and HPC platforms. The use of higher level communication protocols can be incorporated be means of constructing the messages in a way that they fulfill the particular protocol complaint. This includes larger package payloads in some cases. The selected OCP interfaces, provide burst modes to send a serialized set of packages to build larger messages if the use or large sized ports is not an option due to limited resources. This particular capability must be addressed at design time for a particular application if a hybrid MAS is designed. Additionally, in cases where heterogeneous platforms are attempted to be integrated, generally a single circuitry module is designed to serve as a bridge among the platforms that adjust timing, packet headers and packet size: PCI or Ethernet controllers, WI-FI modules, USARTs, etc. In those cases, the interfaces can also play the role of incorporating the required packet headers for higher-lever interpretation to those simpler messages intended for communication only between hardware agents, preserving in this way the small area footprint incurred by the routers in the Star-NoC. The total amount of agents that can be accommodated within an FPGA device is determined mainly by the area footprint used to implement the agent´s functionality. In addition, the traffic load for a particular application determines the size of the buffers in both the routers and the network interfaces which also determines the size of the system. Large scale agent-based simulations such as market behavior [Chen 2003], disease spreading [Stroud et al. 2007] and genome searching using reduction algorithms [Varghese et al. 2014], can be achieved taking advantage of the connection capabilities between FPGA boards such as PCI, Ethernet, proprietary interconnection modules, etc., to create large clusters [Markettos et al. 2014] in addition to the inherent parallel processing and accelerated results that reconfigurable hardware offers. Efficient interfaces must be created in such case between the particular communication ports in the boards with the Star-NoC.

Mobile agents that are implemented as pieces of code able to move across different platforms are not easily achieved in hardware. FPGAs are able to instantiate soft-core processors that can act as agents with adequate programming and more importantly, maintaining consistent interfaces with the proposed NoC as discussed in [Baklouti and Abid 2014], which in turn can facilitate the agent´s payload mobility. The main limitation in the model discussed in this paper lies in a scenario where the dynamic change in the number of agents is required at run-time. Some recent FPGA devices offer dynamic partial reconfiguration, in which an FPGA can be “reprogrammed” at run time. These

characteristics need to be explored under certain restrictions, since the reconfigurable partitions require large times for reprogramming compared with the frequencies achieved in high performance applications, which in turn implies that the remaining static regions should not depend on the outputs from any partial reconfigurable regions while the procedure is been executed. In addition, partial reconfiguration assumes that different portions of the FPGA can implement different logic, swapping from one to another when required, with the restriction that the new logic should be pre-synthesized and stored in the form of bitstreams in the configuration memory blocks (CRAM) that controls the functionality of the different regions in the FPGA. As opposed to software implementations where dynamic memory allocation at run-time is possible to create and destroy agents, FPGAs must remain relatively static after compilation. Agents can implement flag mechanisms to determine their active state without being physically erased from the system. The use of EDRA modelling and the hierarchical Star-NoC provides a complete model to implement MAS in FPGAs and opens the door for future agent implementations to reach acceleration in large scale agent-based simulations and real life applications. Future work will include the validation and refinement of the model on real life large-scale applications.

REFERENCES

(OCP International Partnership). 2009. Open Core Protocol Specification 3.0. (2009), 494.

Rob Allan. 2010. *Survey of Agent Based Modelling and Simulation Tools - Technical ReportsDL-TR-2010-007*,

Anon. 2012. Cognitive Agent Architecture (Cougaar) Open Source Project site. (2012). http://cougaar.org

Mouna Baklouti and Mohamed Abid. 2014. Multi-Softcore Architecture on FPGA. *Int. J. Reconfigurable Comput.* 2014 (2014), 13.

Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood. 2007. *Developing Multi-Agent Systems with JADE*, West Sussex, England: John Wiley & Sons Ltd.

Luca Benini and G. De Micheli. 2002. Networks on chips: a new SoC paradigm. *Computer (Long. Beach. Calif).* 35, 1 (2002), 70–78. DOI:http://dx.doi.org/10.1109/2.976921

Stefan Bosse. 2015. Design and simulation of material-integrated distributed sensor processing with a code-based agent platform and mobile multi-agent systems. *Sensors* 15, 2 (2015), 4513–4549. DOI:http://dx.doi.org/10.3390/s150204513

Stefan Bosse. 2014. Distributed Agent-Based Computing in Material-Embedded Sensor Network Systems With the Agent-on-Chip Architecture. *IEEE Sens. J.* 14, 7 (July 2014), 2159–2170. DOI:http://dx.doi.org/10.1109/JSEN.2014.2301938

R.A. Brooks. 1986. A robust layered control system for a mobile robot. *IEEE J. Robot. Autom.* 2, 1 (1986), 14–23. DOI:http://dx.doi.org/10.1109/JRA.1986.1087032

Snaider Carrillo et al. 2013. Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations. *IEEE Trans. Parallel Distrib. Syst.* 24, 12 (December 2013), 2451–2461. DOI:http://dx.doi.org/10.1109/TPDS.2012.289

Edward Chen, Victor Gusev, Dorian Sabaz, Lesley Shannon, and William A. Gruver. 2011. Holonic and Multi-Agent Systems for Manufacturing Vladimír Mařík, Pavel Vrba, & Paulo Leitão, eds. *Holonic Multi-Agent Syst.* 6867 (2011), 94–102. DOI:http://dx.doi.org/10.1007/978-3-642-23181-0

Xiaorong Chen. 2003. Co-evolutionary multi-agent-based modeling of artificial stock market by using the GP approach. In *Proc. IEEE Int. Conf. on Computational Intelligence for Financial Engineering (CIFEr'03)*. Hong Kong, China: IEEE, 159–165. DOI:http://dx.doi.org/10.1109/CIFER.2003.1196256

Reetuparna Das, Soumya Eachempati, Asit K. Mishra, Vijaykrishnan Narayanan, and Chita R. Das. 2009. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *IEEE 15th Int'l Symp. on High Performance Computer Architecture (HPCA'09)*. Raleigh, NC, USA: IEEE, 175–186. DOI:http://dx.doi.org/10.1109/HPCA.2009.4798252

Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, and Hannu Tenhunen. 2011. Agent-based On-Chip Network Using Efficient Selection Method. In *19th Int.Conf. on VLSI and System-on-Chip (VLSI-SoC), IEEE/IFIP2011*. Hong Kong, China: IEEE, 284–289. DOI:http://dx.doi.org/10.1109/VLSISoC.2011.6081593

Jacques Ferber. 1999. *Multiagent Systems: An Introduction to Distributed AI*, Addison-Wesley.

Eduardo A. Gerlein, T.M. McGinnity, Ammar Belatreche, Sonya Coleman, and Yuhua Li. 2014a. Hardware-based Agent Modelling : Event-Driven Reactive Architecture (EDRA). In *13th Int.Conf. on Autonomous Agents and Multiagent Systems - AAMAS14*. Paris, France: International Foundation for Autonomous

Agents and Multiagent Systems (IFAAMAS), 1497–1498.
Eduardo A. Gerlein, T.M. McGinnity, Ammar Belatreche, Sonya Coleman, and Yuhua Li. 2014b. Multi-agent Pre-trade Analysis Acceleration in FPGA. In *Int'l Conf. on Computational Intelligence for Financial Engineering and Economics - CIFEr2014*. London, U.K.: IEEE.
Maya Gokhale et al. 2008. Hardware Technologies for High-Performance Data-Intensive Computing. *Computer (Long. Beach. Calif).* 41, 4 (April 2008), 60–68. DOI:http://dx.doi.org/10.1109/MC.2008.125
Enrique González, Jamir Avila, and César Bustacara. 2003. BESA: Behavior-oriented, Event-driven and Social-based Agent Framework. In *Parallel and Distributed Processing Techniques and Applications - PDPTA'03*. Las Vegas, Nevada, USA: CSREA Press.
Enrique González and Miguel Torres. 2006. Organizational Approach for Agent Oriented Programming. In *8th Int. Conf. on Enterprise Information Systems - ICEIS*. Paphos - Cyprus, 75–80.
IEEE. 2012. The Foundation of Intelligent Physical Agents (FIPA). (2012). http://fipa.org/
Natalie Enright Jerger and Li-Shiuan Peh. 2009. *On-Chip Networks* Madison Mark Hill, University ofWisconsin, ed., Morgan & Claypool Publishers.
Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu. 2013. Mathematical formalisms for performance evaluation of networks-on-chip. *ACM Comput. Surv.* 45, 3 (2013), 1–41. DOI:http://dx.doi.org/10.1145/2480741.2480755
Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. 2005. MASON: A Multiagent Simulation Environment. *Simulation* 81, 7 (July 2005), 517–527. DOI:http://dx.doi.org/10.1177/0037549705058073
Slobodan Lukovic and Nikolaos Christianos. 2010. Hierarchical multi-agent protection system for NoC based MPSoCs. In *Proc. of the Int. Workshop on Security and Dependability for Resource Constrained Embedded Systems - S&D4RCES '10*. New York, New York, USA: ACM Press, 1. DOI:http://dx.doi.org/10.1145/1868433.1868441
A. Theodore Markettos, Paul J. Fox, Simon W. Moore, and Andrew W. Moore. 2014. Interconnect for commodity FPGA clusters : standardized or customized ? In *Field Programmable Logic and Applications, FPL´14*. 1–8.
Yan Meng. 2005. An Agent-based Reconfigurable System-on-Chip Architecture for Real-time Systems. In Laurence T. Yang, Evi Syukur, & Seng W. Loke, eds. *2nd. Int. Conf. on Embedded Software and Systems (ICESS'05)*. Xian, China: IEEE, 166–173. DOI:http://dx.doi.org/10.1109/ICESS.2005.21
Paul A. Merolla et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science (80-.).* 345, 6197 (2014), 668–673. DOI:http://dx.doi.org/10.1126/science.1254642
Nelson Minar, Roger Burkhart, Chris Langton, and Manor Askenazi. 1996. *The swarm simulation system: A toolkit for building multi-agent simulations - Working Paper 96-06-042*, Santa Fe, USA.
Hamid Reza Naji, Letha Etzkorn, and B. Ear. Wells. 2004. Applying multi agent techniques to reconfigurable systems. *Adv. Eng. Softw.* 35, 7 (July 2004), 401–413. DOI:http://dx.doi.org/10.1016/j.advengsoft.2004.05.008
Hamid Reza Naji, B. Ear. Wells, and Letha Etzkorn. 2004. Creating an adaptive embedded system by applying multi-agent techniques to reconfigurable hardware. *Futur. Gener. Comput. Syst.* 20, 6 (August 2004), 1055–1081. DOI:http://dx.doi.org/10.1016/j.future.2004.02.002
Timothy O'Sullivan and Richard Studdert. 2005. Agent technology and reconfigurable computing for mobile devices. In *Proceedings of the ACM Symposium on Applied computing - SAC '05*. Santa Fe, New Mexico, USA: ACM Press, 963. DOI:http://dx.doi.org/10.1145/1066677.1066901
Sunghyun Park, Tushar Krishna, Chia-Hsin Chen, Bhavya Daya, Anantha Chandrakasan, and Li-Shiuan Peh. 2012. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI. In *Proc. 49th Annual Confe. on Design Automation - DAC '12*. New York, New York, USA: ACM Press, 398. DOI:http://dx.doi.org/10.1145/2228360.2228431
Dirk Pawlaszczyk and Steffen Strassburger. 2009. Scalability in distributed simulations of agent-based models. In *Proc. of the 2009 Winter Simulation Conference (WSC)*. Austin, USA: IEEE, 1189–1200. DOI:http://dx.doi.org/10.1109/WSC.2009.5429429
Phillip Stroud, Sara Del Valle, Stephen Sydoriak, Jane Riese, and Susan Mniszewski. 2007. Spatial Dynamics of Pandemic Influenza in a Massive Artificial Society. *J. Artif. Soc. Soc. Simul.* 10, 4 (2007), 3. DOI:http://dx.doi.org/Article
Blesson Varghese, Gerard McKee, and Vassil Alexandrov. 2014. Automating fault tolerance in high-performance computational biological jobs using multi-agent approaches. *Comput. Biol. Med.* 48, 1 (2014), 28–41. DOI:http://dx.doi.org/10.1016/j.compbiomed.2014.02.005