Hardware-based Agent Modelling: Event-Driven Reactive Architecture (EDRA) (Extended Abstract)

Eduardo A. Gerlein
gerlein-e @email.uslter.ac.uk

T.M. McGinnity
tm.mcginnity@ulster.ac.uk

Ammar Belatreche
a.belatreche@ulster.ac.uk

Sonya Coleman
sa.coleman@ulster.ac.uk

Yuhua Li
y.li@ulster.ac.uk

University of Ulster, Intelligent Systems Research Centre. Londonderry, Northern Ireland (UK)

ABSTRACT
Multi-Agent Systems (MAS) have been recognised as a promising solution to address complex problems in many areas. However such systems are extremely hungry in terms of computational resources. Field Programmable Gate Arrays (FPGAs) offer great performance improvement over software implementations in terms of computational resource allocation but applications of multi-agent systems in such hardware have been poorly explored. This paper describes an Event Driven Reactive Architecture (EDRA), which is a novel multi-agent architecture for reconfigurable hardware. The EDRA approach enables the design and implementation of the internal architecture of agents targeted to be deployed in FPGA, based on fine-grained task decomposition to generate reactive structures triggered by signals through consistent hardware interfaces that enable the internal flow of information.

Categories and Subject Descriptors
I.2.11 [**Distributed Artificial Intelligence**]: Languages and structures, Multi-agent systems

General Terms
Design, Theory, Languages.

Keywords
Multi-Agent Systems, Agent Architectures, FPGA, Reconfigurable Hardware.

1. INTRODUCTION
A system of interacting agents is in essence highly complex and parallel, and therefore consumes substantial computing power [1]. Generally agent-based frameworks and APIs are targeted at traditional serial processors, whose lack of inherent parallelism may seriously affect both performance and the scalability of models when simulating at massive scale. Practical experiences in parallel and distributed implementations have shown that such implementations have difficulties in scaling up to large simulations [10]. Field Programmable Gate Arrays (FPGA) are a very promising technology for high performance computing with a highly parallel and flexible architecture [5]. Paradoxically, although reconfigurable hardware seems be the next logical step in the development of multi-agent technology, only a very limited number of projects have reported reconfigurable hardware multi-agent implementations[3] [7] [8].

To date, hardware agents are difficult to engineer since there is no clear methodology for design and deployment that incorporates a similar level of conceptualization as in software implementations while at the same time, takes into account the specific requirements for deployment in reconfigurable hardware [9]. The EDRA model proposed in this paper presents a systematic approach for agent design targeted for FPGA implementation, based on fine-grained task decomposition at agent level to generate reactive structures called *behaviours*, triggered by signals named *events* driven by consistent hardware interfaces that enable internal flow of information.

Appears in: Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.

2. EVENT-DRIVEN REACTIVE ARCHITECTURE (EDRA)
The EDRA model relies on Brooks' subsumption architecture [2], which states that intelligent agents can be modelled as an aggregation of interacting reactive modules called "triggered behaviours". EDRA implements the Organizational Approach for Agent Oriented Programming (AOPOA) methodology introduced in [6]. EDRA modelling goes deeper into the design than AOPOA, proposing an internal micro-architecture using an agency approach, establishing structured fine-grained task decomposition inside agents to generate reactive *behaviours* triggered by *events*, and linking them with consistent hardware interfaces to enable internal flow of information. The triggered *behaviours* described in the subsumption architecture can be easily encoded into circuits to map physical perception channels into hardware modules offering design simplicity and deterministic latency and throughput. The EDRA model supports its agency approach with two main concepts: (a) fine-grained decomposition of agent goals by means of *behaviours*; (b) interaction management through the design of consistent interfaces called *events*.

2.1 Agent's Goal Decomposition: *Behaviours*
According to the AOPOA methodology [6] a Multi-agent System (MAS) can be described as an iterative tree of *goals*, where the root represents the system's main *goals* and the derived child nodes represent the agent's *sub-goals* named *roles*. In this iterative decomposition, the final leaf nodes represent the agents that will be deployed in the system. To implement hardware agents, EDRA states that the task decomposition must continue inside the agents seeking the simpler of these *roles* or *tasks*, until reaching the lowest level of complexity, evaluated by heuristic assessment, due to the fact that they must be implemented using reconfigurable circuitry.

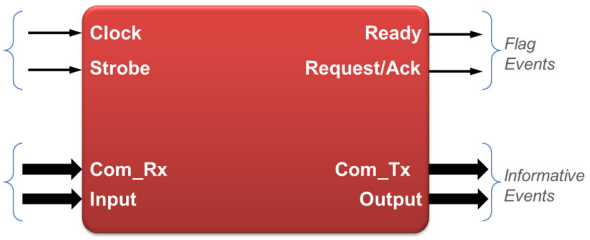

Figure 1. Architecture of an EDRA Behaviour based in the model proposed in [8] **. Events used to communicate behaviours are represented by ports and signals. Flag events can be represented by bit-related types and informative events can be represented also by bit-related and vector-related types**

These basic task structures are named *behaviours*. Inside an agent, *behaviours* will interact in order to achieve the particular goals for which they have been designed, in the same way that agents interact to pursue the system's goals, forming internal micro-societies. From a purely practical perspective, the defined *behaviours* are modelled and implemented as independent modules that are further instantiated in a single hardware description file to construct each individual agent.

The *behaviour* interactions are called *events,* and can be grouped into two types: *informative events* and *flag events*. *Informative events* involve the passing of raw information, i.e., processed results or information from and to the environment. *Flag events* are those representing announcements, requests, or acknowledge in handshake protocols. The *events* can be associated with I/O ports and Communication ports for *informative events* and Control ports for *flag events* in a VHDL entity. Naji's agent architecture [8], actually serves as basis for the *behaviour*'s architecture in the EDRA model.

2.2 Behaviours' Hardware Model

Figure 1 presents a generic architecture of a *behaviour*. In order to produce a synthesizable model in hardware, flag events can be represented by bit-related signals and *informative events* can be represented also by bit-related signals or vector-related signals depending on the needs and characteristics of the information.

The ports used for managing *flag events* are: (a) *Clock* – used in most of the complex digital circuits to coordinate general execution; (b) *Strobe* – listening port used for activation of a particular *behaviour,* requesting the execution of a programmed *task* or to indicate that particular information is available to be processed; (c) *Ready* – signal used by a *behaviour* to indicate its programed task is finished. It is meant to be connected to a strobe port in another *behaviour* or agent; (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)* – this port handles outgoing information inside the system; (d) *External output* –used to send data or information to an external destination. If communication protocols are consistent across the system it is possible to create generic facilitator *behaviours* that in practice will be beneficial for code reuse. On the other hand, external outputs such as drivers for sensors or actuators, or network adapters are generally media-dependent which must be customized according to different applications such as video interfaces or analog-to-digital sensor drivers.

If the system has relative low complexity or incorporates a small number of agents and the interactions between them are predefined *ad initio* at design time, the same approach can be adopted to communicate different agents using a peer-to-peer communication strategy as proposed in [8] .

3. DISCUSSION AND FUTURE WORK

The EDRA model presented in this paper allows a designer to construct internal agent architecture using a systematic approach favouring modular construction, flexibility and re-use of structures. EDRA provides consistency in signal treatment at the hardware level while keeping the agency abstraction intact. EDRA methodology has been used to implement a Multi-Agent Trading Engine suitable for high frequency trading [4]. Future research will investigate how to design a truly Multi-Agent System on Chip (MASoC) using the EDRA methodology at agent level, incorporating System on Chip techniques at society and system levels.

4. ACKNOWLEDGMENTS

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

5. REFERENCES

[1] Ajitha, S. et al. 2009. Predicting Performance of Multi-Agent systems during feasibility study. *2009 Int. Conf. on Intelligent Agent & Multi-Agent Systems* (Luxembourg, Jul. 2009), 1–5.

[2] Brooks, R.A. 1986. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*. 2, 1 (1986), 14–23.

[3] Chen, E. et al. 2011. Dynamic Partial Reconfigurable FPGA Framework For Agent Systems. *Proceedings of the Industrial Applications of Holonic and Multi-Agent Systems*. (2011).

[4] Gerlein, E.A. et al. 2014. Multi-agent Pre-trade Analysis Acceleration in FPGA. *Computational Intelligence for Financial Engineering and Economics - CIFEr2014* (London, U.K., 2014).

[5] Gokhale, M. et al. 2008. Hardware Technologies for High-Performance Data-Intensive Computing. *Computer*. 41, 4 (Apr. 2008), 60–68.

[6] González, E. and Torres, M. 2006. Organizational Approach for Agent Oriented Programming. *8th Int. Conf. on Enterprise Information Systems - ICEIS* (Paphos - Cyprus, 2006), 75–80.

[7] Meng, Y. 2006. An Agent-Based Architecture on Reconfigurable System-on-Chip for Real-Time Systems. *Handbook on Mobile and Ubiquitous Computing: Innovations and Perspectives*. American Scientific Publishers. 1–17.

[8] Naji, H.R. et al. 2004. Applying multi agent techniques to reconfigurable systems. *Advances in Engineering Software*. 35, 7 (Jul. 2004), 401–413.

[9] O'Sullivan, T. and Studdert, R. 2005. Agent technology and reconfigurable computing for mobile devices. *Proceedings of the 2005 ACM symposium on Applied computing - SAC '05* (New York, New York, USA, 2005), 963.

[10] Pawlaszczyk, D. 2009. Scalability in Distributed Simulations of Agent-Based Models. *WSC '09 Winter Simulation Conference*. section 3 (2009), 1189–1200.