VALIDATION of NEURAL NETWORKS ONTO FPGA

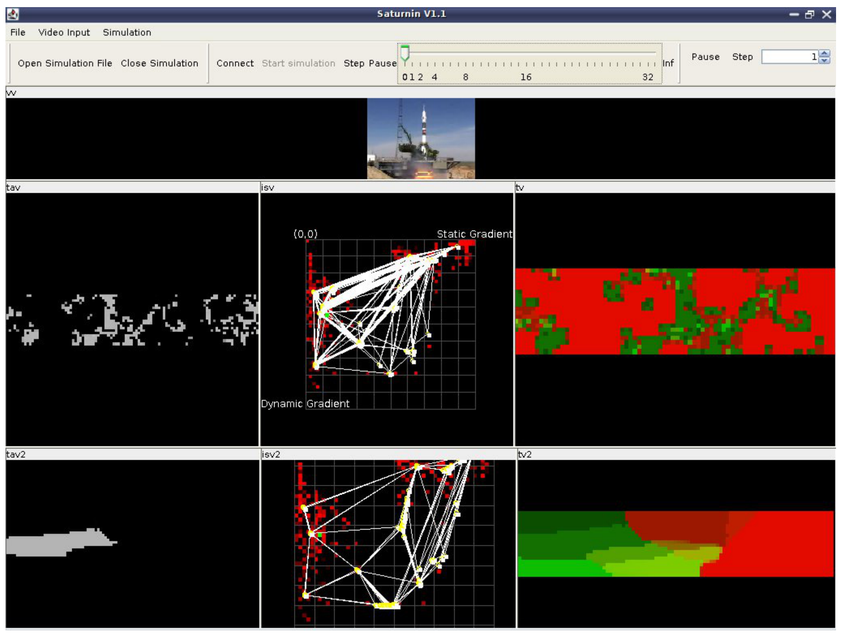
Laurent Rodriguez, Laurent Fiack and Benoît Miramond
ETIS / CNRS UMR 8051 - ENSEA - Université de Cergy-Pontoise
6 avenue du Ponceau F-95000 Cergy-Pontoise Cedex, France

Abstract

Recent works in artificial neural networks simulation showed that sizable networks, of the order of thousands of mammalian neurons, are now achievable. In the domain of microelectronics, rapid prototyping of complex hardware neural networks (hundreds) is still a major challenge for executing in real-time high-level cognitive tasks onto FPGAs. This paper addresses the related problem of validating these complex networks when the observation on the chip interface of the whole system, specially the high number of internal signals, is not feasible anymore.

CONTEXT OF HARDWARE NEURONS

Today, the fastest way to **prototype Artificial Neural Networks** into hardware devices is to use **Field Programmable Gate Arrays (FPGA)**. Beyond the scalability problem, their integration onto FPGA raises the problem of their validation, on-field, where the observation is limited to the chip pads.

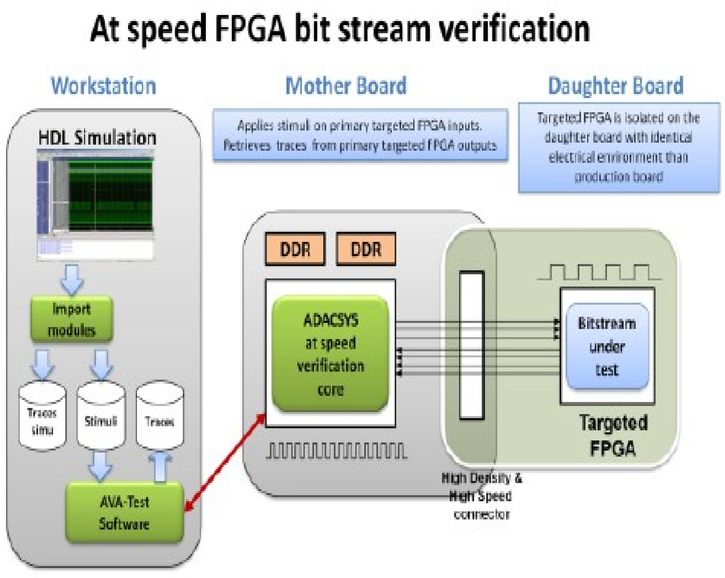


In the context of the SATURN project[3], we used and extended the verification methodology proposed by **Adacsys** to conserve the same verification process from high to low level of description:

- at high-level, the models are developed in C++ language using the **Saturnin simulation framework**, specialized in neural modeling.
- at RTL level, a VHDL description (fixed point version) of the high-level model is developed. The same GUI is reused to **inject stimuli and read back the traces** coming from the **behavioral simulation**. The precision error can be measured and compared to the model results.
- after place and route synthesis, the system runs onto the FPGA. The **Ava-soft tool** and its Hw transactor are used as communication channel to write stimuli and read the traces coming from the device.

VALIDATION METHODOLOGY ONTO FPGA

Adacsys's Hardware Verification and Debug Methodology has three major phases through the entire process of FPGA design development. It enables an easy silicon proven phase and allows to continually verify that any netlist modification still complies in hardware with the initial specification [1]:

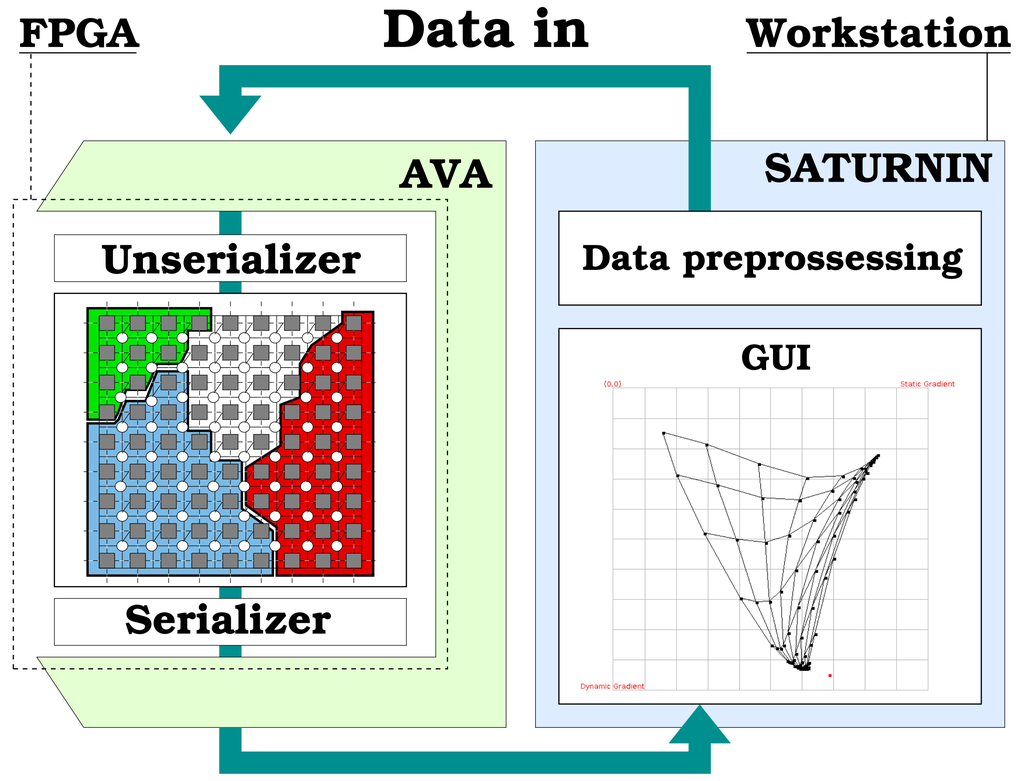


- **Module and IP level** hardware functional verification from RTL simulation are made with an **x10 000 acceleration factor**.
- FPGA level bit stream file **verification** is done **at speed** with same electric IO voltage on the application board.
- Application level at speed debug is made by **internal signal observation** and is done with or without a dedicated debug board connected to the available IOs.

APPLICATION TO SELF-ORGANIZING MAPS

We applied the verification methodology described in previous sections in the domain of mobile robotics and more precisely on ANN models for visual perception [2]. Following a bio-inspired approach, the controller of the robot has been organized as multiple neural networks, called maps:

- the **pre-processing maps**: each map has been designed to extract a specific characteristic from the visual input (edges, movements, texture ...). The related ANN does not use a learning process (the topology and the synaptic weights have been determined explicitly),
- and the **self-organizing maps**, that implements a complex learning process derived from the SOM general case.



- The size of our Static Maps highly depends on the resolution of the input camera. Onto the recent FPGA families the maximum number of user I/O is around 1000 pins per device (1200 onto Virtex 7 series and 840 onto the Stratix V series). The first property brought by our method is to **serialize automatically the input stimuli and output traces** between the workstation and the FPGA. By doing so, the size of the design under test (the size of its interface) is not limiting anymore but only slows down the debugging frequency.
- The entire network evolves as a non-linear process and the convergence of the system depends on the initial state and on the initial parameter's value. The proposed design flow, with **Ava-soft**, allows to **bring out the internal signals** from the design hierarchy. The user's HDL code is parsed in order to easily access to the chosen internal signals (the lateral connection in the case of our dynamic map). The logic specific to the communication (transactor) is added automatically. In all the cases, the traces coming from the FPGA board are brought **up to the Saturnin GUI** in order to conserve the same validation environment.

References

[1] Shakeel Jeeawoody, Erik Hochapfel, and Benoit Miramond. Best fpga design practices to comply with new standards while reducing field failures. In *Sopia Antipolis Micro-Electronic, SAME 2012*, october 2012.

[2] Laurent Rodriguez, Benoît Miramond, Imen Kalboussi, and Bertrand Granado. Embodied computing : self adaptation in bio-inspired reconfigurable architectures. In *proc. of 19th Reconfigurable Architectures Workshop*, page 6, june 2012.

[3] ANR SATURN team. The SATURN project - Self-Adaptive Technologies for Upgraded Reconfigurable Neural computing, french research agency (anr) http://projet-saturn.ensea.fr.