so_ip_edte_un_p

Product Specification

Decision Tree Ensemble Evaluation Core – Parallel Architecture

General Description

Machine learning is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A learner can take advantage of examples (data) to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables.

A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases.

There are many different predictive models (classifiers) in machine learning, including artificial neural networks (ANNs), decision trees (DTs) and recently introduced support vector machines (SVMs).

Recently a new way of making more accurate predictive models has emerged, ensemble learning.

Ensemble learning requires creation of a set of individually trained classifiers, typically decision trees (DTs) or neural networks, whose predictions have to be combined during the process of classification of previously unseen instances. Although simple, this idea has proved to be effective, producing systems that are more accurate then single a classifier.

All ensemble systems consist of two key components. First component is used to calculate the classifications of the current instance for every ensemble member. A second module is then needed to combine the classifications of individual classifiers that make up the ensemble into one single classification, in such a way that the correct decisions are amplified, and incorrect ones are cancelled out.

The main advantage of ensemble classifiers over single classifiers is in higher accuracy and greater robustness. The price to be paid is large amounts of memory to store the ensemble classifier and high computing power. This is because ensemble classifiers typically combine 30 or more individual classifiers, which means that 30+ times more memory and computational power is required if we want to get the same performance in classification speed as with the single classifier.

Ensemble classifiers are typically implemented in software. But in applications that require rapid classification or ensemble creation, hardware implementation is the only solution.

So_ip_edte_un_p core can be used to implement the ensemble member evaluation module as a part of an ensemble classifier consisting from decision tree with the previously defined structure directly in hardware. It implements every DT from the ensemble as a separate module using an area efficient sequential architecture in order to save logic resources.

So_ip_edte_un_p core is delivered with fully automated testbench and a compete set of tests allowing easy package validation at each stage of SoC design flow.

The so_ip_edte_un_p design is strictly synchronous with positive-edge clocking, no internal tri-states and a synchronous reset.

The so_ip_edte_un_p core can be evaluated using any evaluation platform available to the user before actual purchase. This is achieved by using a time-limited demonstration bit files for selected platform that allows the user to evaluate system performance under different usage scenarios.

Features

- Implements ensemble classifier comprised from DTs with previously defined structure
- Each ensemble member is implemented as a separate module using area efficient sequential architecture
- Supports classification problems that are defined by numerical attributes only
- DTs with univariate or multivariate tests are supported
- DTs with nonlinear tests are supported
- Ensemble can be composed from a combination of oblique and nonlinear DTs
- Possibility to alter the implemented DT structure during the actual operation
- No special IP blocks are needed to implement the core, only memory, adders and multipliers
- User can specify the number format for all DT parameters in order to achieve the best performance/size ratio after implementation

Applications

- Speech and handwriting recognition
- Computer vision
- Machine perception
- Pattern recognition
- Medical diagnosis
- Robot locomotion

Deliverables

- Source code:
  - VHDL Source Code
- VHDL test bench environment
  - Tests with reference responses
- Technical documentation
  - Installation notes
  - HDL core specification
  - Datasheet
- Instantiation templates
- Example application
- Technical Support
  - IP Core implementation support
  - Variable length maintenance
    - Delivery of IP Core updates, minor and major changes
    - Delivery of documentation updates

- Telephone & email support

Licensing

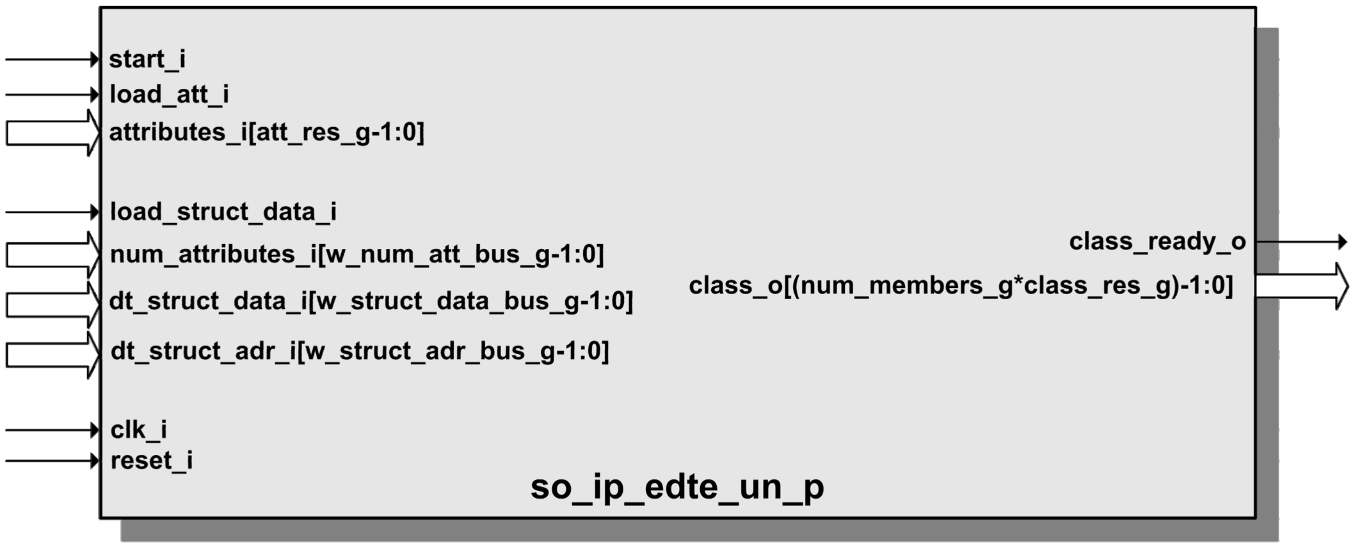
Netlist License

- Post-synthesis netlist
- Self checking testbench
- Test vectors for testing the core
- Place&Route scripts
- Constraints
- Instantiation templates
- Documentation

VHDL Source License

- VHDL RTL source code
- Complete verification plan together with testbenches needed to verify correct operation of the core
- Self checking testbench
- Vectors for testing the functionality of the core
- Simulation & synthesis scripts
- Documentation

Symbol

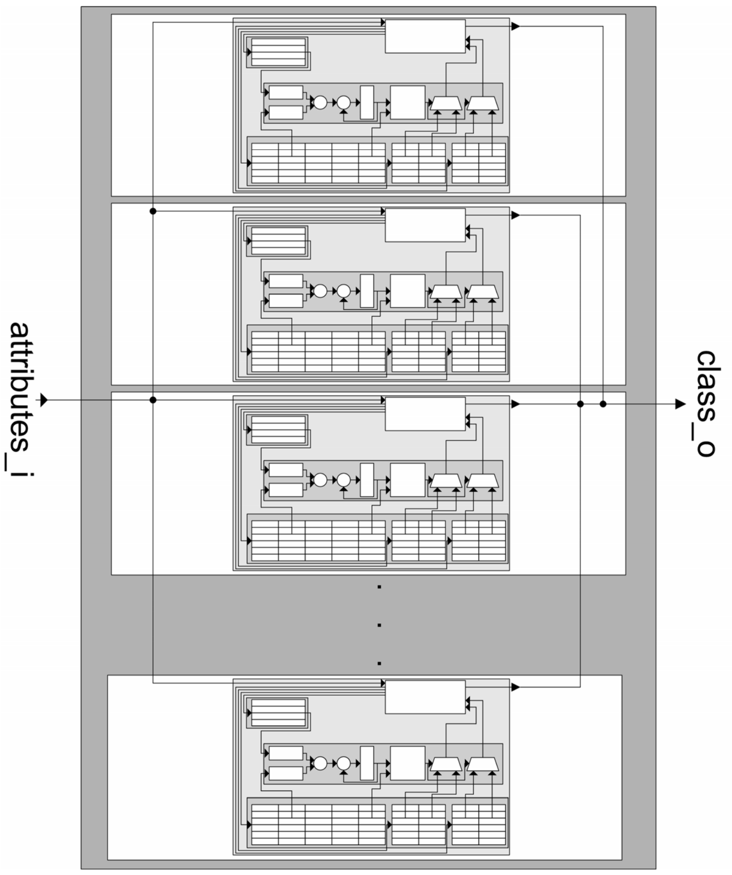

start_i
load_att_i
attributes_i[att_res_g-1:0]
load_struct_data_i
num_attributes_i[w_num_att_bus_g-1:0]
dt_struct_data_i[w_struct_data_bus_g-1:0]
dt_struct_adr_i[w_struct_adr_bus_g-1:0]
clk_i
reset_i
class_ready_o
class_o[(num_members_g*class_res_g)-1:0]
so_ip_edte_un_p

Pin Description

Name	Signal Direction	Description
Global Clocks and Reset Ports		
clk_i	Input	Main clock input
reset_i	Input	Main reset
Decision Tree Structural Interface		
load_struct_data_i	Input	Signal that is used to load the new structural information about the selected decision tree from the ensemble into the core
num_attributes_i[w_num_att_bus_g-1:0]	Input	Data bus that is used to convey the number of attributes that define the current classification problem
dt_struct_data_i[w_struct_data_bus_g-1:0]	Input	Data bus that is used to transfer the structural information about the selected decision tree from the ensemble that should be modified
dt_struct_adr_i[w_struct_adr_bus_g-1:0]	Input	Address bus that is used to specify what decision tree from the ensemble should be accessed
Input Instance Interface		
start_i	Input	Indication that all the attributes for the instance that should be classified are stored inside the

		core and that the classification process can commence
load_att_i	Input	Indication that there is a new attribute value for the current instance that should be stored inside the core
attributes_i[att_res_g-1:0]	Input	Value of the current attribute from the current instance that needs to be classified
Output Class Interface		
class_ready_o	Output	Indication that the classification for the instance is ready
class_o[(num_members_g*class_res_g)-1:0]	Output	Class values predicted by the ensemble members for the current instance that has been classified

Block Diagram



Functional Description

The previous diagram shows all major modules of the so_ip_edte_un_p core that are described here in more detail.

Architecture of the so_ip_edte_un_p core consists of *N* identical modules, *N* being the number of decision trees that make up the ensemble. Each of these modules implements one decision tree from the ensemble. Decision trees that make up the ensemble can be of different sizes (in terms of nodes), and depths. Even more, decision trees that make up the ensemble don't have to be of the same type, some DTs can be trees with linear tests, while others can be trees with non-linear tests.

At regular interval, which depends on the depth of the deepest DT from the ensemble, classifications of the current instance become available at the output port of the core, *class_o*.

Using ensemble configuration interface, user can modify the structure of any DT from the ensemble directly in hardware, during the operation of the core. Trees can be removed from or added to the ensemble without re-running the synthesis and implementation steps.

DT Modules

Modules that implement DTs from the ensemble are based on an area efficient sequential architecture that requires minimum hardware resources while still having acceptable classification speed. Each pipeline stage corresponds to a level of the DT. Each DT module consists of three major sub-modules: attribute memory, instance position calculation module and a memory storing the relevant information about the nodes from the same DT level.

Verification Methods

Decision tree ensemble evaluation core was tested both using sophisticated verification environment and in dedicated hardware platform. Verification environment was used to extensively verify the so_ip_edte_un_p core's operation for different types and sizes of decision tree ensembles. After reaching all verification goals, IP core was next tested using dedicated hardware platform. Using this platform so_ip_edte_un_p core was implemented in FPGA and tested in real applications to estimate the performance of the core. The details about the verification methodology that was used and performance results during hardware testing can be obtained from So-Logic upon request.

Device Utilization & Performance

So_ip_edte_un_p core has a very regular structure that allows an easy estimation of required hardware resources and classification speed.

The size of required memory, number of multipliers and adders and throughput of the so_ip_edte_un_p core are shown in the following table and expressed in terms of: number of DTs in the ensemble (*N*), number of nodes of every DT from the ensemble (N_{dt_i}), depth of every DT from the ensemble (M_i), number of visited nodes in the DT during the instance classification (N_{nv_i}), number of problem attributes (*n*), number of classes (N_{cl}), number of bits for representation of attributes values (N_a) and coefficients values (N_c) and clock period (*CP*). All reported values are calculated under the assumption that ensembles are built from oblique DTs.

Required Memory (bits)	Multipliers	Adders	Throughput (number of classified instances per second)
$N \cdot n \, x \, N_a$, $\left(\sum_{i=1}^{N}(n+1)\cdot N_{dt_i}\right) x \, N_c$, $\sum_{i=1}^{N}\left(2\cdot N_{dt_i} \, x \left\lceil ld\left(N_{dt_i}\right)\right\rceil\right)$, $\left(2\cdot\sum_{i=1}^{N}N_{dt_i}\right) x \left\lceil ld\left(N_{cl}\right)\right\rceil$	N	N	$\frac{1}{\max_{i=1}^{N}\left\{N_{nv_i}\right\}\cdot(n+1)\cdot CP}$

Contact Information

So-Logic
Lustkandlgasse 52/22
A-1090 Vienna
Austria/Europe
Phone: +43-1-3157777-11
Fax: +43-1-3157777-44
E-Mail: ip_edte_un_p@so-logic.net
URL: http://www.so-logic.net

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/10/2009	1.0	Initial release.