so_ip_idt

Product Specification

Decision Tree Inference Core

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.

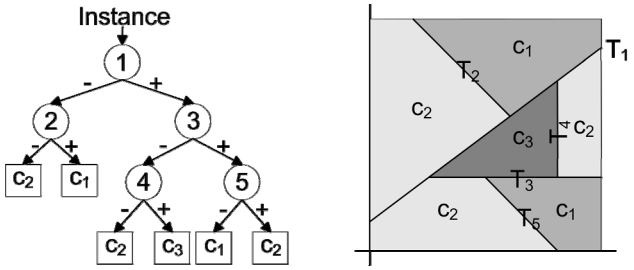
There are many different predictive models (classifiers) in machine learning, including artificial neural networks (ANNs), decision trees (DTs) and recently introduced support vector machines (SVMs).

Decision Trees are rooted tree structures, with leaves representing classifications and nodes representing tests of features that lead to those classifications.

Although not as popular as ANN classifiers, DTs have several important advantages when compared with ANNs. Although DTs can be less accurate than ANNs, DT learning algorithms are much faster, have smaller number of free parameters to be fine-tuned and require little data preparation, in comparison with ANN learning algorithms. In addition to that, apart from being simple to understand and interpret, DTs could be directly converted into the set of if-then rules, which is not the case with ANNs. DTs are also robust and scale well with large data sets.

In DT learning, target function is represented by a decision tree of finite depth. Every node of the tree specifies a test involving one or more attributes of the function to be learned, and every branch descending from a node matches one of the possible outcomes of the test in the considered node. To classify an instance we perform a sequence of tests associated to the sequence of nodes, starting with the root node and terminating with a leaf node. If we allow numerical attributes only, the resulting DT will be binary. The majority of algorithms allow only numerical attributes.

This concept is illustrated in the Figure below.



Basic structure of a decision tree with the possible classification regions in the case of two attribute classification problem.

In the above figure a structure of a typical DT is shown. In this example a classification problem with two attributes (that correspond to the *x* and *y* axes in the graph on the right) and three possible classes (c_1, c_2, c_3) is assumed. Two dimensional attribute space is divided into classification regions by the DT using five linear tests (that correspond to the linear segments marked as T_1, …, T_5 on the graph). Each of these test is located in one of the DT nodes (marked with numbers 1, …, 5 in the DT) shown on the figure. Although in this example linear test have been used, DTs, in general, can also use any form of nonlinear tests to divide the classification space.

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

So_ip_idt core can be used create a decision tree directly in hardware. It can create DTs with univarite, multivariate and non-linear tests. Creating DTs directly in hardware results in the significant increase of DT inference speed, compared with the traditional software-based approach.

So_ip_idt core implements a proprietary DT inference algorithm based on the evolutionary algorithms, developed at So-Logic, that enables quick DT inference with very favorable DT characteristics (measured in terms of inferred DT size and accuracy) when compared with the existing popular DT inference algorithms (C5, CART, etc.).

After the inference process is complete, complete structural information about the created DT is transferred through the output port. This information can be easily transferred to some of the So-Logic's DT evaluation cores enabling hardware implementation of the inferred DT. By combining these two cores a hardware-based adaptive learning systems can be easily designed.

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

The so_ip_idt 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

- Enables DT creation directly in hardware
- Speedup of inference time of over 1000x compared to the traditional software approach
- Supports classification problems that are defined by numerical attributes only
- DTs with univariate or multivariate tests are supported
- DTs with nonlinear tests are supported
- 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

- Can be easily integrated with some of the So-Logic's DT evaluation cores to create hardware-based adaptive learning systems

Applications

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

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

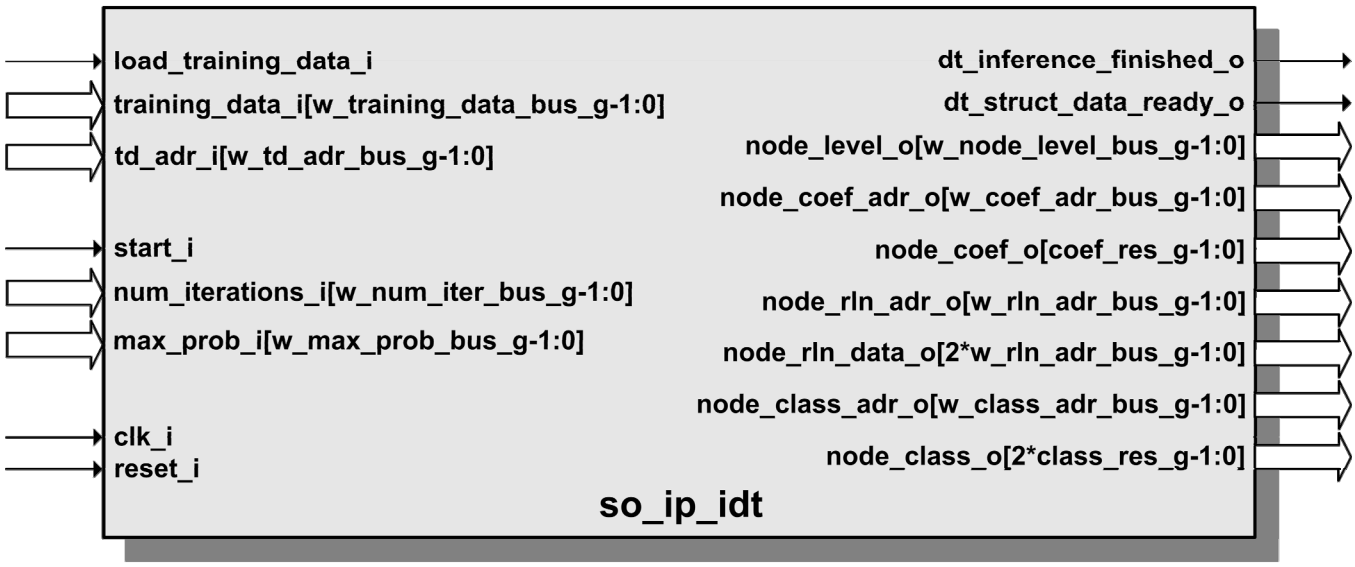
Netlist License

- 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

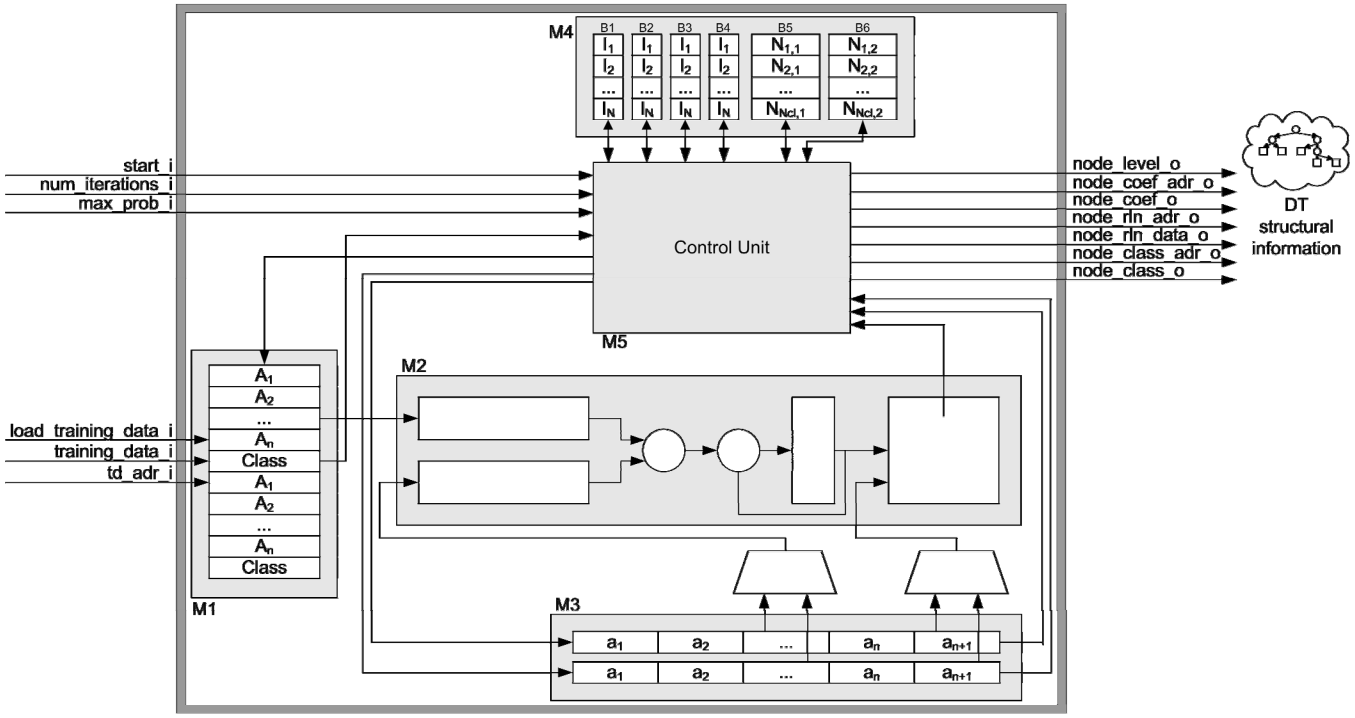


Pin Description

Name	Signal Direction	Description
Global Clocks and Reset Ports		
clk_i	Input	Main clock input
reset_i	Input	Main reset
Training Set Interface		
load_training_data_i	Input	Signal that is used to load the new training set data (attribute values and class membership information for the current instance) into the training set memory
training_data_i[w_training_data_bus_g-1:0]	Input	Data bus that is used to transfer the information (attribute values or class membership value) about the instance that is being transferred to the training set memory
td_adr_i[w_td_adr_bus_g-1:0]	Input	Address bus that is used to specify the training set memory location where the training data should be stored

Control Interface		
start_i	Input	Indication that the training set is loaded into training set memory and the process of DT inference can commence
num_iterations_i[w_num_iter_bus_g-1:0]	Input	Number of iterations of the DT optimization algorithm that should be performed during the optimization of every DT node
max_prob_i[w_max_prob_bus_g-1:0]	Input	Maximum mutation probability that should be used during the node optimization procedure
Inferred DT Structural Information Interface		
dt_inference_finished_o	Output	Indication that the inference of the DT is completed
dt_struct_data_ready_o	Output	Indication that new structural information about the inferred DT is ready to be transmitted
node_level_o[w_node_level_bus_g-1:0]	Output	Information about the DT level in which is the current node located
node_coef_adr_o[w_coef_adr_bus_g-1:0]	Output	Address of the location in the node coefficient memory where the current coefficient value should be stored
node_coef_o[coef_res_g-1:0]	Output	Coefficient value that should be stored in the node coefficient memory
node_rln_adr_o[w_rln_adr_bus_g-1:0]	Output	Address of the location in the next-node memory where the information about the current node's successors should be stored
node_rln_data_o[2*w_rln_adr_bus_g-1:0]	Output	Information about the locations of the two successors of the current DT node that should be stored in the next-node memory
node_class_adr_o[w_class_adr_bus_g-1:0]	Output	Address of the location in the class memory where the information about the output class membership for the current node should be stored
node_class_o[2*class_res_g-1:0]	Output	Output class membership for the current node that should be stored in the class memory

Block Diagram



Functional Description

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

Architecture of the so_ip_idt core consists of five major modules: training set instance memory (M1), instance evaluation module (M2), chromosome memory (M3), instance index and count memories (M4) and control unit (M5).

Module M1 holds the training set instances. Each instance is represented by the vector of *n* attribute values followed by the class membership value. Size of the memory depends on the size of the training set that is used to create a DT.

Module M2 calculates the instance position relative to the candidate hyperplane. This information is needed to calculate the value of the cost function during the hyperplane position optimization.

Module M3 is used to store both the encoding of the best so-far hyperplane and the encoding of the current candidate hyperplane (that is to store two chromosomes).

Module M4 consists of six memories, four of which are used to store the instance indexes associated with the nodes in the current and the next DT level. To store the information about the number of instances from every class that are located above and below the candidate hyperplane, two count memories are also needed. This information is needed to calculate the value of the cost function during the evolutionary optimization of the hyperplane position.

Module M5 is the control unit responsible for the correct operation of the entire system. Control unit performs the steps defined in the proprietary DT inference algorithm to build the DT, and controls the overall operation of the core.

Using the control interface user can modify the parameters of the DT inference algorithm and by doing so alter the overall DT inference process.

At regular interval, which depends on the current settings of the DT inference algorithm parameters, structural information about the next DT node that has been optimized is transferred to the user through the DT structural information interface.

Verification Methods

Decision tree inference core was tested both using sophisticated verification environment and in dedicated hardware platform. Verification environment was used to extensively verify the so_ip_idt core's operation for different types and sizes of decision trees. After reaching all verification goals, IP core was next tested using dedicated hardware platform. Using this platform so_ip_idt 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

In order to get a better estimation about the required resources in actual applications, following table presents the so_ip_idt core implementation results for selected datasets obtained from the UC Irvine Machine Learning Repository. The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms.

UCI Dataset	Size of Training Dataset	Number of Problem Attributes	Number of Classes	Number of Slice LUTs	Number of BRAMs	Number of DPS48E BLocks
Australian Credit Approval	690	14	2	5603	3	9
Balance Scale	625	4	3	5228	3	9
Breast Cancer	264	9	2	4830	3	9
Breast Cancer Wisconsin	699	9	2	4672	3	9
Car Evaluation	1728	6	4	5609	3	9
Contraceptive Method Choice	1333	9	3	5569	3	9
German Credit	1000	24	2	5780	3	9
Glass Identification	214	9	6	4782	3	9
Cleveland Heart Disease	297	13	5	5592	3	9
Statlog Heart Disease	270	13	2	4983	3	9
Hepatitis	155	19	2	4981	3	9
Ionosphere	351	34	2	6023	3	9
Iris	150	4	3	4701	3	9
Liver Disorders	345	6	2	5414	3	9
Lymphography	148	18	4	5066	3	9
Page Blocks	5427	10	5	5916	3	9
Pima Indians Diabetes	768	8	2	5339	3	9
Sonar	208	60	2	6173	3	9
Thyroid Disease	215	5	3	4668	3	9

Tic-Tac-Toe Endgame	958	9	2	5324	3	9
Statlog Vehicle Silhouettes	846	18	4	5609	3	9
Vote	232	16	2	5060	3	9
Vowel Recognition	990	13	11	5479	3	9
Waveform21	5000	21	3	6079	3	9
Waveform40	5000	40	3	6583	3	9
Wisconsin Diagnostic Breast Cancer	569	30	2	6113	3	9
Wine	178	13	3	4928	3	9
Wisconsin Prognostic Breast Cancer	194	33	2	5356	3	9
Zoo	101	17	7	4918	3	9

Notes:

1. All core I/O signals are routed off chip
2. Results were obtained using Xilinx ISE 9.1.03i version of software, targeting Virtex-5 family FPGA devices
3. The synthesis results provided are for reference only. Please contact So-Logic for estimates for your particular application.

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_idt@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.