Product Specification

so_ip_edt_un

Decision Tree Evaluation Core – Serial 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.

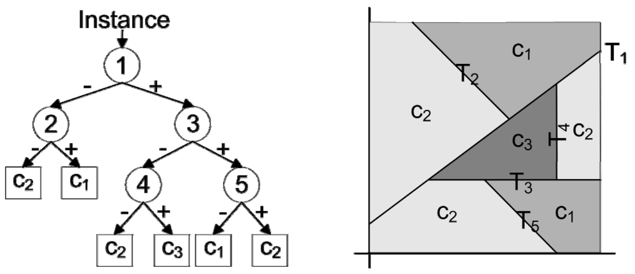
A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (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).

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_edt_un core can be used to implement the decision tree with the previously defined structure directly in hardware. It uses a simple sequential architecture that allows the smallest possible DT hardware implementation.

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

The so_ip_edt_un 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 DTs with previously defined structure
- Uses simple sequential architecture that allows the smallest possible DT hardware implementation
- Supports classification problems that are defined by numerical attributes only
- DTs with univariate or multivariate tests are supported
- DTs with nonlinear tests are supported
- 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

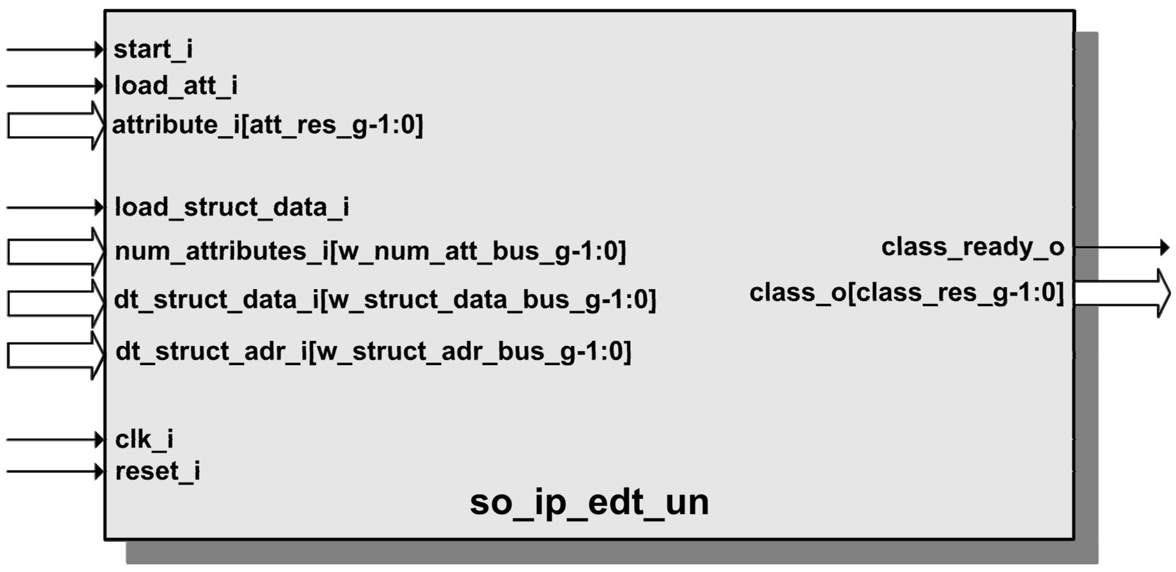
Netlist License

- 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
attribute_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[class_res_g-1:0]
so_ip_edt_un

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 decision tree 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 decision tree that will be implemented by the core
dt_struct_adr_i[w_struct_adr_bus_g-1:0]	Input	Address bus that is used to specify the memory location where the structural information about the decision tree that will be implemented by the core should be stored

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[class_res_g-1:0]	Output	Class value for the current instance that has been classified

Verification Methods

Decision tree evaluation core was tested both using sophisticated verification environment and in dedicated hardware platform. Verification environment was used to extensively verify the so_ip_edt_un core's operation for different types and sizes of DTs. After reaching all verification goals, IP core was next tested using dedicated hardware platform. Using this platform so_ip_edt_un 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_edt_un 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_edt_un core are shown in the following table and expressed in terms of: number of nodes of oblique DT (N_{dt}), depth of the tree (M), number of problem attributes (n), number of bits for the representation of attributes values (N_a), coefficients values (N_c) and clock cycle period (T_{clk}).

Required Memory (bits)	Multipliers	Adders	Throughput (number of classified instances per second)
$n \times N_a, (n+1) \cdot N_{dt} \times N_c, 2 \cdot N_{dt} \times \lceil ld(N_{dt}) \rceil, 2 \cdot N_{dt} \times \lceil ld(N_{classes}) \rceil$	1	1	$\frac{1}{num\_visited\_nodes \cdot (n+1) \cdot T_{clk}}$

In order to get a better estimation about the required resources and achievable clock speeds in actual applications, following table presents the DT 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	Slices	F_{max} (MHz)
Australian Credit Approval	122	197
Balance Scale	68	206
Breast Cancer Wisconsin	90	251
Breast Cancer	104	249
Car Evaluation	92	207
Contraceptive Method Choice	107	180
German Credit	146	227
Glass Identification	90	243
Cleveland Heart Disease	116	235
Statlog Heart Disease	101	231
Ionosphere	158	228
Liver Disorders	92	207
Lymphography	119	233
Page Blocks	107	186
Pima Indians Diabetes	84	202
Sonar	295	190
Tic-Tac-Toe Endgame	78	249
Statlog Vehicle Silhouettes	121	241
Vowel Recognition	94	191
Waveform21	111	183
Waveform40	238	182
Wisconsin Prognostic Breast Cancer	142	251
Zoo	88	246

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_edt_un@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.