Product Specification

so_ip_edt_smpl

Decision Tree Evaluation Core – Pipelined Archtiecture

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.

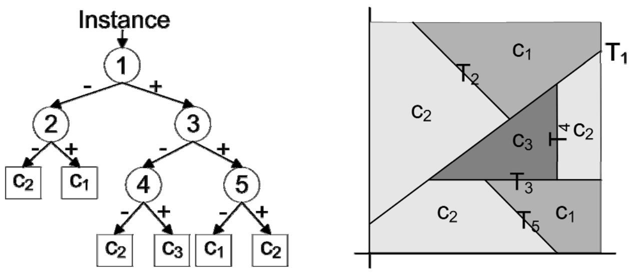
A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (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_smpl core can be used to implement the decision tree with the previously defined structure directly in hardware. It uses advanced pipelined architecture that allows the fastest possible classification speed.

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

The so_ip_edt_smpl 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 advanced pipelined architecture that allows the fastest possible classification speed
- 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

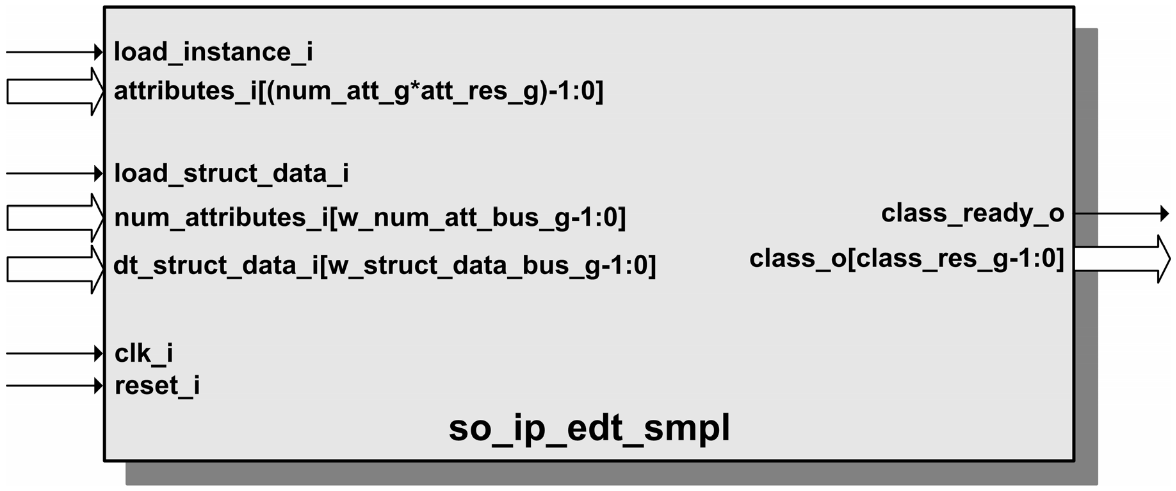
Netlist License

- 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

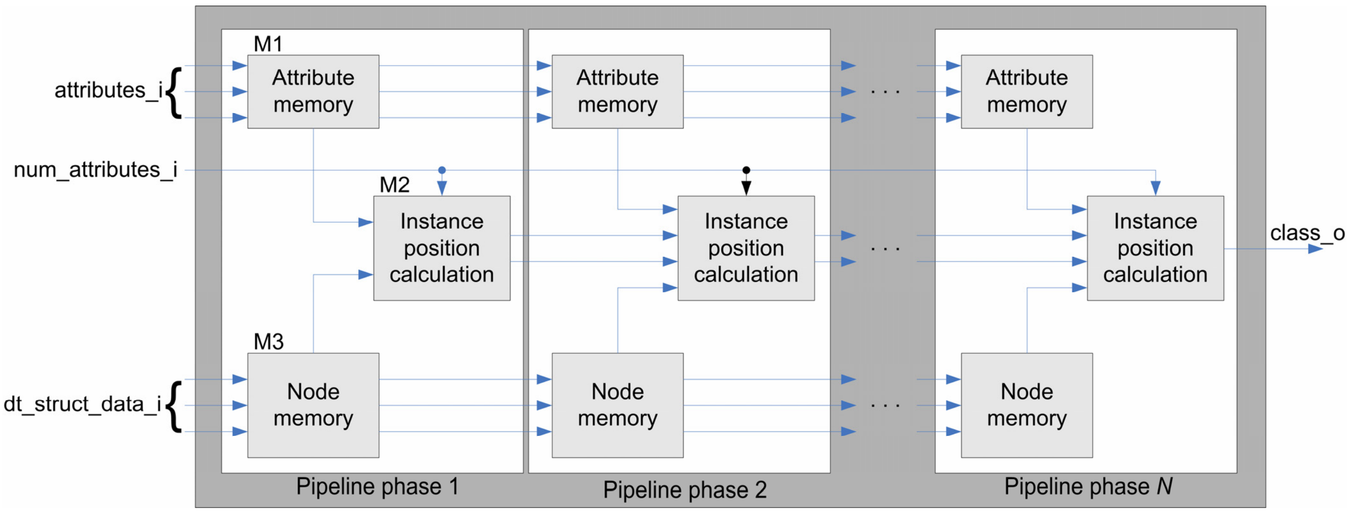
Symbol


load_instance_i
attributes_i[(num_att_g*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]
clk_i
reset_i
class_ready_o
class_o[class_res_g-1:0]
so_ip_edt_smpl

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
Input Instance Interface		
load_instance_i	Input	Indication that there is a new instance of the classification problem that needs to be classified
attributes_i[(num_att_g*att_res_g)-1:0]	Input	Values of the problem attributes for the 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

Block Diagram



Functional Description

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

Architecture of the so_ip_edt_smpl core consists of *M* pipeline stages, *M* being the depth of the realized DT. Each pipeline stage corresponds to a level of the DT. User can configure the number of pipeline stages in order to allow for the implementation of DTs of different depths. This configuration can be done only prior to synthesis and is not possible during the run-time.

Each pipeline stage consists of three major modules: attribute memory, instance position calculation module and a memory storing the relevant information about the nodes from the same DT level. Instance position calculation modules and attribute memories form two pipeline chains of length *M*. Each attribute memory from the memory chain is also connected to the corresponding instance position calculation module in the pipeline chain, as shown on figure above.

Attribute Memory Module

Attribute Memory Module (M1) is used to store the attribute values for the current instance. Its size is determined by the number of attributes, n, and the length of words, N_a, used to encode attributes.

Instance Position Calculation Module

Instance Position Calculation module (M2) calculates both the position of the instance relative to the hyperplane associated to the selected node from the current level, and the address of the node from the next level to be visited. Next node address is then transferred to the next pipeline stage.

When a leaf is reached, M2 module calculates the class value of the current instance and transfers it to the next pipeline stage.

M2 module calculates instance position relative to the hyperplane sequentially using one multiplier and one adder. An additional module compares accumulated value with the free running coefficient and determines the position of the current instance relative to the hyperplane. Instance position relative to the hyperplane is then used to select the correct node from the next level in the DT.

Node Memory Module

Node Memory Module (M3) stores information about the nodes from the associated level in the DT. It consists of three memory units. First memory unit stores the hyperplane coefficients. This memory is organised into slices. Every slice stores the coefficients for one hyperplane.

Second memory unit stores addresses of the child nodes for all nodes of the associated level. Address value of every leaf node can be set to an arbitrary value.

Third memory unit stores class values of child nodes for all nodes belonging to the associated level. If a successor of the considered node is not a leaf its class value should be set to zero, to indicate that current instance hasn't been classified yet.

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_smpl 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_smpl 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_smpl 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_smpl 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)
$M\cdot(n \times N_a), (n+1)\cdot N_{dt} \times N_c,$ $2\cdot N_{dt} \times \lceil ld(N_{dt}) \rceil, 2\cdot N_{dt} \times \lceil ld(N_{classes}) \rceil$	M	M	$\frac{1}{(n+1)\cdot T_{clk}}$

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	554	153
Balance Scale	426	201
Breast Cancer Wisconsin	314	200
Breast Cancer	485	200
Car Evaluation	539	157
Contraceptive Method Choice	1113	141
German Credit	937	147
Glass Identification	480	154
Cleveland Heart Disease	532	146
Statlog Heart Disease	294	200
Ionosphere	439	154
Liver Disorders	539	157
Lymphography	270	200
Page Blocks	915	144
Pima Indians Diabetes	609	148
Sonar	722	152
Tic-Tac-Toe Endgame	356	201
Statlog Vehicle Silhouettes	950	151
Vowel Recognition	857	155
Waveform21	1410	136
Waveform40	1851	137
Wisconsin Prognostic Breast Cancer	467	154
Zoo	361	200

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_smpl@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.