HIGH-LEVEL INTELLIGENCE-ORIENTED SIMULATION

Tudor Niculiu[1], Chouki Aktouf[2], Sorin Coţofană[3]

[1] TU Bucharest,EE Faculty, Iuliu Maniu Blvd. 1-3, 77202 Bucureşti 6, Romania
[2] Iut Valence, Ee Faculty, 51 Barthélémy de Laffemas, 26901valence, France
[3] Tu Delft, Ee Faculty, Mekelweg 4, 2600 GA Delft, The Netherlands

Abstract.–Intelligence is complementary to faith = (intuition, inspiration, imagination) and it needs (conscience, adaptability, intention). Conscience simulation demands transcending the present limits of computability to what we call simulability, by an intensive effort on extensive research to integrate essential mathematical and physical knowledge guided by philosophical goals. Applying "Divide et Impera et Intellige" to hierarchy types reveals their comprehensive constructive importance based on structural approach, symbolic meaning, object-oriented representation. Formalizing hierarchical descriptions, we created a theoretical kernel that can be used for self-organizing systems. A way to begin is hierarchical simulation. Conscience supposes sincerity; this can be implemented as testability at high-level.

1. ARGUMENT & CONCEPTS

After adaptability, conscience is the next part of intelligence that needs better understanding. It can be considered self-awareness of individual faith and intelligence, as well as of the relation to the local context (society) and to the global one (universe). To appear, conscience needs both correct self-knowledge as sincerity to the community. To simulate it hard-soft we aim convergence of concurrent research directions: formal abstraction by theory of hierarchy types, comprehensive construction of structural symbolic concepts, and high-level simulation of testability.

Competent design of hardware-software systems need the study of hierarchy types, the intelligent communication between different domains, the formal verification/ test. We extend the theory of hierarchy types to integrate communication properties as well as correctness and testability, to suit the behavioral specification of today's complex system design. The high-level approach of these problems permits the intervention of an intelligent agent to adapt techniques, models or methods to the particular design. The agent can be a designer, assisted by man-machine dialog interface, or an intelligent system. Testability measures the difficulty of test; it is used in this paper to emphasize the high-level strategy.

2. HIERARCHY TYPES

Coexistent interdependent hierarchies structure the universe of models for complex systems, e.g., hard/ soft ones. They belong to different hierarchy types, defined by abstraction levels, block structures, classes, symbolization and knowledge abstractions. Abstraction and hierarchy are semantic and syntactical aspects of a unique fundamental concept, the most powerful tool in systematic knowledge; hierarchy results formalizing abstraction. Different hierarchy types correspond to the abstraction kind they reflect (abstraction goal):

- Class hierarchy (↑concepts) ↔ virtual framework to represent any hierarchy, based on form-contents dichotomy, modularity, inheritance, polymorphism.
- Symbolization hierarchy (↑mathematics) ↔ stepwise formalism for all kind of types.
- Structure hierarchy (↑managing) ↔ recursive "Divide et Impera et Intellige".
- Construction hierarchy (↑simulation) ↔ simulation = design/verification framework of autonomous levels for different grades of abstraction.
- Knowledge hierarchy (↑theories) ↔ reflexive abstraction ("in a deeper sense"), aiming that each level has knowledge of its inferior levels, including itself.

3. INTELLIGENCE SIMULATION

Understanding and construction [Niculiu] have correspondent hierarchy types: their syntax relies on classes, their meaning on symbols and their use on modules. The hierarchy types can

0-7803-7440-1/02/$17.00 © 2002 IEEE

be formalized in the theory of categories. Constructive type theory permits formal specification and verification by generating an object satisfying the specification. Conscience is self-awareness of individual intelligence (adaptability, conscience, intention) and faith (intuition, inspiration, imagination), as well as of the relation to the local context (society) and to the global one (universe). The convergence process of evolution demands struggle against time, with structure as ally. Conscience needs, more than discrete recurrence, continuous feedback. Social and individual conscience are mostly divergent nowadays, i.e., we only performed "Divide et Impera", neglecting "et Intellige". It's high time to correct this.

The structure of the communication between heterogeneous parts of the simulated system and with its exterior should reflect the hierarchies of the simulation technique/ model/ method. Representation is a 1-to-1 mapping from the universe of systems to a hierarchical universe of models, so a representation can be inverted. Object-orientation suits for this as a model needs knowledge and manipulation, so it has two complementary parts: description and operation. We define a general hierarchical approach for complex simulation, applying it to handle communication between different domains implied by hard-soft systems. It combines dynamic objects handled in software with parallel activities realized in hardware. A main constraint for the simulation is testability. Design-for-testability techniques applied to different models assisted by specific methods increase the fault coverage and reduce the test generation time. Modification of the system's specification to improve testability performed at higher levels of the design hierarchy reduces the complexity of their generation/ application. We propose a behavioral adaptable DFT technique. Behavior of the complex system under design is specified initially in a high-level description language, representing the highest level of the construction hierarchy. This way we also contribute to hard-soft DFT

4. DESIGN FOR TESTABILITY

DFT must suit the behavioral specification of today's complex system design. Referring to high-level synthesis, DFT can operate before, while or after it. The first choice permits the intervention of an intelligent agent for adapting the DFT technique, model or method to the particular design. We call it behavioral adaptable DFT. It improves the testability, measured with adequate methods. This is done direct on the behavioral specification or aided by special representations, which permit return to the behavioral description after improving the testability of the system to be designed. The second choice is synthesis for testability, while the third could be called - register-transfer level - DFT. This paper concentrates on an improved structural BADFT.

Memory Elements Description

Memory elements - registers (arrayed FFs)/ flip-flops/ latches (clock-less FFs) - are represented in behavioral descriptions by variables or signals. Variables are description objects local to processes/ subprograms, used to store intermediate values between sequential statements, characterized by free assignment (exception: global variables). Signals are permanent description objects to link concurrent elements: components/ processes/ concurrent assignments, demanding synchronized assignment, declared locally - within architecture, block or other declarative region, or globally - in extended package. In the context of a process synchronized by a clock signal, in a behavioral description, signals implicated in simple/ multiple signal assignment generate memory during synthesis. Instances of this rule are:

- Multiple synchronization points, i.e., several wait statements with identical synchronization conditions) infer memory elements. This allows describing Finite State Machines without declaring the state variables. Several wait statements with different synchronization conditions are not yet able for synthesis.
- If a signal is read, its value used, before being assigned, it infers memory. A particular case of this rule is a conditional statement that does not affect a signal in every of its branches.

An analog rule can be formulated for variables:

- Inside a process, a variable that must hold values between iterations of the process implies memory elements, i.e., a variable

which is set but not used between synchronization statements infers memory. This happens also for variables being read before assigned.

The context is not restrictive, as all concurrent statements are equivalent to processes (excepting direct/ component instance creation). For called subprograms, the rules of memory inference can be deduced directly: pure functions (without side effects) do not infer memory elements - while procedures, because of their side effects, do.

DFT Techniques

The most used DFT techniques are Scanning, Built-In Self-Test and Test Point Insertion. They can be applied at the different levels of the design hierarchy (behavior, RTL, logic) and can be combined. We begin with PS applied to the autonomous blocks of the behavioral (V)HDL specification, but the other techniques can contribute to improve the testability of the behavioral specification or the way to this goal. All types of hierarchy are implied in this approach: class-object framework, symbolization degree and module structure, as well as design/ knowledge approach.

High-Level Improved Partial Scan

S-graph = (FFs, combinationalPaths) is a weighted directed graph; testability is related to cycles and sequential depth, but can also be represented, partially or totally, by the node weights. The S-graph is used as intermediate format of high-level synthesis in literature, but can be managed as model of the behavioral description. From structural point of view, feedback cycles among registers are mainly responsible for low testability, as their total length influences exponentially the complexity of test generation. Only the maximal strongly connected sub-graphs must be considered when reducing the cycle-size-sum, because strong connected is dual to a-cyclic. The next structural attribute that testability depends on (linearly) is sequential depth. The algorithm to eliminate cycles is minimum Feedback Vertex Set. It consists in finding the smallest set of (weighted) nodes whose removal results in a directed a-cyclic graph (DAG). Self-loops and other loop-structures that do not pose test problems can be excepted, e.g., by absorbing them in nodes. This algorithm is NP-complete; different solutions to reduce the complexity must be applied to the graph: partitioning and FVS-preserving transformations. Formal or heuristic testability measures (to be improved) can be defined on the structural model. Examples of these measures are: probabilistic metrics for signals/ variables (randomness, transparency) operated within a Markov chain model, respectively, with composition rules. The weights of the S-graph retain the cost/ gain to scan a node, e.g., gain $\Leftarrow$ (number $\times$ length of cycles that contain the node, sequential depth reduction).

G=(V,A), directed graph; FVS: G$\rightarrow$min w(V'),
G' = (V - V', A - I(V') - O(V')) a-cyclic.
G' can be replaced by: G'' = (V, A-I(V')).
As FVS is NP-complete, heuristics are used to sort the vertices of the S-graph and to formulate appropriate continuation criteria. Even more, they can improve testability by transformations that preserve functionality as well as timing of the specification. An alternative to heuristics is using graph theoretic algorithms to reduce the graph before FVS.

We enhanced the structural technique by textual methods used for test point insertion: Syntactical analysis on the behavioral description is used to measure the testability, determining different aspects that reduce it for the specified module [Larsson]. Hard-to-test parts, represented by variables/ signals, are indicated, so register selection for PS can be guided. A HDL behavioral specification is used directly to measure and to improve testability. Different aspects that contribute to low testability of the hardware corresponding to objects of the system's/ component's specification of behavior are combined and compatibilized by algebraic operations to reflect the testability of a signal/ variable. Experimented textual testability measures are: restricted value range of variable/ signal in a description statement, non-uniform distribution caused by an operation on the values of variables/ signals, reduced accessibility of instructions caused by conditions.

Behavioral Enhanced S-Graph for BADFT

We present an example of the approach. An

intelligent interface assures the translation, in both senses, from B-VHDL description to a structural representation of the required behavior that guides the PS (BS-graph). A knowledge base assists generating the weighted directed graph (FFs, combinational paths) and to return to text the differences caused by transformation for testability improvement. The rules of correspondence between description object assignments and registers, as well as rules to translate the data flow of behavioral specification to weighted arcs in the graph counterpart and to combine different testability measures in node weights guide the first step, while incrementing rules for VHDL description solve the last one. Testability for behavioral description based on data objects and data functions results, as usual, from controllability and observability. If the test vectors applied on the input of a description module form a complete vector set, the module is controllable. A complete vector set causes every element of the module, on the current description level, to traverse its whole-defined value range for objects and input range for functions. If the values of all elements of the module can be determined of its response to a complete vector set, it is observable. The FS case is solved [Fleury]. First, memory elements are located among the HDL-objects; each is related to a number of FFs depending on the type of the considered variable/ signal.. Then, a scan chain is built that contains all design FFs; this implies following steps: the number n(x) of FFs for each behavioral object x is determined; the behavioral VHDL description of scan is added for each located memory element; the n local scan chains corresponding to the memory elements contained in the array mem are connected. The difference to FS needed by PS for the return translation is a pointing scheme for the scanned objects among signals/ variables of the behavioral specification. This can be managed by an adequate data structure or by using access types or attributes in VHDL. What remains to add is the partial register selection. FFs are selected for scan, but when a register is used in parallel, it candidates entirely for scan. For PS, the variables/ signals inferring memory are sorted to select incrementally the scan elements that will be eventually mapped to the scan register. The selection is integrated, applying together, for every iteration of the graph's method, the function to measure and that to improve testability. We combine structural and textual DFT methods. A supplementary analysis of node relationship can reduce the number of iterations affected to testability improvement: $\Leftarrow$ grouping hard-to-test parts according to their interdependencies, so that, at each iteration, parts of different groups can be improved together without affecting each other. Nodes are grouped together, if they share the shortest justification path from primary input or shortest propagation path to primary output. The testability values for all variables/ signals are ranged and the objects sorted in increasing order of their testability. After selecting the hardest-to-test object (represented as node in the BS-graph model) at the current iteration, the sorted list of nodes is traversed grouping the nodes on the shortest paths of the selected object to I/O together, while the rest of the nodes are selected for scan, in decreasing hardness-to-test order. The idea can be extended to cycle reduction.

V. CONCLUSIONS

A necessary condition to find the right way to intelligent simulation is hierarchical hardware-software analog-digital intelligence simulation, which has to be assisted by high-level adaptable design for testability. Applying "Divide et Impera et Intellige" to hierarchy types (structural approach, symbolic meaning, object-oriented representation) comprehensive construction is revealed. Further, hierarchical descriptions in the formalism of categories guide us to a kernel for self-organizing systems demanding for self-awareness.

REFERENCES

1. H. Fleury, C. Aktouf, C. Robach: A. Practical Technique for Behavioral Scan Insertion, International Test Synthesis Workshop, 1999.
2. E. Larsson: High-Level Testability Analysis & Enhancement Techniques, Ph.D. Thesis, Linköping University, 1998.
3. T. Niculiu, S. Cotofana: Hierarchical Intelligent Mixed Simulation, European Simulation Multiconference, 2002.