A High-level Synthesis of Embedded Composite Systems for Environmental Applications

Cesare Alippi, Vincenzo Piuri, Fabio Scotti

***Abstract* — The paper presents a methodology for supporting a high-level synthesis of embedded composite systems (solutions comprising soft-computing and traditional algorithms). The methodology is particularly suitable for those environmental applications characterised by partly unspecified or computationally unacceptable solutions. The system characterisation at the behavioural level is given by providing a set of (Input, Output) examples, oriented computational graphs and black-box modules. A set of application requirements such as accuracy, computational complexity and latency fulfil the system specifications. The methodology outcome is a list of composite systems ranked according to performance and constraints satisfaction; from such a list the designer selects the most suitable solution for the specific application.**

I. INTRODUCTION

Recent developments and expectations in the field of environmental applications are strongly modifying the structure of conventional monitoring systems.

In a typical processing chain signals coming from the environment are transduced by sensors and –possibly- converted in a digital format by means of A/D converters. Data are then collected and processed to provide the operators with the requested environmental information. Frequently, data are reported to the operator with user-friendly interfaces enhanced by multimedial features (e.g., touch screen, dedicated buttons, sound and visual alarms)[1].

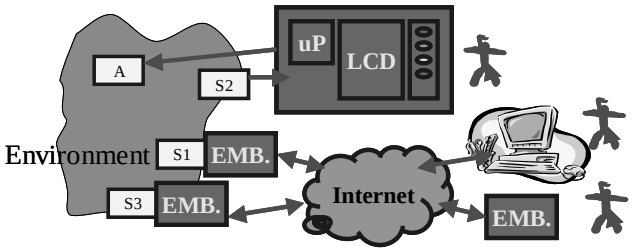

Fig.1. Embedded systems for environmental applications

This trend is characterised by scenarios in which remote sensorial data are sent to the final processing/reporting system via Internet or dedicated LAN/WAN. To reduce communication bandwidth data are often pre-processed on site to extract the relevant features; in other cases signals are compress before transmission. More and more often such tasks are carried out by embedded systems [2,3]. A common situation is depicted in fig. 1. There, sensors (S_i) and actuators (A_i) interact with the environment by means of embedded systems which interface with network/Internet connections and humans.

An embedded monitoring system provides appealing features such as flexibility (Internet communication and remote control support), user friendly interfaces (the user interface is simplified once compared with that provided by a typical PC station), small, cheap, and low-power devices being the electronic system tailored to the specific application [4]. These positive elements are supporting and leading the embedded systems market where it is expected the embedded system sales to overpass the PC ones in 2002 [4, 5].

The selection of the “best” algorithm, i.e., the most suitable for the specific embedded application requires a careful exploration of the solution space which, in turn, involves study of accuracy, power consumption and real-time computation issues just to name some of the critical aspects. For an effective design all the available "a priori" information must be collected and used to guide the composite system design, namely the partition of the solution in traditional and soft computing techniques (neural networks, fuzzy logic, etc.) [6,7].

Interestingly enough, it has been proven in several applications [9] that a composite system can solve the application and satisfy the application requirements when the solely use of traditional or soft computing techniques can not.

In general, this situation arises when the solution of the application is too complex or not completely specified.

For instance, this happens in environmental monitoring applications where sensorial information are pre-processed (filtered and amplified), a set of feature are extracted and then used by a classifier to identify the occurrence of critical conditions. Pollution alert stations and alarms systems in cars and domotic applications are examples in this direction. In general, the features extraction phase is generally carried out with traditional algorithms and the subsequent classifier design is most of times solved by considering soft computing techniques.

Alippi and Scotti are with the Dip. Elettronica e Informazione, Politecnico di Milano, Milano, Italy. Tel: +390223993550, Fax:+390223993411, email: {alippi,scotti}@elet.polmi.it, Piuri is with the Department of Information Technologies, University of Milan, Crema, Italy, tel:+390373898242, fax: +390373898253, email: piuri@dti.unimi.it

There are several composite system solving an application starting from traditional and soft computing modules. In fact, once identified the computational flow solving the application we can generate different mixed solutions. Surely, constraints and the complexity nature of the design space make the analysis extremely difficult so that only trial and error approaches have been considered by now.
The paper proposes a methodology to support a semi-automatic design of the most suitable composite system solution.
The structure of the paper is as follows. Section II describes the current state of the art in the high level design and identifies the abstraction level where a composite synthesis design can be inserted. The composite system design methodology is then described and detailed in section III.

II. HIGH-LEVEL SYSTEM DESIGN

In the last decades we have been experiencing a new trend in the complex system design philosophy driven by the necessity of rapid prototyping and a reduced time to market.
To cope with such challenging aspects it has been necessary to abstract the design methodology from the very low levels (in which physical realisation details are taken into account) to higher levels where sophisticated analyses are introduced to characterise the application performance and constraints.
A high-level system design methodology must be flexible to support an automatic/semi-automatic generation of the solution, yet powerful enough to deal with a large class of applications.
The actual state of the art in high-level system design can be summarised in the abstraction/applicability plot of fig. 2. The figure shows in grey the existing software layers and with a darker colour the expected composite-based ones. The horizontal axis, defined as *range of applicability*, is a qualitative evaluation of the easiness in using the software to design embedded systems with different families of processor and co-processors.

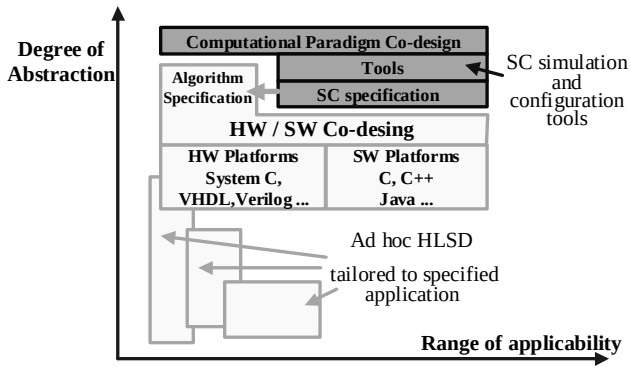

Fig.2. Representation of the high-level system design state of the art (SC: Soft Computing, HLSD: High Level System Design).

Existing software is often tailored to specified applications and presents a low range of applicability. More complex and flexible tools are represented by the so-called hardware platforms [4], i.e., software for designing, simulating and test basic modules of a fixed family of architectures such as I/O subsystem, memories, programmable cores, etc. Similarly, software platforms allow for abstracting the hardware, the programmable cores and the memory subsystem by means of a real-time operating system, the I/O subsystem with the Device Driver, etc. [4,8].

The HW/SW co-design layer can also be considered during the system design phase. The layer is composed of tools partitioning the computational complexity of the specified algorithm between software (a portion of the computation executed on a general purpose device, i.e., machine code) and dedicated hardware (computation implemented directly as dedicated HW), optimising suitable figures of merit defined by the designer.
When composite systems are envisaged an upper layer must be generated to solve the given application. Basically, it is necessary to partition the application solution described by means of traditional algorithms from the ones requiring soft computing techniques.
The main difference between traditional and soft computing modules is that the latter requires a parameter configuration and a validation phase before being completely specified. In other words a training/tuning procedure is necessary to define the behaviour of soft computing/unspecified modules.
Of course, when all modules of the composite system are completely specified the computational flow associated with the application solution is specified as well and, consequently, can be passed to the HW/SW co-design layer for further optimisation.

III. THE PROPOSED METHODOLOGY FOR COMPOSITE SYSTEM DESIGN

The high-level design of composite systems can be interpreted as a "codesign" activity between traditional and soft computing algorithms. Up to our knowledge, a trial and error approach is generally considered nowadays to configure a composite system for a given application since no automatic design tools are available.
Conversely, an effective high-level design tool must help and support the designer in

- carrying out the design space exploration automatically
- allowing for the reuse of existing modules
- better exploring the design space by also exploiting available information and experienced gained in similar applications
- monitoring the computational complexity, the latency and other features of the solution at the very high-level.

The methodology suggested in this paper aims at integrating all the above positive requirements within a

homogeneous framework hence granting generality, flexibility and design understanding.
The structure of the high-level design methodology for composite systems is presented in fig. 3.
The basic task carried out by the semi-automatic design system is generation of a set of composite systems -or candidate solutions- and test whether constrains are satisfied or not. Of course, candidate selection must be smart enough to guide the search exploration in the composite system space towards the identification of the most promising solutions. This can be accomplished by considering different models for the composite system (stored in a models database) and exploiting experience gained in previous applications (such information is present in a knowledge database).

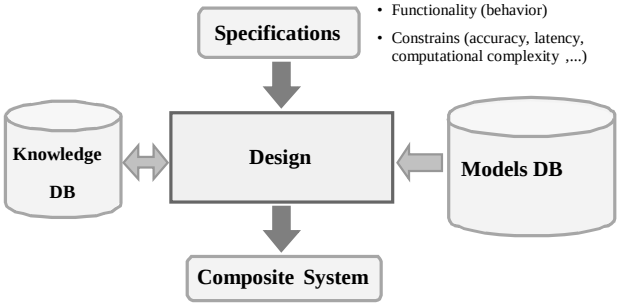

Fig.3. Overview of the high-level composite system

The different activities to be accomplished for an effective composite system design can be summarised in six steps:

1. Module decomposition
2. Model family selection
3. Search space reduction
4. I/O pairs decomposition
5. Modules tuning
6. Composite system ranking

III.1 Module decomposition step

The module decomposition step accounts for the identification of a composite system structure for the application solution.
Initially, the solution structure can be seen as a single black box whose composing parts –or functionalities- must be further detailed. In fact, we could envision a situation in which the application can be solved with a single computational module (either traditional or soft computing) or with a more complex situation in which the application solution requires several sub-modules (again, each of which is either traditional or soft computing based) suitably connected.
We can therefore consider a recurrent procedure for the composite system subdivision in modules in which a single module can be further subdivided in sub-modules having –possibly- different nature.
In this iterative process we have to end up-with some atomic units for the modules or Simple Modules (SM). Each SM can be either traditional or a soft computing based module.
The recurrent decomposition of a module in sub-modules is depicted in fig. 4. We have that the generic module M can be decompose with a serial or parallel configuration into two sub-modules M, each of which can be either a single module SM or a composite model CM (which again comprises a module M).

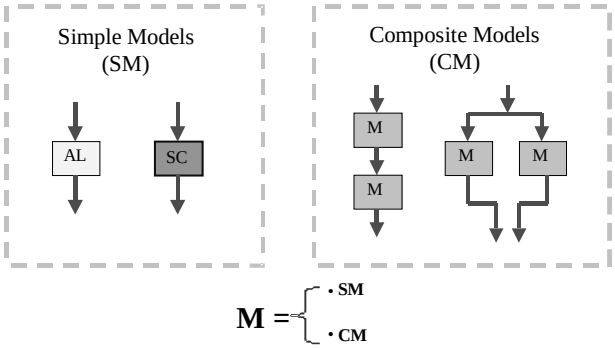

Fig.4. Model of computation for Composite models: the definition is recurrent

It is very interesting to note that different compositions of the system provide different candidate solutions to be tested for constraints satisfaction and that each candidate solution has its own degree of constraints satisfaction.
For instance we could identify, within a composite system framework, different design solutions having the same accuracy but different cost (in some arbitrary units) and latency. Typically, the hardware cost/latency trade-off is obtainable by changing the hardware architecture (fully hardware and ASICs, micro-controller and coprocessor, PC plus software etc.) or changing the technology [8].

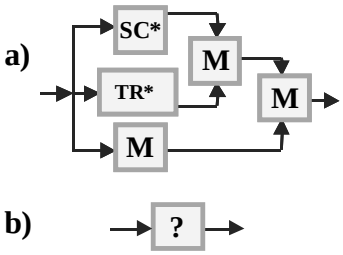

Fig. 5. Two possible inputs of the methodology: partially specified oriented graph (a) and black-box (b)

At each step of the refinement of the composite system structure we end up with an oriented computational graph similar to that of fig. 5. We labelled with a * those modules whose TR/SC nature and computation have been identified. Such modules are already available, e.g., they come from a module reuse library.
The topological decomposer by receiving an incomplete computational graph generates a new refined candidate composite system by exploiting some decomposition rules present in an actions database.

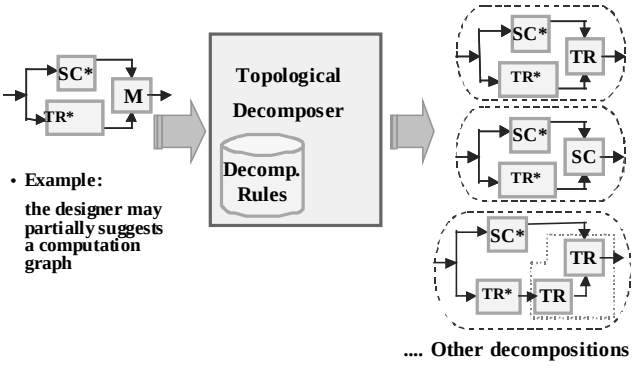

Fig. 6. Example of topological decomposition for a partially specified graph

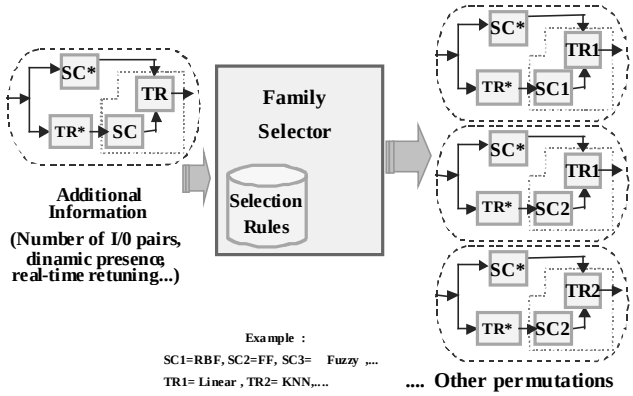

Fig.7. Family selection. The model database contains a limited number of families

Fig. 6 shows an example in which the topological decomposer, by starting from module decomposition, suggests three final composite systems. In particular, module M has been decomposed and labelled as traditional, soft computing and the series of two traditional modules; of course other decompositions can be envisaged.

If the partially specified graph has more than one M, the previous operation can be repeated for each M according to the series/parallel decomposition of fig. 4 to generate new composite models.

We have to note that, in general, no information is created during the decomposition process.

Nevertheless, the topological decomposer can integrate some constraints at the very high level and, hence, discard or prune proliferation of unfeasible composite systems. Composite systems not satisfy the requirements can be immediately discarded.

III.2 Model family selection

The second step of the methodology is model family selection, which accounts for associating a specific model family to the unspecified traditional module or the soft-computing one.

The model database contains a limited number of families, which can be used for module labelling. For instance, we can associate the soft-computing modules with a list of families: SC_1=RBF, SC_2=FF, SC_3=Fuzzy, SC_4=Neuro-Fuzzy (RBF stands for Radial Basis Functions, FF for FeedForward neural networks). All the assignments satisfying the rules contained in the selection rules DataBase can be considered to generate a composite system having a fully specified computation (the structure of the computation is fixed, the model families must then be trained to fix the computation). Fig. 7 shows three outputs for a serial composite system composed of two modules.

Since generation of all possible permutations for the model families is associated with a large number of composite systems to be subsequently tested, the family selection rules aim at strongly limiting proliferation of a priori uninteresting solutions.

Selection rules have also to identify the most promising composite systems based on high level information about the nature of the application/module such as the presence of a dynamic behaviour, the number of I/O pairs available, real-time constrains, etc.

In fact, we have that different model families have different topological/computational complexity and potentialities in modelling static/dynamic behaviours.

Directives and guidelines exploiting a priori beliefs coming from the designer greatly help this procedure by focusing the attention on particular model families.

For example, and referring to fig. 6, if the designer marks a module as a “classifier”, the selection rules extract from the lists only those model families able to implement a classifier. A Radial Basis Function Neural Network or a generic Feed Forward Neural Network can be extracted from the SC_i list and a Nearest Neighbour Classifier from the TR_i one.

III.3 Search space reduction

In the search space reduction phase the composite systems modules are well defined in terms of nature and model family. For example a module can be marked as a two-layered feed-forward neural network with *n* and *m* hidden units, an ARMAX model with *j* delays and a low pass filter of the *t*-th order but the internal parameters of the models *n*, *m*, *j*, *t* are not yet fixed.

The search space reduction aims at pruning the model hierarchy by considering only a subset of such models. There is no need to consider a model family with an effective number of parameters larger than the number of available training examples. Additional application requirements can be further envisaged to reduce the feasible interval for the structural parameters of the models. In the case presented in fig. 8 latency and computational complexity requirements can be used to prune the composite system space. If the training set is composed of 20 examples we can surely limit the number of hidden neurons to the [1:20] range (in reality one should consider the effective number of degrees of freedom used by the model but such an estimate requires a trained model). Computational complexity and latency could also limit the

number of units for the *k*-nearest neighbour classifier. We could discover that system complexity limits the *k* to belong to the [1:5] range.
Search space pruning depends on the figures accepted and a priori information about the nature of the supporting executing architecture.

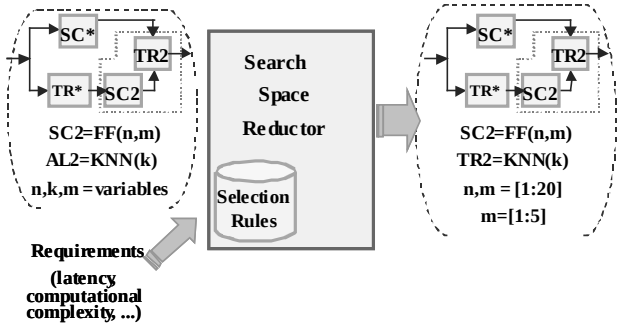

Fig.8. Search space reduction: the structural parameter ranges (n, m, k, etc.) are fixed

The feasible range for structural parameters are identified by what we name *mapping functions*. These functions receive as input a module and its description as well as the tolerable values for the constraints and return the feasible values for parameters.
Mapping functions can be considered to limit the search space only for those requirements directly related to the nature and structure of the chosen model family. Examples of such constraints are latency, number of training data and computational complexity.
The accuracy (or performance) requirement cannot be taken into account by the mapping functions since performance are available only at the lower layers of the composite system design when each parameters/weights of the model are suitably tuned (trained).

Once this step has been completed, each model is completely bounded from the topological point of view. The specific model to be associated with the module must be finally identified by fixing its free parameters (e.g., the weights of the hidden neurons or the coefficients of an ARMAX model).
Before carrying out the training/ system tuning phase we need the training examples to be available at the output of the module to be configured. In fact, we have the example pairs available at system output and not at the module output. A propagation of the input/output examples to the module output is therefore necessary: the task is carried out by the input/output pair decomposition.

III.4 Input/output pair decomposition

The input/output pair decomposition step aims as propagate the system input examples to the module inputs and back-propagate the associated system outputs to the module output.
This step is not always possible for a generic decomposition system due to the mathematical impossibility of propagating the system outputs to the module outputs (it is required inversion for the modules between the one under configuration and the system output).

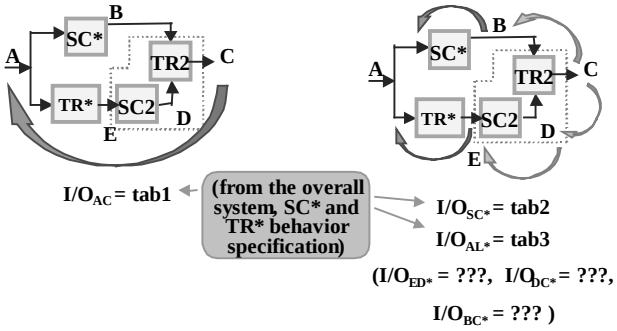

Fig.9. The Input/Output pairs decomposition step

Fig.9 shows how is possible to determine the I/O pairs for modules TR2 and SC2.
Similarly to electrical circuits in which the Kirchhoff's Voltage laws are applied, some of the not specified I/O pairs can be found starting from the given I/O pairs. For example, if the I/O$_{AC,}$ I/O$_{AB}$, and I/O$_{AE}$ pairs are known we can determine I/O$_{BC}$: I/O$_{ED}$ and I/O$_{DC}$ pairs are still indeterminate. The problem can be somehow solved in the training phase.

III.5 Modules tuning

Once input/output pairs are available at a module output the module itself can be trained. A first strategy for training the module can be summarised in tree sub-steps

1. tuning the modules needed to propagate the I/O system examples up to the module under analysis.
2. Tune the module under analysis
3. Execute a global tuning of the system for optimising all trained parameters.

Conversely, a second strategy suggests computing the unknown I/O pairs for modules SC2 and TR2 by relying on some hypotheses. For instance, module TR2 can be "forced" to perform the behaviour described by I/O$_{EC}$ pairs (I/O$_{EC}$ can be obtained by subtraction from the given I/O$_{AC}$ and I/O$_{AE}$ pairs) instead of that induced by the unknown I/O$_{DC}$ pairs. Once the TR2 module has been tuned we can train module SC2 by considering the TR2 SC2 subsystems as a whole. The soft computing module SC2 performs, in a way, a sort of behaviour "correction" to the TR2 modules.
These rules can be generalized for all sets of two modules when the relationship between external points is known.
It can be noted that if all the I/O pairs can be resolved, no final optimisation is required, since all the parameters have been tuned in the single module optimisations.

As described before, the single steps of the methodology generate and prune the number of the candidate composite systems. The larger the number of candidates the more probable the identified solution space contains the optimal composite system.

III.6 Composite system ranking

The final step, called composite system ranking, tests the performance of all candidate composite systems and, by using a figure of merit given by the designer, generates the ranking list. The optimal system is the first of the verified solution space.

The proposed methodology describes the logical sequence to perform a high-level automatic design for composite systems, but the real search for the optimal composite system can be achieved in different sequences. The situation is described in fig. 10 where a partially specified computational graph is analysed (the ranking phase is not considered).

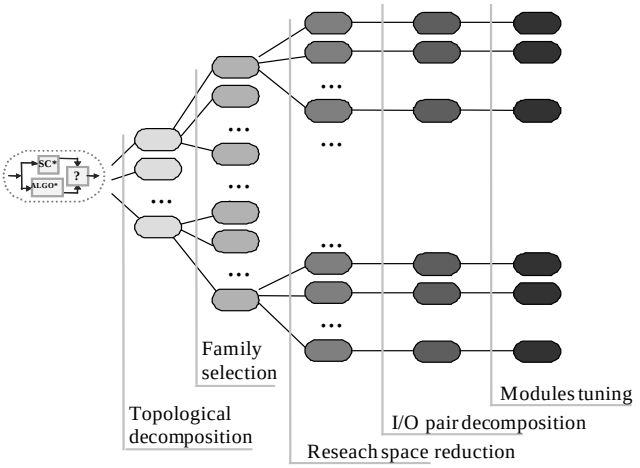

Fig.10. The increment of the number of objects during the methodology steps before the ranking phase.

The number of candidate composite systems (the oval elements) tend to increment during the first tree phases; during the I/O pair decomposition phase and the module-tuning phase, the number of system does not increment. Once we are in the ranking phase all models are completely specified and their performances can be tested.

The exploration of the space solution can be represented as a visit to a tree and can be

- Exhaustive (sequencing, parallel, branch&bound,…)
- Heuristic (sequencing, parallel, tabu search, simulated annealing, genetic algorithm, greedy, …).

In the exhaustive visit, all the leaves of the tree are considered. This corresponds to control the performance of all the tuned composite systems.

The procedure can be done in *parallel* (all the tuned composite systems are processed and finally verified in performances) or in a *sequencing* mode (each module is tuned and immediately verified).

If we consider a heuristic approach, the visit of some tree branches can be limited and other branches are envisaged. The pruning decision can be taken by observing the performances of the "nearest" composite system in the tree, and/or considering the requirements, e.g., computational complexity. It is important to note that the computationally demanding steps in the methodology are the I/O pair decomposition and the modules tuning; anyway, for each module these operations can be made in parallel. This characteristic is very promising for a network computing solution, if the overall computational complexity of the two last phases is inadequate for a single workstation.

IV. CONCLUSIONS

The presented high level methodology provides an effective method for achieving a high-level synthesis of embedded composite system for environmental application. Starting from a high-level description of the required behaviour of the system and constrains (computational complexity, latency, etc.), different composite systems are generated and tested. The candidate systems are created automatically by changing topologies, families of the models, testing different model families, tuning the models, validating the system and, finally, ranking the results. The methodology can perform a comprehensive search in the solution space of the problem, or can apply heuristics to reduce the computational load. The designer experience (selecting proper models and combinations of models) can be inserted in the designing system by DataBase of rules for models-composition in each methodological step. Further work is required to finish the implementation of the methodology, but preliminary results obtained emulating each single step are promising.

REFERENCES

[1] *L.Cristaldi, A. Ferrero, V. Piuri*: "Programmable Instruments, Virtual Instruments, and Distributed Measurement System" IEEE Instrumentation & Measurement Magazine, Sept. 1999 pp. 20-27.

[2] *A. Ferrero, V. Piuri*: "A simulation tool for virtual laboratory experiments in a www environment", proc. IMTC98, St.Paul, MN, May 1998, pp. 102-107

[3] *L. Benetazzo, M. Bertocco, et.al.*: "A Web-Based Distributed Virtual Educational Laboratory", IEEE Trans. on Instrumentation and measurement, vol.49, no.2, April 2000, pp.349-356.

[4] *K. Keutzer, S. Malik, et. Al.*: "System-level Design: Orthogonalization of concerns and platform-based Design" IEEE Trans. on computer-aided design of integrated circuits and system, vol.19,no.12, dec.2000,pp.1523-1543.

[5] *Business Week*, Mar.1999

[6] *C. Alippi, S. Ferrari, V. Piuri, M. Sami, F. Scotti*: "New trends in intelligent system design for embedded and measurement application, IEEE- I&M Magazine, Vol. 2, No. 2, June 1999

[7] *Ferrero, V. Piuri*: "Soft-Computing Technologies for Instrumentation and Measurement", in IEEE Technology Update Series: Instrumentation and Measurement, IEEE Press, 1998

[8] A. Ferrari, A. Sangiovanni-Vincentelli: "System design: traditional concepts and new paradigms", Int. Conf. Computer Design, Austin, TX, Oct. 1999.

[9] *C. Alippi, G.M. Bertoni, G.M. Diani, V. Piuri, F. Scotti*: "A Composite System Design Methodology for Instrumentation and Embedded System", Proc. IEEE Instrument. and Measurement Technology Conference Baltimore, USA, may 2000, pp.1086-1089.