

A Composite System Design Methodology for Instrumentation and Embedded System

Cesare Alippi, Guido M. Bertoni, Gian Marco Diani, Vincenzo Piuri, Fabio Scotti

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy

Abstract

Surely, the main goal when designing an embedded system is performance maximization. Nevertheless, physical constraints such as silicon area and/or power consumption take an active part in the system design by limiting, most of time, the solution space and hence reducing the system performance.

In this paper we present a methodology for selecting the best composite model for an embedded system in a constrained environment. The envisaged constraints are computational complexity and latency which respectively address the computational complexity issue in SW and HW realizations, respectively. It is assumed that the algorithm to be implemented in the embedded system is not given and must be constructed by relying on some (input, output) examples. Models considered for the system identification phase can be linear (e.g., ARMAX), non linear (e.g., neural based models) or composite (a suitable mix of linear and non linear models). The best solution is then selected from the candidate ones to optimally satisfy the application requirements

1. Introduction

In many applications related to the instrumentation and measurement field an embedded system must be considered to satisfy real time requirements, compactness of the final solution, cost minimization, reliability issues, etc [1, 2]. Quite often design of an embedded system requires an identification procedure whenever the equations ruling the process are unknown or too complex to be cast into a dedicated device/board [2-5].

In these cases, if the main and unique goal is system performance and the application can be solved with an equation-based or a linear model, the problem can be tackled with traditional identification approaches.

The system solution increases in complexity when some parts of the application are inherently non linear. In such a case, non linear models have been considered in the technical literature as viable solutions to address the non linear issue. In addition, in the design of an embedded system, constraints come into play; constraints depend on the particular application and vary according to the final

solution being implemented. Examples of constraints are computational requirements, silicon area and power consumption. In general, a compromise between system performance and competing constraints must be solved to obtain the best solution for the given application.

A composite system design requires a methodology for supporting a flexible selection of the most suitable accuracy/constraints tradeoff. Flexibility allows for reducing the time-to-market, it supports system modules reuse and provides the user with a set of high level tools. Each module of the composite system can then be implemented either on a dedicated hardware or run on a software platform; this partitioning can be carried out with HW/SW codesign methodologies [6,7].

In designing a composite embedded system it is necessary to suitably partition the task in sub tasks –or modules–, each of which is characterized by a model described by means of an algorithm. In this paper, we envisaged a set of models comprising simplified equation-based models, linear models (static and dynamic) and non linear ones based on neural networks as basic candidate modules.

Developing a physical model requires a good knowledge of the process; this task may be too complex or computationally unfeasible and a simplified approximate model must be considered.

In some cases no a priori information about the sub-task is available and the module must be treated as a black-box, whose internal behavior is identified on the basis of a set of examples. According to the model family chosen to characterize the sub-task we end with a set of linear and non linear solutions.

Once a set of possible modules has been obtained the original task can be partitioned in several ways by relying on different modules; constraints drive, at this stage, the search in the partition space towards the best solution.

We considered accuracy, computational complexity and implementation cost as main variables to be taken into account in the design phase.

Accuracy is defined as the validation performance of the system, e.g., measured with the sum of identification squared errors (SSE). The computational complexity measure is an indication of the software implementation complexity; such measure is defined as the number of

elementary instructions (additions, multiplication, memory access, etc) necessary to implement a module. We consider latency to measure the computational complexity on the physical implementation of the embedded system on a digital hardware. More in detail, latency refers to the time necessary to compute the critical path between the input presentation and the output generation.

The methodology introduced in the paper does not specify whether the system will be implemented on a software platform or not and it is implementation independent. Of course, hardware/software codesign techniques can be considered to further define which parts of the application should be run on a dedicated device or on a general purpose one [6,7].

The structure of the paper is as follows. The general methodology for composite system generation is given in section 2 where constraints are used to finally select the best solution. Section 3 applies the methodology to a real case, namely the determination of the best composite model for the common rail process in a naval application

2. A methodology for composite system design

In generating a set of candidates for the composite system it is important to keep the design time under control. Since an exhaustive exploration of the solution space is unfeasible, heuristics procedures are necessary to limit the solution space so as to identify a good solution within a reasonable time. Since a composite system is a composition of elementary modules implementing the generic input output $y=f(x)$ function we start from considering simple modules ruled by linear models.

In the following, we denote by $L_1, \dots, L_N$ the N linear models selected within a set of linear model families (e.g., ARX, ARMAX, MA, etc.). Surely, a linear model is the simplest one we can think about; the positive consequence is that if the best model L_B is optimal with respect to performance and satisfies the constraints then the design phase simply ends. In such a trivial case the composite system is composed of L_B itself.

Operatively, only a set of interesting model families should be generally taken into account; the most common ones are the ARX, ARMAX and Kalman filters models (other linear families can obviously be considered without changing the validity of the approach).

While an equation-based model can be directly inserted in the pool of elementary modules, we focus the attention on modules requiring a preliminary identification phase.

In the latter case performance validation is carried out by evaluating the model performance on a new set of data (cross-validation). If the validation error is a white noise (with a high confidence degree) then the considered model is optimal.

If this is not the case it means that the residual error is biased and a non linear equation-free model should be considered to improve the system performance.

Non linear equation-free families, e.g., based on neural networks [8], should then be envisaged to provide models $N_1, \dots, N_M$. The choice and the structure of the non linear family depend on a priori information, on the designer experience, etc. The designer, based on his experience and on the solutions suggested in the available literature could design a data base containing some hints about the most appropriate non linear functions for the specific application. In this paper we considered neural network based models of regression type for static modeling and recurrent neural networks for non linear ones suggested in [9,10].

We denote by N_B the best model found among the tested non linear ones. At this step of the procedure we have the two best models L_B and N_B . The next step is to compose linear and non linear models in the most appropriate way to generate a composite system. The aim of the composition, which always provides more complex models, is to improve the system performance. Note that in a composite system a HW/SW codesign finds its most natural expression since each obtained module can be either run in HW or SW.

There are two immediate ways to compose linear and non linear models. The basic idea is to generate the space of composite solutions by composing L_B and N_B with linear and non linear models to be trained in the attempt of improving the system performance. It is not so obvious that a combination of a linear model and a neural network is more complex than a single neural network model as happens in the application presented in section 3.

We envisage two possibilities in developing a composite system.

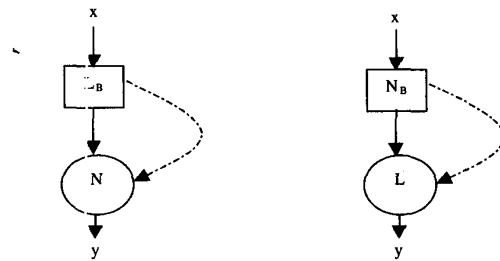


Figure 1: The serial composite model obtainable by combining L_B and N_B .

The first solution is to take the L_B model and build a serial model by means of a non-linear model family. The new system is then trained by configuring the neural network by keeping the L_B configuration frozen. A dual composite system can be obtained by fixing N_B and considering a serial model with a linear model to be

configured on the training samples. The two composite serial models are depicted in Figure 1, where the dotted line refers to the training data flow.

Another set of interesting composite systems can be obtained by considering a parallel connection. Also in this case we can generate two composite systems: one using a linear model, and another using a non-linear one as shown in figure 2. In each case it is possible to propagate the inputs also to the second model.

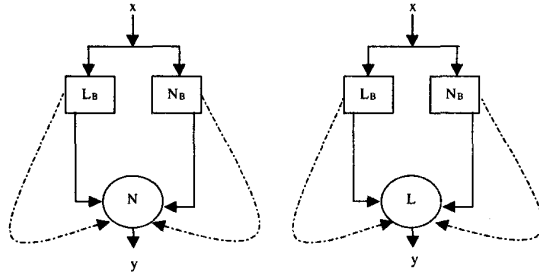


Figure 2: The two parallel models: continuous lines represent the data flow, dotted ones refer to the training data flow

```
Methodology (implementation_constraints, accuracy_constraints){
  create best linear model Lb
  if Lb doesn't satisfy accuracy_constraints {
    create best neural model Nb
    create composite models from Lb, Nb
    select the best model satisfying constraints
    if no models satisfy constraints{
      there is no solution
      relax implementation_constraints or
      explore other non linear families
    }
  }else{
    if Lb doesn't satisfy implementation_constraints{
      there is no solution
      relax implementation_constraints
    }
  }
}
```

Figure 3: a pseudo code for the composite system generation methodology

Once the models have been trained it is possible to associate each model with figures of merit such as Performances, Latency, Computational complexity, etc. This allows for creating a design space in which we can easily compare different models; the final model selection is now simply based on the constraint satisfaction and the considered HW/SW partitioning.

The final methodology for constructing and selecting the appropriate embedded composite system for an application is summarized in figure 3.

3. Applying the methodology to a real case

To shed light on how to use the methodology we considered a real case, the common rail system for fuel

injection in naval engines. A black-box description of the system is as follows.

Inputs:

- The amount of the fuel mass entering the common rail system;
- The rotations per minute of the engine (rpm);
- The common rail input pressure;

Output:

- The fuel mass injected into the cylinder;

Up to now, there are no sensors able to support an online measurement of the injected fuel mass for the common rail system. We implemented a physical equation-based model for the common rail but its computational complexity does not allow for on-line applications (the model contains several differential equations at partial derivatives). To give a hint about the computational complexity, a minute of the simplified physical model takes about six hours on a Sun Ultra Sparc 30 machine. The analysis of a set of output transients needs up-to four days of simulation.

To reduce the computational complexity we adopted a composite system analysis relying on the mentioned methodology. The composite model has been validated by considering system performance, latency and implementation costs.

Dynamic neural networks with the configuration suggested in [9] have been selected as the representative non linear model family. The direct consequence is that, under the assumption of maximum parallelism (a computational unit for each neuron) latency time depends only on the number of considered layers but not on the number of neurons per layer. This choice is characterized by minimum latency and maximum implementation cost for the HW (to be intended as power consumption and silicon area).

Model	SSE	Full parallel implementation			Serial	
		Hardware cost			Latency	Software cost
		Multipliers	Adders	Non-linear Elements		
Lb	2754.2	20	20	0	2	59
Nb	2485.2	481	460	20	5	983
Lb+N	2474.6	191	180	10	7	407
Nb+L	2651.5	491	470	20	7	1003
(Lb+Nb)+N	2418.8	642	610	30	10	1320
(Lb+Nb)+N with Inputs	2455.3	702	670	30	10	1434
(Lb+Nb)+L	2348.0	507	487	20	7	1056
(Lb+Nb)+L with Inputs	2529.1	513	493	20	7	1066

Table 1: The results of the models built in the common rail problem

The other extreme is when the composite system is implemented via software; in this case the solution has a fixed hardware cost (e.g., the processor onto which the SW is running) but latency generally reaches a maximum. To distinguish the SW latency from the hardware one we denote by software cost a measure based on the number of operations computed by the CPU in executing the algorithm.

Table 1 shows performances and costs for several composite systems. The first column addresses the structure of the composite system; the label "with inputs" means that external inputs have been propagated up to the subsequent module to be trained.

For each model we have considered the performance of the composite system in replicating the real one in terms of the SSE, the hardware cost of the fully parallel solution in terms of multipliers, adders and non-linear elements number. The software implementation has been evaluated by computing the necessary FLOPS.

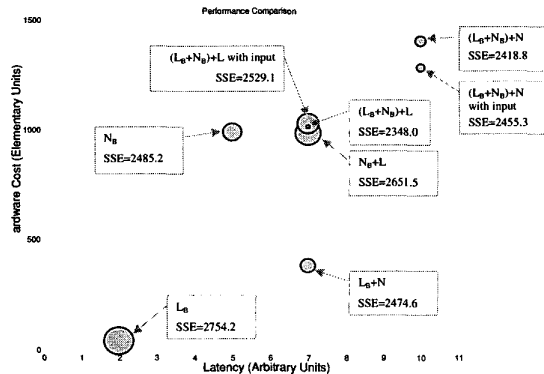


Figure 4: a full hardware implementation

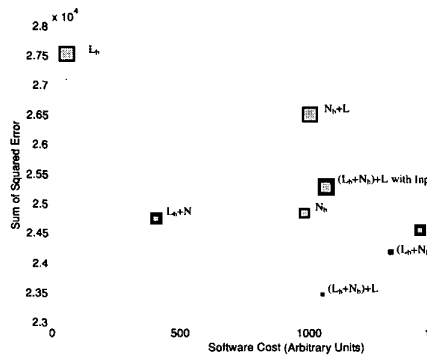


Figure 5: a full software implementation

Note that the L_B model is obviously the simplest one but not necessarily the optimal from the performance point of view if the process is non linear. We experienced in our application two main results: the model associated with the best SSE model is not the worst one according to the hardware cost, software cost and latency. This result was not predictable a priori; the methodology has therefore selected an extremely interesting composite model. The second result is that the L_B+N model is characterized by a good SSE and a contained implementation cost when compared to the N_B model. Figure 4 provides an immediate comparative picture of the full parallel solution given in table 1.

Similarly, figure 5 presents a vision of the performance/requirement relationship for the full software implementation. The main result is that the composite model $(L_B+N_B)+L$ gives the best performance independently from the chosen implementation. Figures 4 and 5 allow to solve the compromise performance/constraints and take care about the HW and SW partitioning.

5. Conclusions

The paper suggests a design methodology for composite systems. The proposed methodology allows for solving the performance/constraints tradeoff in systems in which at least one of its component has been identified by examples. Computational complexity, hardware latency have been the main variables involved in the system definition along with the system performance. It has been discovered that composite systems may provide, by keeping the same performance, an improvement of the computational complexity over more straight non linear compact models.

References

- [1] Ferreiro, V. Piuri, "Soft-Computing Technologies for Instrumentation and Measurement", in IEEE Technology Update Series: Instrumentation and Measurement, IEEE Press, 1998
- [2] 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
- [3] L.Ljung, "System identification: theory for the user". Prentice Hall, 1987
- [4] P.Van Overschee, B. de Moor, "N4SID: Subspace algorithms for the identification of combined deterministic and stochastic systems", ESAT/Sista Report 1992-34-I, Kath University Leuven, Belgium
- [5] F.J. Hill, G.R. Peterson, "Digital system: hardware organization and design" Wiley, 1978
- [6] G.De Micheli, M.G.Sami, "Hardware/Software Codesign", NASO ASI Series, Kluwer Academic Publishers, Vol. 310, 1995
- [7] D. Gajski, "High level synthesis: introduction to chip and system design" Kluwer, 1992
- [8] M.H.Hassoun, Fundamentals of Artificial Neural Networks, The MIT press, 1995.
- [9] K.S. Narendra, K.Parthasarathy, "Gradient methods for optimization of dynamical system containing neural networks". IEEE TNN, vol 1, no. 1, 4-27, 1990