International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 4, No. 2, July 2015, pp. 142~160
ISSN: 2089-4864

Application of New Approach of design flow for Hardware/Software Embedded System with the Use of Design Patterns in Fuzzy control system

Ali Bouyahya, Yassine Manai, Joseph Haggège
Departement of Electrical Engineering, Laboratory of Research in Automatic Control, National Engineering school of Tunis, University of Tunis el Manar, BP 37, Belvédère, 1002 Tunis, Tunisia

Article Info

Article history:

Received Dec 12, 2014
Revised Mar 27, 2015
Accepted Apr 24, 2015

Keyword:

Abstract_Factory
analytic model H
Application/Architecture
Adequacy
Design pattern
Embedded system
MAC_Cos
MAC_Operation
MAC_Sin
UML 2.0

ABSTRACT

This paper present a new method of conception of hardware/software embedded system design methodology based on use of design pattern approach called Abstract_factory. We called this new design tool “smart cell”. The main idea of the conception of embedded systems design is based on the used of object-oriented design ULM2.0. When the smart-cell is implemented, we justify their uses as a design tool that allows, first, to develop a specified application of fuzzy controller called PDC (parallel distributed conpensation). Second, the specification of the generation phases of the system architecture design, and eventually partitioning the application on heterogeneous platform based on hardware resource DSP and FPGA software to illustrate the proposed approach.

Corresponding Author:

Ali Bouyahya,
Departement of Electrical Engineering,
National Engineering School of Tunis,
University of Tunis el Manar BP 37, Belvédère, 1002 Tunis, Tunisia.
Email: ali.bouyahya@gmail.com

1. INTRODUCTION

There are two orientations in embedded systems, the technological field and the methodological one. The methodological orientation [1] try to develop the embedded system design process by examining new design tools in order to decrease the complexity of embedded systems. There are three main problems during the system design: the complexity, the hardware/software (HW/SW) partitioning and the reusability. Many frameworks are developed like transactional environments between application development and architecture synthesis are used to simplify the design process. Designers are recurring to raise the abstraction level, from Register Transfer Level (RTL) to system level. As a consequence, a gap between application development and architecture synthesis appears [2, 3]. In order to solve this problem, many design tools are developed in order to improve embedded system performances [4, 5]. In the field of control system implementation, many solutions were developed for linear time invariant (LTI) control and embedded real time control applications [6-8]. In [9], a design methodology based on a transactional model which is inserted between the application and the architecture is presented. In this way, the application is refined in an intermediate level which contains the architecture parameters. From this level, the implementation step is achieved in order to generate the RTL architecture.

Our approach of design try to solve the complexity problem, it consists to develop two intermediate environments in order to minimize the gap between application development and architecture synthesis [10, 11, 12, 20, 21, 22].

In order to develop reusable design tools in different fields, the design patterns were used [13, 20]. Our approach is used in control field which we develop a fuzzy controller for nonlinear system (inverted pendulum). Many researches in this field are performed. The work of [14, 15] develop an object analysis pattern for embedded system, further, a wrapper design pattern for an adapting behaviour of the soft IPs was proposed in [16]. The reusability of Intellectual Property (IP) blocks have been performed extensively for design hardware applications and IP blocks synthesis [17-18]. Mak [19] presents a design pattern modelling in *Unified Modelling Language* (UML). Many researches are performed for the reusability problem in order to develop new design tools that encapsulate all codesign phases in order to implement intellectual property (IP) blocks [20]. One attempt proposed in [21, 22] have as aim to develop the *smartcell* design tools in order to implement HW and SW IP blocks for heterogeneous platforms. Our contribution to resolve the reusability problem consists in the synthesis of IP blocks for hardware and software solutions from direct acyclic graph (DAC). The proposed approach examines the *Abstract_Factory* design pattern to produce the application and synthesis of IP blocks HW/SW (FPGA/DSP).

This paper is organized as follows. Section 2 introduces the fuzzy controller PDC (Parallel distributed Compensation) and Takagi-Sugeno fuzzy system. Section 3, the application of the proposed approach to the inverted pendulum is discussed in details. The conclusion and the future works are presented in the last section of this paper.

2. FUZZY CONTROLLER PDC (PARALLEL DISTRIBUTED COMPENSATION)

Takagi-Sugeno fuzzy model is a multimodel approach very used to modelize non linear sytems by construction with identification of input-output data. Many mechanical systems are modeling with T-S fuzzy system.

The continousT-S fuzzy model for a nonlinear system is written as follows.

If $z_1(t)$ is M_{i1} and… *and* $z_p(t)$ *is* M_{ip}

then

$$\begin{cases} \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) \end{cases} \quad i = 1......r \tag{1}$$

Which $M_{ij}(i = 1,2....r,\ j = 1,2.....p)$ is the fuzzy set and r is the number of model rules, $x(t) \in \Re^n$ is the states vector; $u(t) \in \Re^m$ is the input vector; $A_i \in \Re^{n\times n}$, the states matrix, $B_i \in \Re^{n\times m}$ the control and $z_1(t),......, z_p(t)$ are know premise variables.

The T-S fuzzy model can be written also as follow:

$$\dot{x}(t) = \frac{\sum_{i=1}^{r} w_i(z(t))\left(A_i x(t) + B_i u(t)\right)}{\sum_{i=1}^{r} w_i(z(t))}, \quad y(t) = \frac{\sum_{i=1}^{r} w_i(z(t)) C_i x(t)}{\sum_{i=1}^{r} w_i(z(t))}$$ r: is the number of model rules.

With

$$\begin{cases} w_i(z(t)) = \prod_{j=1}^{r} M_{ij}(z_j(t)) \\ h_i(z(t)) = \dfrac{w_i(z(t))}{\sum_{i=1}^{r} w_i(z(t))} \quad i = 1,2,\dots,r \end{cases} \tag{2}$$

The term $M_{ij}(z_j(t))$ is the grade of membership of $z_j(t)$ in M_{ij}.

Since $\begin{cases} \sum_{i=1}^{r} w_i(z(t)) > 0 \\ w_i(z(t)) \geq 0 \quad i = 1......r \end{cases}$, We have $\begin{cases} 0 < h_i(z(t)) < 1 \\ \sum_{i=1}^{r} h_i(z(t)) = 1 \end{cases}$ (3)

The final output can be written as follow

$$\begin{cases} \dot{x}(t) = \sum_{i=1}^{r} h_i(z(t))(A_i x(t) + B_i u(t)) \\ y(t) = \sum_{i=1}^{r} h_i(z(t)) C_i x(t) \end{cases} \tag{4}$$

The system used in this paper is the non-linear inverted pendulum modeled by Takagi-Sugeno fuzzy system

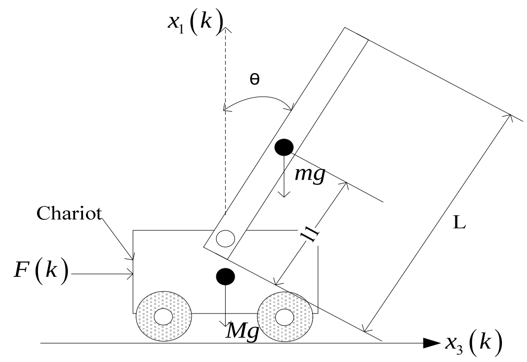


Figure 1. Inverted pendulum

The inverted pendulum is modelised in $x1 \in [-88°, 88°]$ because it's not controllable around $-\frac{\pi}{2}$ and $\frac{\pi}{2}$
The membership functions $h_i(x_1(t)),\ i = 1,2,3,4$ are obtained from the products F_1^1, F_1^2, F_2^1 et F_2^2 [23]

$$\begin{matrix} h_1(x_1) = F_1^1.F_2^1 & h_3(x_1) = F_1^2.F_2^1 \\ h_2(x_1) = F_1^1.F_2^2 & h_4(x_1) = F_1^2.F_2^2 \end{matrix} \tag{5}$$

$$F_1^1 = \frac{\cos(x_1) - 0.0348}{1 - 0.0348},\ F_1^2 = \frac{1 - \cos(x_1)}{1 - 0.0349},\ F_2^1 = \frac{1.5359\sin(x_1) - x_1}{x_1(1.5359 - 1)},\ F_2^2 = \frac{1.5359(x_1 - \sin(x_1))}{x_1(1.5359 - 1)}$$

The Inverted pendulum modeling and gives the following matrices:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 25.91161 & -0.2789 & 0 & 39.71095 \\ 0 & 0 & 0 & 1 \\ -1.1193 & 0.01222 & 0 & -14.801 \end{pmatrix},\ A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.294096 & -0.2789 & 0 & 39.71095 \\ 0 & 0 & 0 & 1 \\ -0.012704 & 0.01222 & 0 & -14.801 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 25.91161 & -0.2789 & 0 & 1.3819 \\ 0 & 0 & 0 & 1 \\ 0.038951 & 0.0004252 & 0 & -14.801 \end{pmatrix},\ A_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.294096 & -0.2789 & 0 & 1.3819 \\ 0 & 0 & 0 & 1 \\ 0.0004421 & 0.0004252 & 0 & -14.801 \end{pmatrix},$$

$$B_1 = \begin{pmatrix} 0 \\ 1.7078 \\ 0 \\ 0.02529 \end{pmatrix},\ B_2 = \begin{pmatrix} 0 \\ 0.05943 \\ 0 \\ 0.02529 \end{pmatrix},\ B_3 = \begin{pmatrix} 0 \\ 1.7078 \\ 0 \\ 0.02529 \end{pmatrix} \text{ et } B_4 = \begin{pmatrix} 0 \\ 0.05943 \\ 0 \\ 0.02529 \end{pmatrix}.$$

Finally the pendulum inverted is modelised via Takagi-Sueno in 4 rules as follow :

$$\text{Rule 1: if } \left(x_1 \text{ is } \mathrm{F}_1^1\right) \text{ then } \begin{cases} \dot{x}(t) = A_1\, x(t) + B_1\, u(t) \\ y(t) = C\, x(t) \end{cases}$$

$$\text{Rule 2: if } \left(x_1 \text{ is } \mathrm{F}_1^2\right) \text{ then } \begin{cases} \dot{x}(t) = A_2\, x(t) + B_2\, u(t) \\ y(t) = C\, x(t) \end{cases}$$

$$\text{Rule 3: if } \left(x_1 \text{ is } \mathrm{F}_2^1\right) \text{ then } \begin{cases} \dot{x}(t) = A_3\, x(t) + B_3\, u(t) \\ y(t) = C\, x(t) \end{cases}$$

$$\text{Rule 4: if } \left(x_1 \text{ is } \mathrm{F}_2^2\right) \text{ then } \begin{cases} \dot{x}(t) = A_4\, x(t) + B_4\, u(t) \\ y(t) = C\, x(t) \end{cases}$$

The PDC control law is written is the next equation

$$u(t) = -\frac{\sum_{i=1}^{r} w_i\left(z(t)\right) F_i x(t)}{\sum_{i=1}^{r} w_i\left(z(t)\right)} = -\sum_{i=1}^{r} h_i\left(z(t)\right) F_i x(t) \tag{6}$$

F_i represents the feedback matrices stabilizing the system in closed loop. This can be founded by the application of the LMIs techniques (Linear Matrix Inequality) with the use the LMI toolbox in Matlab.
The matrices F_i corresponding to each rule are written as follow:

$$F_1 = \begin{bmatrix} 41.2123 & 28.1726 & -76.1726 & 119.1816 \end{bmatrix} \tag{7}$$

$$F_2 = \begin{bmatrix} 23.544 & 6.5301 & -32.6125 & 34.2956 \end{bmatrix} \tag{8}$$

$$F_3 = \begin{bmatrix} 22.8407 & 8.8906 & -45.7892 & 122.6646 \end{bmatrix} \tag{9}$$

$$F_4 = \begin{bmatrix} 14.6584 & 11.7044 & -17.0676 & 53.4315 \end{bmatrix} \tag{10}$$

For the modelisation and conception details for the PDC controller, see [23].
After we describe the Takagi-Sugeno fuzzy system, the modelisation of the pendulum inverted with this technique and we present the PDC control law. We present the approach of conception of Hardware/Software Embedded System with the Use of Design Patterns for the PDC controller.

3. DEFINITION OF THE CONTEXT OF THE PROBLEM

In this approch of design embedded system [20]. We define four principals' actors who are involved in the operation of the unified structure. The first is related to the identification of methods for modeling specification, we call it the application. The second, related problem with the synthesis of IPs blocks HW/SW, we call it the architecture. The third is the communication between the different structure, that's means the problem of networking subsystems of the overall system. The fourth actor is the partitioning hardware / software. The following figure illustrates the context of the unified structure. [20]

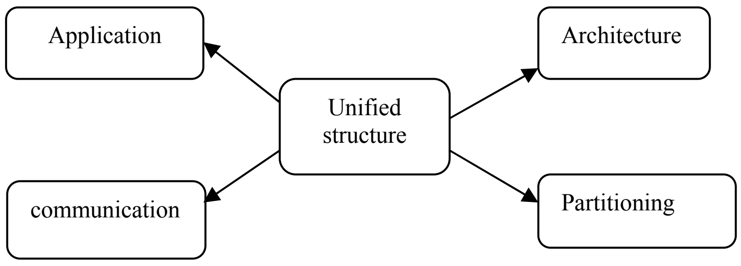


Figure 2. Unified structure [20]

The unified structure is decomposed by four principales actors: the developpement of the application, the synthesis of the architecture, the partionning of the hardware/software and the communication. We define all this actors one by one.

There are two levels of abstraction in the design of the PDC controller. In the first level of abstraction, the whole system is modeled by the unified structure called «Unified Structure system level», it breaks down the whole system into subsystems, each of them is modeled by a second-level unified structure called «Unified Structure second level».

3. 1 Abstraction Levels of the Unified Structure

This approach of two level abstraction design proposed is illustrated by the figure 3 [20]. In the following sections, the corresponding application of PDC controller and its architecture are developed through the unified structures corresponding to the fuzzy model distinguished.

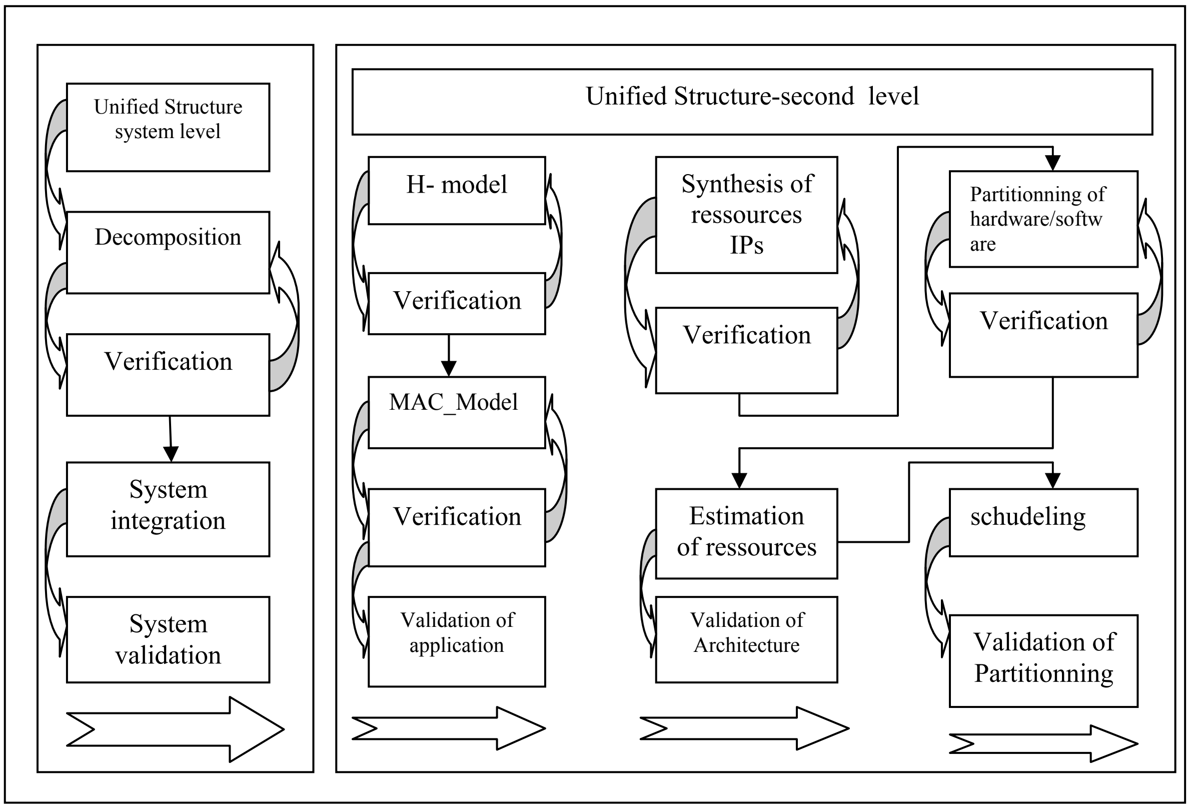

Unified Structure system level
Decomposition
Verification
System integration
System validation
Unified Structure-second level
H- model
Verification
MAC_Model
Verification
Validation of application
Synthesis of ressources IPs
Verification
Estimation of ressources
Validation of Architecture
Partitionning of hardware/software
Verification
schudeling
Validation of Partitionning

Figure 3. Multi-level design abstraction [20]

For the implementation of the hardware and software blocks IPs, two environments interfacing with cards on which the target is going to implement the embedded code examined, the CCS (Code Composer

Studio 3.1) to interface with software processor DSP, and the Xilinx ISE environment interfacing with FPGA board [24,25].

3.2 Unified Structure System Level

This section contains the modeling of system-level application. We present the global model of inverted pendulum. Subsequently, we present the modeling of this system in the state space plant. The system level granularity modeling application used in this approach to is the modelisation in the state space of the physical system (inverted pendulum).

3.3 Unified Structure Second Level

In this section, we present a different second-level unified structures such as unified structure input, the unified structure "controller", "physical system" and "outputs".

3.3.1 Unified Structure Input

The unified structure input contain the following data:
- The acquisition of the signal delivered by the setpoint
- The multiplication of the reference signal delivered by a matrix gain to adjust the static gain of the closed loop system.
- Transmission of the signal supplied to the control device.

3.3.2 Unified Structure Controller

Unified structure controller contains the following data:
- The acquisition of the signal delivered by the unified structure input.
- Generation of the control signal.
- Transmission of the control signal to the physical system unified structure.

3.3.3 Unified Structure Physical System

The unified structure physical contain the following data:
- Acquisition of the control signal supplied by the control device,
- Adjusting the control signal to the physical process.
- Transmission of the output signal to the output device.

3.3.4 Unified Structure Output

The unified structure output has the following features.
- Acquisition of the signal delivered by the physical system.
- Generating the output signal.

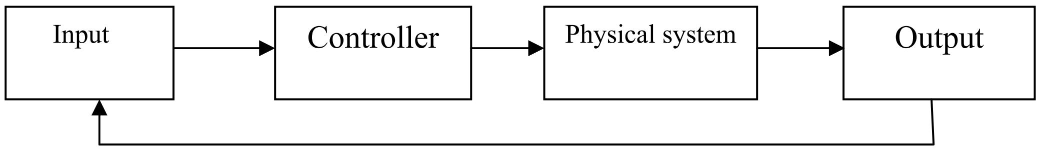


Figure 4. Unified structure second level [20]

The benefits of implementing this unified design approach by mastering the complexity of embedded systems structure; first is solving the complexity by raising the level of abstraction. Second is the reusability of IPs blocks to minimize the time-to-market, it is resolved by the development of the unified structure. Finally, the problem of automating of the design is processed. In addition to these advantages, this approach illustrates other objectives of local order. In the development of the unified structure second level, we examine three models for the development of the application. The first model is the model of the unified structure, the second is the analytical model and finally is the model of MAC_Operation on which we will work to synthesize the architecture. In the architecture, each second-level unified structure has a process for the synthesis of intellectual property blocks IPs hardware and software. Hardware IPs are synthesized with VHDL language, while the software IPs are synthesized with the C ++ language.

4. APPLICATION DEVELOPMENT

The development of the application is performed according to the algorithm presented in Figure 5. It consists in: first, the development of the model of the unified structure, second the development of analytical model H, and finally the development of a work model based on the MAC_Operation. The design pattern Abstract_Factory is responsible to fulfill the tasks of the actor application. Figure 4 illustrates the application actor.

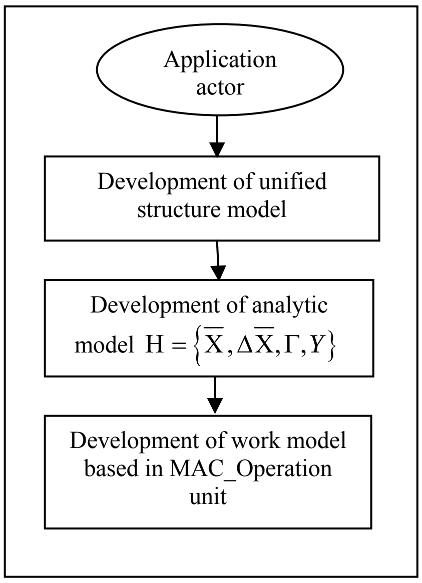

Application actor
Development of unified structure model
Development of analytic model $H=\{\overline{X},\Delta\overline{X},\Gamma,Y\}$
Development of work model based in MAC_Operation unit

Figure 5. Application Actor [20]

4.1 Development of the Analytical Model

The objective of the analytical model is to model the unified structure proposed by the state space representation [20]. The analytical model that we propose is defined by the following equation [20]

$$H=\{\overline{X},\Delta\overline{X},\Gamma,Y\} \tag{11}$$

$$\begin{cases} \overline{X}=\{A,B,C,D,X,y,u\} \\ u : input\ vector\ of\ the\ system \\ y : output\ vector\ of\ the\ system \\ X : state\ vector\ of\ the\ system \\ Y=\bigcup_{i=0}^{n} y_i;\ y_i : protocol\ of\ communication \\ \Gamma=\bigcup_{i=0}^{n} f_i;\ f_i : protocol\ of\ control \\ \Delta\overline{X}=\{\Delta X,\Delta Y\} \end{cases} \tag{12}$$

In this equations, the model $\overline{X}$ encapsulates the model of system written in the state space plant, the state vector, the input vector and the output vector. $\Delta\overline{X}$ represents the part of the system disturbance.The function sets the communication between hardware and software resources. In the analytical model H we define yhree steps. In the first step we define the transfer function model of the unified structure. The second step is to transform each transfer function to the state space representation using a simplified companion model. The third step is to transform the state space companion model representation in the discrete domain using the operator δ to develop recurrences equations.

4.2 Development of Model of Work MAC

The development of the model of work MAC consists to transform the recurrences equations developed from the analytical model to the graphs of tasks based on the MAC_operation. A MAC_Operation (Multiply and Accumulate) is represented by Figure 6 [20].

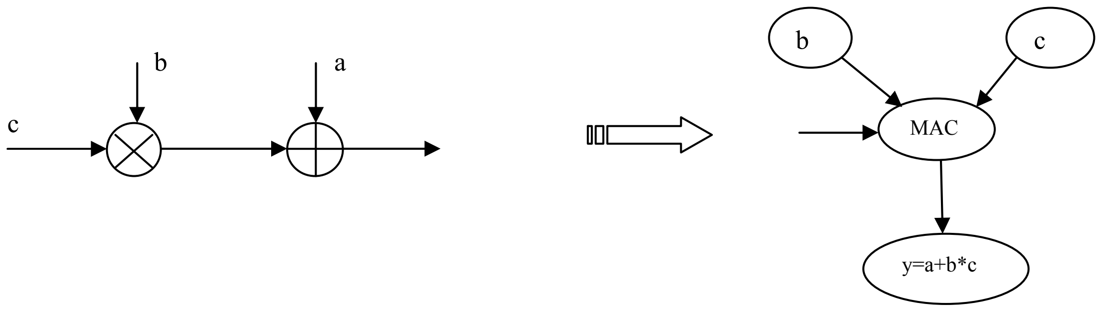


Figure 6. Mac_Operation [20]

The recurrence equation of a 4th order inverted pendulum, it converted into a set of tasks as the graphs shown in Figure 7.

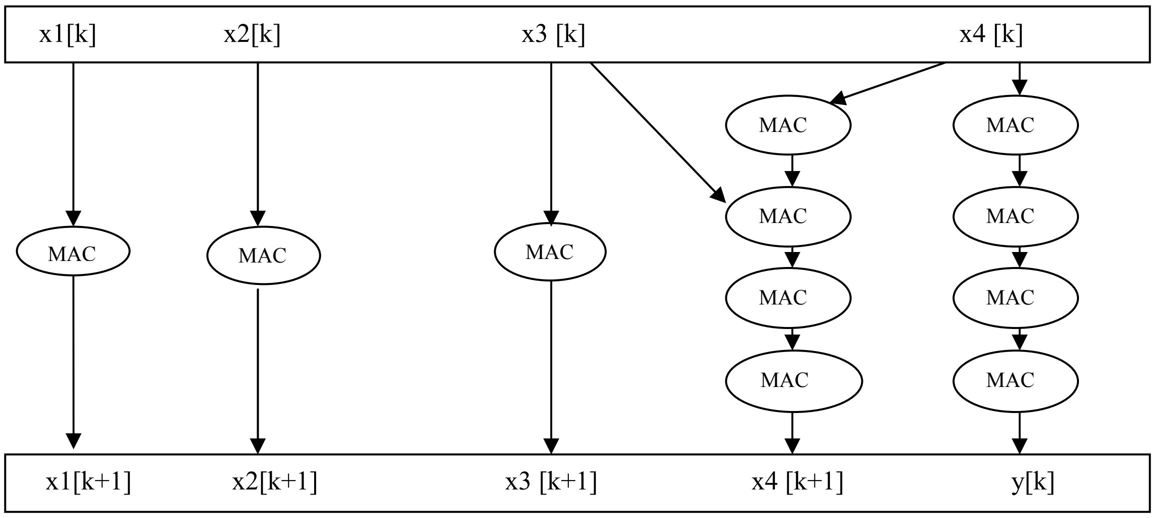


Figure 7. Transformation of a recurrence equation with the MAC unit [20]

The next section is devoted to the synthesis of embedded system of our PDC stabilization control law for inverted pendulum system.

5. SYNTHESIS OF THE ARCHITECTURE OF THE CONTROL LAW PDC

The actor of Architecture is the responsible for synthesis the hardware IPs and software IPs that realize the target architecture. It is within this architecture as the embedded system performs the dedicated application. Two methods of developing software and hardware IPs are distinguished. The first method is to develop the code to implement on target board by examining the corresponding functions. The second method is the exploitation of existing compilers in Matlab such as Real Time Workshop (RTW) environment.

5.1. Development of Architecture IP Soft

This section examines the problem of integration of the control law on a DSP card using the TLC (Target Language Compiler) of Real Time Workshop compiler and development environment CCS IDE TI DSP. belongs to the TI C2000 DSP family, the TI TMS 320C2812.

5.2. Development of Architecture IP Hard

This section examines the problem of integration of the control law on a DSP card using the TLC (Target Language Compiler) of Real Time Workshop compiler and development environment CCS IDE TI DSP. Belongs to the TI C2000 DSP family, the TI TMS 320C2812.

6. IMPLEMENTATION OF THE CONTROL LAW PDC ON HETEROGENEOUS PLATFORM

In this section, we present the implementation of the control law PDC. It begins by breaking the control system of an inverted pendulum in a set of unified structures. The second phase is the development of the application as a set of graphs tasks. The last phase is the synthesis of hardware and software IPs blocks of heterogeneous architecture.

6.1. Unified Design Structure

At the first level of abstraction (system level), the control system of inverted pendulum is decomposed into four sub-systems (fuzzy models TS) are modeled by a second-level unified structure. This decomposition is performed using the design pattern Abtract_Factory presented in Figure 8.

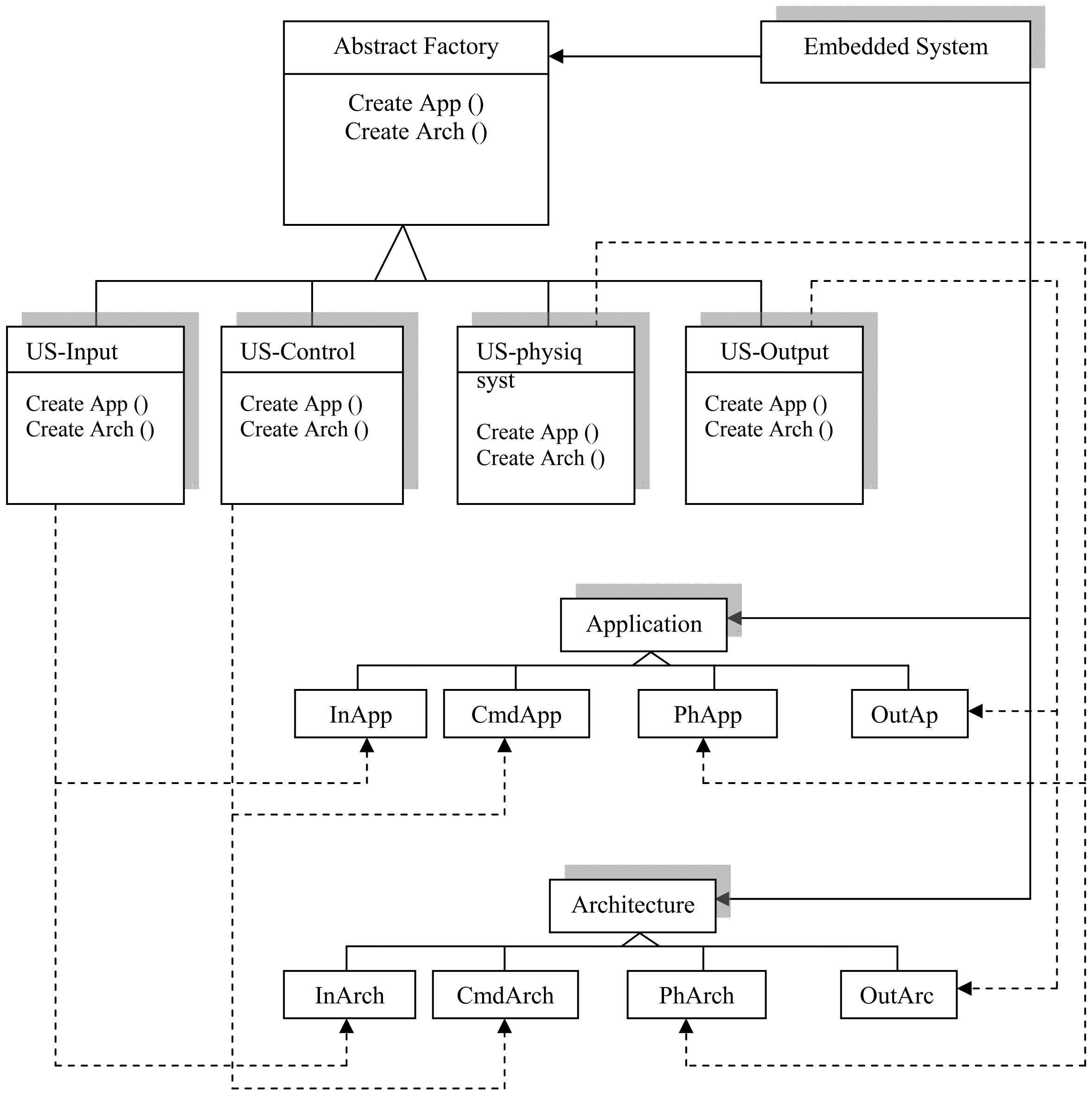


Figure 8. Décomposition of embedded system with design pattern Abstract_Factory [20]

After decomposition of the system in structure unified first level, we develop unified structures second level. We define four unified structures second level. Unified structure input, wich contain the acquisition of the input signal (setpoint) multiplied by the static gain to control the closed loop system. The second unified structure is the controller, it contain the elementaries control laws with states feedback for each subsystems of the pendulum inverted system. The third contain the modelisation of physical system and the foorth contain the output signal.

When we examines the pattern design Abstract-Factory developed by Figure 8, we specify the next classes developed by C++ with visual Studio10.

- Customer: is the embedded system.
- Abstract classes: Application, Architecture,
- Concrete classes: Input, controller, Physical System, output.
- Abstract Products: InApp, Cmdapp, Phapp, outApp, Inarch, Cmdarch, Pharch, Outarch,

In Figure 9, we present little party of C++ code of design pattern Abstact-Factory that allows to decompose the control system of the inverted pendulum. The CreateApp (), CreateArch () are functions declared virtual type. A virtual function is a function defined in a class and is intended to be overridden in derived classes.

```
class absFactory {
public:
 virtual Architecture createarch()
        {   return new architecture;  }
 virtual application createapp()
        {  return new application;  }
class sysemb
 {
public:
sysemb(absFactory& factory)
{
architecture* Architecture = factory.createarch();
application* Application = factory.createapp();
}
```

Figure 9. Design pattern Abstract-Factory

6.2. Development of Application for Inverted Pendulum System

6.2.1. H Analytical Model of the Control System

The development of the analytical model H is based on the state space of each subsystem.There are three phases in the analytical model H. The first phase consists in the development of the transfer function model of the unified second level structure. The second phase is to transform each transfer function in the state space using a simplified representation such as modified companion model. The third phase involves the transformation of state representation model in the discrete domain using the operator δ to develop the recurrence equations.

The analytical model is represented by the following equations

$$\mathrm{H} = \left\{\overline{\mathrm{X}}, \Delta\overline{\mathrm{X}}, \Gamma, Y\right\} \tag{13}$$

$$\begin{cases} \overline{\mathrm{X}} = \{A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4, C_1, C_2, C_3, C_4, D, X, y, u\} \\ u : input\ vector\ of\ system \\ y : output\ vector\ of\ system \\ X : state\ vector\ of\ system \\ Y = \bigcup_{i=0}^{n} y_i ; y_i : communication\ protocole \\ \Gamma = \bigcup_{i=0}^{n} f_i ; f_i : control\ function \end{cases} \tag{14}$$

The discrete system is defined by the following transfer equation:

$$\mathrm{F(z)} = \frac{b_n z^n + b_{n-1} z^{n-1} + ... + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + ... + a_1 z + a_0} \tag{15}$$

We change the variable $\gamma = z^{-1}$ the transfert function become

$$\mathrm{F}(\gamma) = \frac{b_n \gamma^{-n} + b_{n-1} \gamma^{-(n-1)} + ... + b_1 \gamma^{-1} + b_0}{a_n \gamma^{-n} + a_{n-1} \gamma^{-(n-1)} + ... + a_1 \gamma^{-1} + a_0} \tag{16}$$

The operator δ is defined by the next equation

$$\delta(f(t)) = \delta f[k] = f[k+1] - f[k] \tag{17}$$

$\delta = (z-1)T$ wich T is samples set. The transfert function with δ is defined by the following form.

$$\mathrm{F}(\delta) = \frac{p_n \delta^{-n} + p_{n-1} \delta^{-(n-1)} + ... + p_1 \delta^{-1} + p_0}{q_n \delta^{-n} + q_{n-1} \delta^{-(n-1)} + ... + q_1 \delta^{-1} + 1} \tag{18}$$

With

$$\begin{array}{ll} p_n = b_0 T^{n-1} & q_n = a_0 \\ p_{n-1} = b_1 T^{n-2} & q_{n-1} = a_1 T \\ p_1 = b_{n-1} T & q_1 = a_{n-1} T^{n-1} \\ p_0 = b_0 & q_0 = a_n T^n \end{array} \tag{19}$$

After the description of the system by the transfert functions $\mathrm{F}(\delta)$, we transforme each transfert function in the modified compagnon states representation with the following form.

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_4 \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & q_1 & 0 & \dots & 0 \\ 0 & 1 & q_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & q_{n-1} \\ -q_n & -q_n & \cdots & -q_n & 1-q_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_4 \end{pmatrix}_k + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ q_n \end{pmatrix} .u_k \tag{20}$$

$$y_k = \begin{pmatrix} p_1 & p_2 & \dots\dots & p_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_4 \end{pmatrix}_k + p_0.u_k$$

After the presentation of representations of fellow states, we write the recurrence equations and outputs of each fuzzy model of the system described by four subsystems. For the discretization of continuous subsystems, is taken as sampling set T = 0.05s and we treat the case of each sub-fuzzy model.
For the first fuzzy model subsystem we obtain the next transfert equations in δ .

$$\mathrm{F}_{11}(\delta) = \frac{-0.0044\delta^{-2} - 0.0024\delta^{-1} + 0.00211}{-2.006\delta^{-3} + 4.1484\delta^{-2} - 5.148\delta^{-1} + 1} \tag{21}$$

$$\mathrm{F}_{12}(\delta) = \frac{-5,25.10^{-5}\delta^{-3} - 6,585.10^{-5}\delta^{-2} + 3,409.10^{-4}\delta^{-1} - 4.05.10^{-4}}{2.006\delta^{-4} - 7.3522\delta^{-3} + 9.4357\delta^{-2} - 5.148\delta^{-1} + 1} \tag{22}$$

For the second fuzzy model subsystem we obtain the next transfert equations.

$$\mathrm{F}_{21}(\delta) = \frac{2,65.10^{-7}\delta^{-4} - 6,03.10^{-6}\delta^{-3} - 2,23.10^{-4}\delta^{-2}}{\delta^{-4} - 0,2575\delta^{-3} + 0,0103\delta^{-2} - 2,5.10^{-4}\delta^{-1}} \tag{23}$$

$$\mathrm{F}_{22}(\delta) = \frac{5,06.10^{-9}\delta^{-4} + 9,12.10^{-8}\delta^{-3} + 3,53.10^{-6}\delta^{-2} - 2,58.10^{-6}\delta^{-1}}{\delta^{-4} - 0,2575\delta^{-3} + 0,0235\delta^{-2} + 9,19.10^{-4}\delta^{-1} + 1,2.10^{-5}} \tag{24}$$

For the third fuzzy model subsystem we obtain the next transfert equations.

$$\mathrm{F}_{31}(\delta) = \frac{1,297.10^{-7}\delta^{-4} - 3,232.10^{-6}\delta^{-3} - 1,127.10^{-4}\delta^{-2}}{\delta^{-4} - 0,2545\delta^{-3} + 0,02312\delta^{-2} - 9,087.10^{-4}\delta^{-1} + 1,29.10^{-5}} \tag{25}$$

$$\mathrm{F}_{32}(\delta) = \frac{-5,10.10^{-9}\delta^{-4} + 3,825.10^{-9}\delta^{-3} + 3,182.10^{-6}\delta^{-2} - 2.607\delta^{-1}}{\delta^{-4} - 0,2545\delta^{-3} + 0,0231\delta^{-2} - 9,08.10^{-4}\delta^{-1} + 1,29.10^{-5}} \tag{26}$$

For the foorth fuzzy model subsystem we obtain the next transfert equations.

$$\mathrm{F}_{41}(\delta) = \frac{1,312.10^{-7}\delta^{-4} - 3,077.10^{-6}\delta^{-3} - 1,118.10^{-4}\delta^{-2}}{\delta^{-4} - 0,20442\delta^{-3} + 0,023124\delta^{-2} + 2,583.10^{-4}\delta^{-1} + 1,29.10^{-5}} \tag{27}$$

$$\mathrm{F}_{42}(\delta) = \frac{-5,118.10^{-9}\delta^{-4} + 7,497.10^{-8}\delta^{-3} + 3,194.10^{-6}\delta^{-2} - 2,599.10^{-4}\delta^{-1}}{\delta^{-4} - 0,20442\delta^{-3} + 0,023124\delta^{-2} - 9,03.10^{-4}\delta^{-1} + 1,29.10^{-5}} \tag{28}$$

After writing the transfers functions associated for each sub-models, we transform it to the states spaces representations. We treat the case of the second fuzzy model, we obtain two states representations described by the next equations

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & -2,5.10^{-4} & 0 & 0 \\ 0 & 1 & 0,0103 & 0 \\ 0 & 0 & 1 & -0,2575 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_k + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} .u_k \tag{29}$$

$$y_k = \begin{pmatrix} 0 & 2,23.10^{-4} & -6,03.10^{-6} & 2,65.10^{-7} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_k$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & 9,19.10^{-4} & 0 & 0 \\ 0 & 1 & 0,0235 & 0 \\ 0 & 0 & 1 & -0,2575 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_k + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} .u_k \tag{30}$$

$$y_k = \begin{pmatrix} -2,58.10^{-6} & 3,53.10^{-6} & 9,12.10^{-8} & 5,06.10^{-9} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}_k$$

6.2.2. Model of Work MAC

a) Development of Tasks Graphes of Fuzzy Submodels

The development of the work model is based on the development of recurrence equations studied in the analytical model H. The equation (29) is developed by using MAC_Operation in the next form

$$\begin{cases} x_1[k+1] = x_1[k] - 2,5.10^{-4} x_2[k] \\ x_2[k+1] = x_2[k] + 0,0103 x_3[k] \\ x_3[k+1] = x_3[k] - 0,2575 x_4[k] \\ x_4[k+1] = x_1[k] + x_2[k] + x_3[k] + u_k[k] \\ y_k[k] = 2,23.10^{-4} x_2[k] - 6,03.10^{-6} x_3[k] + 2, \end{cases} \tag{31}$$

The tasks graphes associated to the equations (31) is described by the next figure.

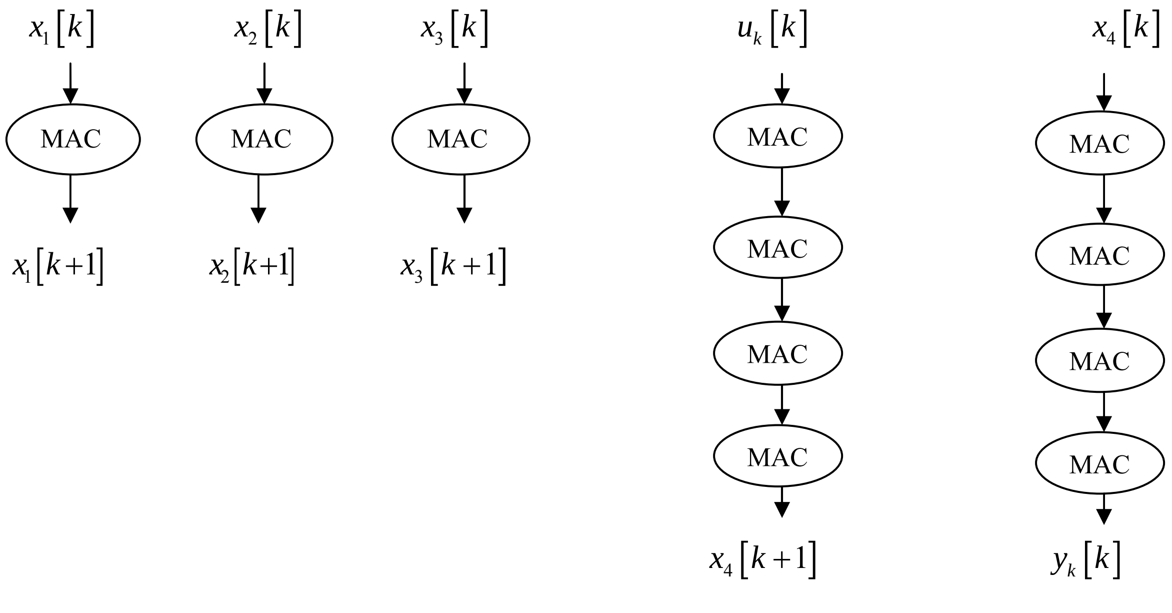


Figure 10. Tasks graphes [20]

b) Task Graphs of Activations Functions

In the model of work, we develop the task graphs associated to the activation functions of each fuzzy sub-system $h_i(x_1), i = 1,2,3,4$. The membership functions $h_i(x_1)$ are defined by the equations below

$$h_1(x_1) = 2,9696.\frac{\sin(2x_1)}{2x_1} - 1,9334.\cos(x_1) - \frac{\sin(x_1)}{x_1} - 0,0672 \quad (32)$$

$$h_2(x_1) = 2,9696.\cos(x_1) - 2,9696.\frac{\sin(2x_1)}{2x_1} - 0,1033.\frac{\sin(x_1)}{x_1} - 0,1033 \quad (33)$$

$$h_3(x_1) = 2,97.\frac{\sin(x_1)}{x_1} - 2,97.\frac{\sin(2x_1)}{2x_1} + 1,866.\cos(x_1) - 1,866 \quad (34)$$

$$h_4(x_1) = 2,97.\frac{\sin(2x_1)}{2x_1} - 2,97.\frac{\sin(x_1)}{x_1} - 2,97.\cos(x_1) - 2,97 \quad (35)$$

The activation functions contain non-linear terms such as $\frac{\sin(2x_1)}{2x_1}, \frac{\sin(x_1)}{x_1}$ and $\cos(x_1)$. We replace these terms by their limited development to order 4:

$$\cos(x_1) = 1 - \frac{x_1^2}{2} + \frac{x_1^4}{4!} - \frac{x_1^6}{6!} + \frac{x_1^8}{8!} \quad (36)$$

$$\frac{\sin(x_1)}{x_1} = 1 - \frac{x_1^2}{3!} + \frac{x_1^4}{5!} - \frac{x_1^6}{7!} + \frac{x_1^8}{9!} \quad (37)$$

$$\frac{\sin(2x_1)}{2x_1} = 1 - \frac{(2x_1)^2}{3!} + \frac{(2x_1)^4}{5!} - \frac{(2x_1)^6}{7!} + \frac{(2x_1)^8}{9!} \quad (38)$$

The limited development can be written as follow

$$\cos(x_1) = 1 + (-\frac{1}{2} + (\frac{1}{4!} + (-\frac{1}{6!} + \frac{x_1^2}{8!})x_1^2)x_1^2)\,x_1^2 \quad (39)$$

$$\frac{\sin(x_1)}{x_1} = 1 + (-\frac{1}{3!} + (\frac{1}{5!} + (-\frac{1}{7!} + \frac{x_1^2}{9!})x_1^2)x_1^2)\,x_1^2 \quad (40)$$

The nonlinear terms and are presented by two MAC_COS and MAC_SIN operation using a MAC_Operation on. The MAC_COS operation is summarized by Figure 11.

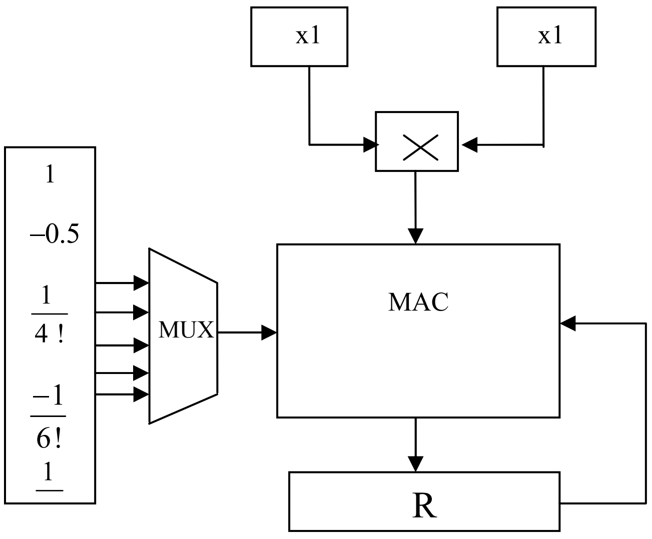


Figure 11. Mac_cos Operation [20]

R is a register initially charged by $\frac{1}{8!}$.

To run a operation MAC_cos or Mac_sin, it takes four Mac_operation. After we model the nonlinear terms by MAC_operations, the final graphs of tasks associated with activation functions is describe by the next figure.

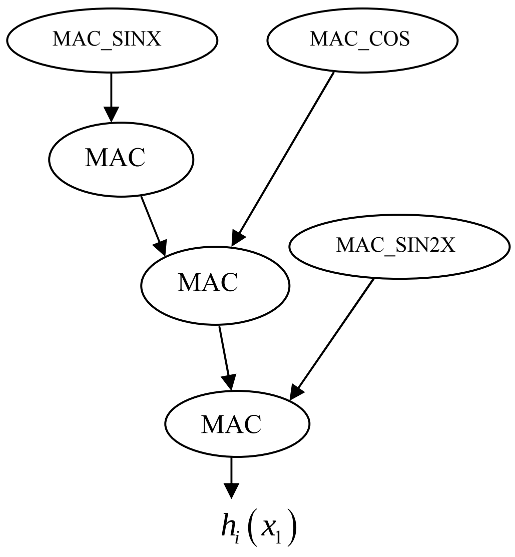


Figure 12. Task graph of activations functions

c) Tasks Graphs of Control Law:

We define the tasks graphs of elementary control law and the resulting control law PDC.The basic commands are described by the following equations.

$$u_1[k] = 41.2123x_1[k] + 28.1726x_2[k] \text{-} 76.1726x_3[k] + 119.1816x_4[k] \quad (41)$$

$$u_2[k] = 23.544x_1[k] + 6.5301x_2[k] \text{-} 32.6125x_3[k] + 34.29566x_4[k] \quad (42)$$

$$u_3[k] = 22.8407x_1[k] + 8.8906x_2[k] \text{-} 45.7892x_3[k] + 122.6646x_4[k] \quad (43)$$

$$u_4[k] = 14.6584x_1[k] + 11.7044x_2[k] \text{-} 17.0676x_3[k] + 53.4315x_4[k] \quad (44)$$

The control law PDC is defined as follow:

$$u_k[k] = \sum_{i=1}^{r} h_i(x_1)u_i[k] \quad (45)$$

The development of the unified structure of the application of the second level is to develop the four products abstract classes according to design pattern Abstract_Factory structure.

The products abstract classes are: InApp, Cmdapp, Phapp, outApp.

The InApp class contains the instructions signals multiplied by static gains. Static gains are calculated by the following formula

$$G_i = \frac{1}{C_i(-A_i + B_iF_i)^{-1}B_i} \quad (46)$$

The Cmdapp class contain the task graphs of elementary commands with state feedback and the final PDC control law.

The Phapp class contain the task graphs developed from the recurrence equations for each subsystem.

The outApp class contains the task graphs outputs of each system and the final output written as follow

$$y_k = \sum_{i=1}^{r} h_i(x_1) C_i x_k \quad (47)$$

6.3 Development of Architecture IPs Hardware/Software:

The creation of the architecture is based on the synthesis of block IPs hardware and software. The IP hardware is to generate VHDL related to FPGA hardware target and the C ++ related to DSP (TI TMS 320C2812) software target from the Simulink block "fuzzyblock" presented by the figure 13 representing the closed loop control of each sub-systems of the inverted pendulum system, their outputs are multiplied by the activation functions to give the final output written $y_k = \sum_{i=1}^{r} h_i(x_1) C_i x_k$.

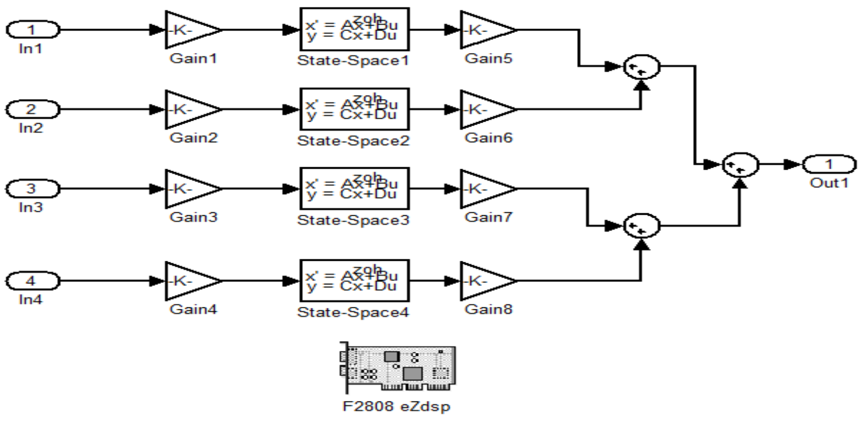


Figure 13. Simulink Matlab “Fuzzyblock” of inverted pendulum

After testing the Matlab/Simulink called "fuzzyblock", we synthesize the associated VHDL code. Matlab environment generates a "hdlsrc" project, which contains six vhd files. The files: “Fuzzyblock, fuzzyblock_pkg, State_Space1, State_Space2, State_Space3, State_Space4”.

The fuzzyblock.vhd file contains the entity and architecture associated “ Fuzzyblock ” . The fuzzyblock_pkg file contains the type of final output.

The statespace.vhd files contain entities and architectures associated with block of states spaces representations of each sub-systems fuzzy models.

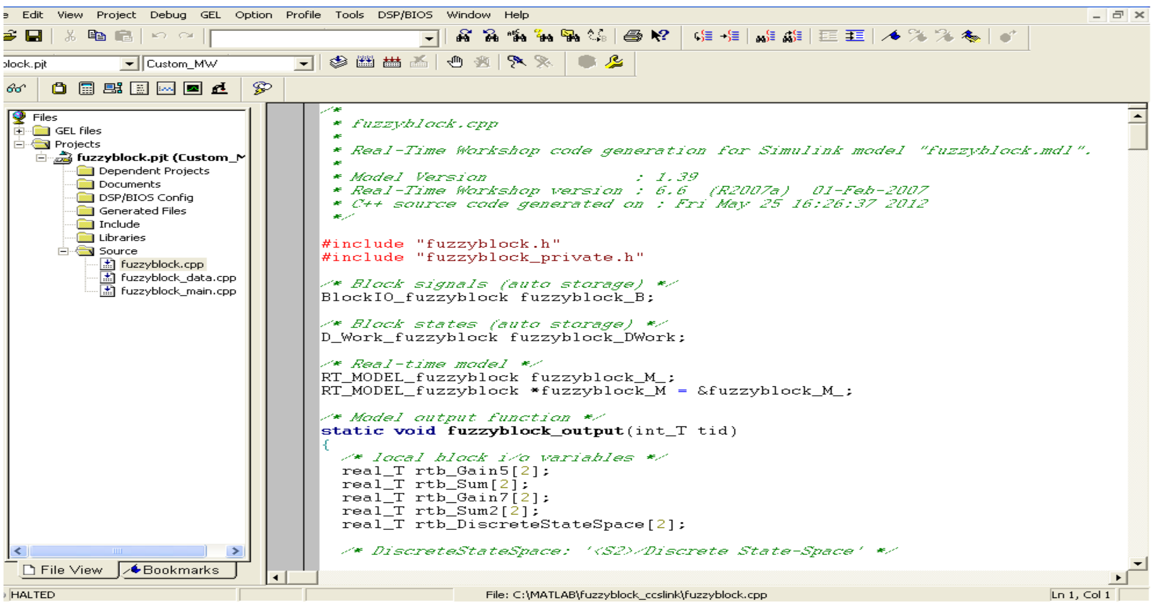
Figure 14 present a part of VHDL code generated by Matlab Simulink “fuzzyblock”.

```
--Définition du package fuzzy block_pkg
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE fuzzyblock_pkg IS
  TYPE vector_of_real IS ARRAY (NATURAL
RANGE <>) OF real;
END fuzzyblock_pkg;
--Définition de l'entité et l'architecture du
 --fuzzyblock

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
USE work.fuzzyblock_pkg.ALL;
ENTITY fuzzyblock IS
  PORT( In1:   IN    real;  -- double
        In2:   IN    real;  -- double
        In3:   IN    real;  -- double
        In4:   IN    real;  -- double
```

```
Out1:   OUT   vector_of_real(0 TO 1)  -- double [2]
        );
END fuzzyblock;
ARCHITECTURE rtl OF fuzzyblock IS
  -- Component Declarations
  COMPONENT State_Space1
    PORT( In1:   IN    real;
Out1:OUT vector_of_real(0 TO 1));
  END COMPONENT;
  COMPONENT State_Space2
    PORT( In1:IN real;
Out1: OUT vector_of_real(0 TO 1));
  END COMPONENT;
  COMPONENT State_Space3
    PORT( In1:   IN    real;
 Out1:OUT vector_of_real(0 TO 1));
  END COMPONENT;
  COMPONENT State_Space4
    PORT( In1:IN    real;
Out1:OUT vector_of_real(0 TO 1));
  END COMPONENT;
```

Figure 14. VHDL code

T the REAL_TIME Worshop with TI C2000 DSP generates the project containing the sources code in C ++ related to "fuzzyblock" Simulink. We chose code generation for TI TMS320C2812/DSP. Figure15 shows the creation of the CCS project.



```
/*
 * fuzzyblock.cpp
 *
 * Real-Time Workshop code generation for Simulink model "fuzzyblock.mdl".
 *
 * Model Version              : 1.39
 * Real-Time Workshop version : 6.6  (R2007a)  01-Feb-2007
 * C++ source code generated on : Fri May 25 16:26:37 2012
 */

#include "fuzzyblock.h"
#include "fuzzyblock_private.h"

/* Block signals (auto storage) */
BlockIO_fuzzyblock fuzzyblock_B;

/* Block states (auto storage) */
D_Work_fuzzyblock fuzzyblock_DWork;

/* Real-time model */
RT_MODEL_fuzzyblock fuzzyblock_M_;
RT_MODEL_fuzzyblock *fuzzyblock_M = &fuzzyblock_M_;

/* Model output function */
static void fuzzyblock_output(int_T tid)
{
  /* local block i/o variables */
  real_T rtb_Gain5[2];
  real_T rtb_Sum[2];
  real_T rtb_Gain7[2];
  real_T rtb_Sum2[2];
  real_T rtb_DiscreteStateSpace[2];

  /* DiscreteStateSpace: '<S2>/Discrete State-Space' */
```

Figure 15. Creation of the CCS project fuzzyblock

7. CONCLUSION

In this work, a multilevel design flow for embedded system through investigating the design pattern concept was used. In system level of abstraction, the system decomposition is realized with the design pattern *Abstract_Factory*. In the second level, each Smart_Cell realizes the development of the model of work based in the graphs tasks, the DAG development and the IP_hard/IP_soft blocks synthesis.

The VHDL and C++ DSP codes are tested on target cards. the next work is the partitioning of graphs of tasks using the algorithm "Ant Colony Optimization" and communication between the various graphs between hardware and software targets.

REFERENCES

[1] R.C. Dorf, “Systems, Controls, Embedded Systems, Energy, and Machines”, Taylor & Francis, New York, 2006, pp. 486-511.

[2] K. Virk and J. Madsen, “*A System-Level Multiprocessor System-on-Chip Modeling Framework*”, Proceedings of SOC, 2004.

[3] A.D. Pimentel and C. Erbas, “A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels”, *IEEE Transactions on Computer*, Vol. 55, No. 2, February 2006, pp. 99-112.

[4] B. Zhou, W. Qiu and C. Peng, “*An Operaing System Framework for Reconfigurable Systems*”, Proceedings of CIT, Salt Lake, 2005.

[5] S. Pasricha, N. Dutt and M.B. Romdhane, “*Using TLM for Exploring Bus-Based SoC Communication Architectures*”, Proceedings of ASAP, Atlantic, 2005.

[6] R. Cumplido, S. Jones, R.M. Goodall and S. Bateman, “A High Performance Processor for Embedded Real-Time Control”, *IEEE Transactions on Control Systems Tech- nology*, Vol. 13, No. 3, May 2005, pp. 485-492.

[7] X. Wu, V.A. Chouliaras, J.L. Nunez and R.M. Goodall, “A Novel DS Control System Processor and its VLSI Implem-Entation”, *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, Vol. 16, No. 3, March 2008, pp. 217-228.

[8] D.L. Sancho-Pradel and R. M. Goodall, “Targeted Processing for Real-Time Embedded Mechatronic Sys- tems”, *Control Engineering Practice*, Vol. 15, 2007, pp. 363-375.

[9] Y. Atat and N.E. Zergainoh, “Automatic Code Generation for MPSoC Platform Starting From Simulink/Matlab: New Approach to Bridge the Gap between Algorithm and Architecture Design”, *Conference of ICTTA*, Bali Island, 2008.

[10] G. Wang, W. Gong and R. Kastner, “Application Parti- tioning on Programmable Platforms Using the Ant Colony Optimization”, *Journal of Embedded Computing*, Vol. 2,

[11] Y. Manai, J. Haggège and M. Benrejeb, “HW/SW Partitioning in Embedded System Conception Using Design Pattern Approach”, *Conference of JTEA*, Hammamet, 2008.

[12] K.B. Chehida, “Méthodologie de Partitionnement Logiciel/Matériel Pour Plateformes Reconfigurables Dynamiquement”, PhD Thesis, Université de Nice-Sophia Anti- polis, France, 2004.

[13] E. Gamma, *et al*., “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, Massachusetts, 1995.

[14] S. Konrad, H.C. Cheng and L.A. Campbell, “Object Analysis Patterns for Embedded Systems”, *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, December 2004, pp. 970-990.

[15] S. Konrad and B. Cheng, “*Requirement Pattern for Embedded System*”, Proceedings of the IEEE Joint International Conference on Requirements Engineering, Atlanta, 2002.

[16] R. Damasevicius, G. Majauskas and V. Stulikys, “*Application of Design Patterns for Hardware Design*”, Proceedings of DAC, Anaheim, 2-6 June 2003, pp. 48-53.

[17] F. Rincon, F. Moya and J. Barba, “*Model Reuse through Hardware Design Patterns*”, Proceedings of Design, Automation, and Test in Europe, 2005.

[18] P. Coussy, *et al*., “Constrained Algorithmic IP Design for System-on-Chip”, *Integration, the VLSI Journal*, Vol. 40, No. 2, 2007, pp. 94-105.

[19] J.K. Mak, C.S. Choy and D.P. Lun, “*Precise Modeling of Design Patterns in UML*”, Proceedings of International Conference on Software Engineering, 2004.

[20] Y. Manai, “Contribution à la conception et la synthèse d’architecture de systèmes embarqués utilisant des plates-formes hétérogènes”, Ph.D. Dissertation, Ecole Nationale d’Ingénieurs de Tunis, Tunisia, 2009.

[21] Y. Manai, J.Haggège, M. Benrejeb “New Approach for Hardware/Software Embedded System Conception Based on the Use of Design Patterns”, *J. Software Engineering & Applications*, vol 3, pages 525-535,2010.

[22] Y. Manai, J. Haggège and M. Benrejeb, “PI-Fuzzy Con- troller Conception with Design Pattern Based Approach”, 14*th IEEE International Conference on Electronics, Cir- cuits and Systems*, Marrakech, 2007, pp. 483-489.

[23] A. Bouyahya, Y.Manai and J. Haggège ” New Condition of Stabilization for Continuous Takagi-Sugeno Fuzzy System based on Fuzzy Lyapunov Function”, *IEEE International Conference on Electrical Engineering and Software Applications (ICEESA)*, 21-23 Mars 2013

[24] Bharatesh N, Rohith S, ”FPGA Implementation of Park-Miller Algorithm to Generate Sequence of 32-Bit Pseudo Random Key for Encryption and Decryption of Plain Text”, *International Journal of Reconfigurable and Embedded Systems* Vol. 2, No. 3, November 2013, pp. 99~105

[25] S.M. Shashidhara*, P. Sangameswara Raju**, “**FPGA Based Embedded System Development for Rolling Bearings Fault Detection of Induction Motor”, *International Journal of Reconfigurable and Embedded Systems*, Vol. 2, No. 3, November 2013, pp. 127~134

BIOGRAPHIES OF AUTHORS

Ali Bouyahya was born in Tunisia on March 1986, he received the B.S. degree in Electrical Engineering from "Ecole Supérieure de Technologie et d'informatique (ESTI)" and Master degree in Automatic and Signal Processing 2012 from "Ecole Nationale d‟Ingénieurs de Tunis" He is currently in Phd. His research interests include embedded systems, nonlinear control, Takagi-Sugeno fuzzy uncertain systems.

Yassine Manai was born in Tunisia on December 1979. He received the Master degree in Automatic and Signal Processing and the Doctorate degree in Electrical Engineering from the "Ecole Nationale d‟Ingénieurs de Tunis" (ENIT) Tunisia in 2005 and 2009 respectively. His Doctorate thesis is prepared within the framework of unit research "Laboratoire de Recherche en Automatique" (LA.R.A) about Embedded System Architectures Design and Synthesis by the use of Heterogeneous Platforms. His research interests are embedded systems, stability and stabilization of Takagi-Sugeno fuzzy systems, and its applications.

Joseph Haggège was born in 1975 in Tunis, Tunisia. He graduated from "Ecole Nationale d'Ingénieurs de Tunis" in 1998, he received the PhD degree in Electrical Engineering 2003 and the Habilitation in 2010. He is currently Senior Lecturer at the "Ecole Nationale d'Ingénieurs de Tunis". His research interests are in the area of heuristic optimisation, embedded systems and robust digital control.