SystemC-A Modeling of an Automotive Seating Vibration Isolation System

H. Al-Junaid
University of Bahrain

T. Kazmierski and L. Wang
University of Southampton

Abstract—A modeling methodology for mixed physical domains system in a new modelling Language is presented. The system is automotive seating vibration isolation system with electronic control. It is described and simulated in SystemC-A, an extended version of SystemC which provides analogue, mixed-signal and mixed-domain modeling capabilities. Results show that SystemC-A provides efficient means to model and investigate performance of complex mixed-domain systems for automotive applications.

I. INTRODUCTION

SystemC-A [1] is an extended version of SystemC [2] intended to extend the modeling capabilities of SystemC to the analogue domain. Most of the powerful features of VHDL-AMS and Verilog-AMS are provided in SystemC-A in addition to a number of extra advantages such as high simulation speed, support for hardware-software co-design and for high levels of modeling. SystemC-A was validated by the authors in previous publications in modelling mixed-signal systems such as switched-mode power supply and phased locked loop. The systems were modelled at circuit level, behavioural level, system level or a mixture of them. In this paper, SystemC-A will be validated by modelling analogue mixed-physical domain system.

In automotive industry the attenuation of road disturbances of light-duty and off-road vehicles is a very important issue in their riding quality. The occupants are subjected to high and prolonged disturbances on rough roads. One strategy to attenuate the vibrations between the passenger seat and the vehicle’s floor is the use of vibration isolation systems by placing a well-controlled actuator in between. The system comprises electrical, hydraulic and mechanical components and the traditional way of modeling such a system at component level is to model each domain separately in different languages and/or environment. A good model of the whole system is therefore required to reflect all the details of the actual system.

Hardware Description Languages (HDL) such as VHDL-AMS [3] and Verilog-AMS [4] have appeared recently to provide excellent environments to model mixed physical domain systems. In HDLs, different parts of the system are represented in hierarchical modules connected together through ports and signals. Different types of ports, signals, cross and through quantities (electrical, magnetic, mechanical, etc) are provided in HDLs mixed-domain libraries. The use of HDL in automotive design has been started recently. However, the current underlying simulators are slow compared with C-based simulations [1].

The system modelled and simulated in this paper was designed originally by Liu and Wagner [5] [6]. They simulated the system in a non-HDL environment. Our proposal is to reintroduce the system model using highly flexible HDLs. The SystemC-A model of this complex, mixed-domain system is based on the earlier HDL model developed in VHDL-AMS [7]. This work shows that SystemC-A, being based on C++, can offer additional benefits as explained in the next section.

II. SYSTEMC-A

SystemC-A provides constructs to support user-defined ordinary differential and algebraic equations, analogue system variables, and analogue components to enable modeling of analogue and mixed-signal systems from very high levels of abstraction down to the circuit level. Support for digital-analogue interfaces has been provided for smooth integration of digital and analogue parts. An underlying analogue simulator has been added using efficient linear and nonlinear solvers to assure accurate and fast simulations. A novel implementation of the lock-step mixed-signal synchronization method is used to integrate the analogue kernel with the digital one [8]. The following three paragraphs gives a brief description of the SystemC-A main extension which will help the reader in understanding the models in Section IV.

A. Analogue system variables

In order to provide a mechanism for modelling non-linear AMS systems, the new language should provide a notation for DAEs. In the set of DAEs the analogue system variables are the unknowns. The C++ concept of inheritance is used to define various types of analogue system variables, such as node, flow, and free variables. In SystemC-A, they represent a hierarchy of system variables, all derived from an abstract base class called sc_a_system_variable. Currently only three types of variables derived from the base class have been defined, and this proved enough to model most application examples. The variable classes hierarchy can be extended further to model other types of applications. In a SystemC-A description, it is an error if the total number of variable objects does not correspond with the total number of equations provided by SystemC-A component objects and this is to have a symmetric system matrix.
B. Analogue components

Analogue circuit components have been developed to provide equations which describe analogue behaviour. Similarly to the system variable hierarchy, components are derived from an abstract base class (\texttt{sc\_a\_component}). A component abstract base class contains virtual build method to be invoked by the analogue kernel at each time step to build the system matrix. A sample component class hierarchy is illustrated in Fig.1 with examples of SPICE-like circuit elements such as resistor, capacitor, diode, and various types of autonomous sources. Arbitrary differential and algebraic equations can be included as user-defined components.

```
struct sc_a_component {
    virtual void build(void) = 0;
    virtual void BuildM(void) = 0;
    virtual void BuildB(void) = 0;
    virtual void BuildRhs(void) = 0;
};
```

![Fig. 1. SystemC-A analogue components inheritance hierarchy.](image)

C. Equation Build Method

The build method supports the automatic equation formulation of the system to be modelled. It is a virtual method in the abstract component base class (\texttt{sc\_a\_component}) and inherited by all derived components. The build method consists of two functions, BuildM() and BuildB(). They contain C++ code which defines one or more DAEs. Fig.2 illustrates the use of the build functions in a capacitor model as a SystemC-A component. Also, it gives a better understanding of the elements of the build method and how the modeller can use them.

BuildB() represents the component’s contribution to the Jacobian, the associated function is Jacobian(). It is a function to add a single contribution to the Jacobian matrix according to Modified Nodal Analysis (MNA). For instance, the capacitor contribution needs four entries to the Jacobian, therefore four calls to the Jacobian() function is performed in the capacitor model shown in Fig.2. The arguments of Jacobian() are, first the two nodes to where the capacitor is connected and the third is the contribution element of its stamp.

BuildM() represents the component’s contribution to the RHS vector of the system equation. The associated function with BuildM() is BuildRhs() function. It is a function to add a single contribution to the RHS of the system equation. The arguments of BuildRhs() are, first the corresponding capacitor node and the second is the contribution element of its stamp.

For the capacitor model the build method is implemented as:

```
void sc_a_capacitor::BuildB(void){
    Jacobian(b,b,-S*C);
    Jacobian(a,b,-S*C);
    Jacobian(a,a,S*C);
    Jacobian(b,a,-S*C);
}
```

```
void sc_a_capacitor::BuildM(void){
    S=S();
    Jacobian(a,a,S*C);
    Jacobian(b,a,-S*C);
    Jacobian(b,b,S*C);
}
```

```
void sc_a_capacitor::BuildRhs(void){
    BuildRhs(a,-C*Vdotn);
    BuildRhs(b,C*Vdotn);
}
```

![Fig. 2. Capacitor mathematical model and its SystemC-A build functions.](image)

\[ \partial \text{part of capacitor’s SystemC-A model} \]

The system’s objective is to attenuate low frequency vibrations arising from road surface disturbances. It consists of three main parts, the plant (i.e. the passenger seat and vehicle chassis), electromechanical actuator, and controller as shown in Fig.3. There are two sensors which monitor the seat and chassis and hence generate input signals to the controller. The control signal is connected to an active electromechanical actuator which is a force generator introduced to improve ride quality.

A. Mathematical model of chassis and seating system

As shown in Fig.3 the vehicle’s mass \( M_s \) and passenger/seat mass \( M_p \) are separated by a passive spring \( K_s \) and damper \( C_s \). The seat is further isolated from the chassis by a force actuator in parallel with the spring and damper. An external displacement \( x_d \) represents the system input and acts through a passive spring \( K_e \) and damper \( C_e \). The equation of motion for the seat can be written as:

\[ \begin{bmatrix} v_{\text{a}} & v_{\text{b}} \end{bmatrix} \text{RHS} \]

\[ \frac{d^2 v_{\text{a}}}{dt^2} = -SC v_{\text{a}} - C X_{\text{a}} + SC v_{\text{b}} + C X_{\text{b}} \]

\[ \Delta v_{n+1} = \begin{bmatrix} SC & -SC \\ -SC & SC \end{bmatrix} \]

\[ \begin{bmatrix} V_{\text{a}} \\ V_{\text{b}} \end{bmatrix} \]
The actuator is an electromechanical hydraulic design, which operates in parallel with springs and dampers [9]. It consists of a DC motor, some mechanical parts (such as the gear trains and rack) and a hydraulic vibration absorber. The actuator input from the controller is a DC voltage \( e_a \), which drives the motor to output a rotational torque \( T_m \). The gear train converts the rotational velocity of the rack, whose displacement impacts the pressures of the upper and lower chambers of the hydraulic piston. The vehicle’s chassis is attached to the hydraulic cylinder’s piston rod and the seat sits on the cylinder cap. The actuator force is dependent on the pressure difference between the upper and lower chambers \( \Delta P \). The generated force attenuates the vibrations by acting on the vehicle’s chassis and passenger’s seat. The mathematical equations which describe the model of the actuator are explained briefly. The equations are from electrical, mechanical and hydraulic domains. The DC motor develops a torque \( T_m \) which is proportional to the armature current \( i_a \), where \( K_t \) is motor-torque constant.

\[
T_m = K_t i_a \tag{5}
\]

When the armature rotates, a back emf is induced and is proportional to the flux and angular velocity as defined in Eq. 6, where \( K_b \) is constant.

\[
e_b = K_b \frac{d\theta_{g1}}{dt} \tag{6}
\]

Applying Kirchoff’s voltage low to the electric circuit of the armature,

\[
L_a \frac{di_a}{dt} + R_a i_a + e_b = e_a \tag{7}
\]

Applying Newton’s law for the input rotational system dynamics \((J_m)\),

\[
J_m \frac{d^2 \theta_{g1}}{dt^2} + b_m \frac{d\theta_{g1}}{dt} + T_{g1} = T_m \tag{8}
\]

Ideal gears are assumed by neglecting friction losses and gear mass, hence the equations of input and output angular velocities and torque can be written as in Eq. 9.

\[
\frac{\omega_{g1}}{\omega_{g2}} = \frac{r_{g1}}{r_{g2}} = \frac{n_{g1}}{n_{g2}} \quad \text{and} \quad T_{g1}\omega_{g1} = T_{g2}\omega_{g2} \tag{9}
\]

Also, applying Newton’s law to the load shaft \((J_{l2})\) gives Eq. 10.

\[
J_l \frac{d^2 \theta_{g2}}{dt^2} + b_l \frac{d\theta_{g2}}{dt} + T_l = T_{g2} \tag{10}
\]

The rack’s linear velocity \((V_{r2})\) can be determined as in Eq. 11, and \( V_{r2} = V_{r1} \) since the two hydraulic pistons are connected.

\[
V_{r2} = \omega_{g2} r_{g2} \tag{11}
\]

The torque on the load shaft \((J_{l2})\) is determined in Eq. 12,

\[
T_L = A_{r2} (P_{2} - A_{r1} P_{1} r_{g2}) \tag{12}
\]

The hydraulic pressure in the upper and lower actuator chamber are described in Eq. 13 and 14.

\[
\frac{dP_1}{dt} = A_p V_{rel} - A_{r1} V_{r1} \left( \frac{\beta_1}{U_1} \right) \tag{13}
\]

\[
\frac{dP_2}{dt} = -A_p V_{rel} + A_{r2} V_{r2} \left( \frac{\beta_2}{U_2} \right) \tag{14}
\]
C. Controllers

In order to attenuate the vibrations, the system was tested with three types of controllers, PI, Variable structure and optimal controllers. The inputs to the controller are the dynamic seat and chassis motions i.e., the displacement, velocity, and acceleration of the passenger seat \( (x_s, v_s, a_s) \), the displacement and velocity of the vehicle chassis \( (x_c, v_c) \). Any single controller may have any set of inputs. The output of the controller is the voltage sent to the DC motor \( (e_a) \). Please refer to reference [6] for full controllers designs.

1) Proportional- Integral controller: The PI controller is used to act on the error between the seat’s set-point acceleration and the actual value \( (e = a_p - a_s) \) [10]. Through the integral operator the PI controller brings quickly the error signal to zero. Eq. 15 defines the PI controller equation, where \( K_p = 13.5Vs^2/m \) and \( K_i = 0.27Vs/m \) [6] are proportional and integral gains respectively. They are selected using an analytic and trial/error process.

\[
e_a = K_p e + K_i \int e \, dt \tag{15}
\]

2) Variable-Structure controller: Variable structure controller relays on a high-speed switching feedback strategy to establish a robust control for uncertain plant model [11]. The switching control algorithm drives the plant’s state trajectory to a user selected sliding surface and then maintains the trajectory at that surface. The sliding surface is chosen such that the system motion exhibits the desired stability and/or tracking characteristics. The first step in the controller design is to select a sliding surface, hence \( S = [1.72, 262.4, 4.5, 65.7] \) [6]. The second step is to compute the controller’s feedback gains which derive the plant trajectories to the sliding surface. The full state feedback has the form of \( u = K x = k_1 x_1 + ... k_6 x_6 \). The gains are selected as \( k_1 = -94.8N/m, k_2 = -5700N/m, k_3 = -1440Ns/m, k_4 = 437Ns/m \) [6].

3) Optimal controller: The optimal controller is based on full state feedback Linear Quadratic Regulator LQR. LQR controller minimizes a specified linear quadratic performance index [12] defined in Eq. 16, where \( F \) and \( R \) are the state and input weighting matrices, respectively. The performance index is selected based on balance tradeoffs between convergence speed to the system states and the input amplitudes.

\[
J = \int_0^\infty (x^T F x + u^T R u) dt \tag{16}
\]

The controller gains are \( K_1 = 116N/m, K_2 = 6520N/m, K_3 = 1670Ns/m, \) and \( K_4 = 460Ns/m \) [6].

IV. SystemC-A Models

The automotive system was modeled in SystemC-A making use of the modular modeling where the main parts of the system (the plant, actuator and controller) were described as individual modules at behavioral level together with a test-bench. The test-bench includes instances of all parts connected together by signals as illustrated in Fig. 4. The input stimulus is generated by a sinusoidal voltage source from the SystemC-A library of circuit components. The stimulus is then connected to the chassis and seating module via nodes as shown in Listing 1.

Listing 1. SystemC-A testbench of the automotive vibration isolation system.

```c
void testbench::system()
{
  // global connecting signals
  sc_signal< double > Vrel, as, deltaP, ea;
  // instantiating nodes and components
  n2 = new Node("0");
  n1 = new Node("n1");
  sc_a_voltageS_sin <I> = new sc_a_voltageS_sin("Vs1", n1, n2, 5.125,0,1,0,0);
  seating s1 = new seating("s1",n1,&deltaP,&Vrel,&as);
  actuator sact1 = new actuator("act1".,&Vrel,&ea,&deltaP);
  PI spil = new PI("pil",&ea,&as);
  sc_start(2.5,SC_SEC); \// start simulation for 2.5Sec
}
```

An example of SystemC-A model of one of the system parts is shown in Listing 2 of the chassis and seating module. The seating is modelled as SystemC-A component [1]. In Listing 2, the component’s constructor defines the number and type of the component’s inputs and outputs for each controller. Also in the constructor, system variables are defined as well as the system constants. BuildB() functions include DAEs of the system which hint that the system modeled at behavioural level. The chassis and seating module contains 4 system variables which need 4 Equation() functions to be defined in BuildB(), while the actuator has 13 system variables and needs 13 Equation() functions to be defined.

Listing 2. SystemC implementation of chassis and seating.

```c
... seating :: seating (char nameC)[5], SystemVariable <node_a ,
  sc_signal< double >deltaP1 , sc_signal< double >Vrel1 ,
  sc_signal< double >as1 > component (nameC, node_a , 0,0){
  deltaP1=deltaP1;
  Vrel1=Vrel1;
  as1=as1;
  xQ = new sc_free_variable("xQ"); \// system variables
  xQ = new sc_free_variable("xQ");
  yQ = new sc_free_variable("yQ");
```
A. Comparison with VHDL-AMS

The model of the vibration isolation seating system was modelled and simulated in VHDL-AMS [7]. SystemC-A shows accurate results by comparing its simulation performance of $x_{dp}$ with VHDL-AMS results in Table I. The maximum relative percentage differences in the maximum value of seat position $x_{dp}$ between SystemC-A and VHDL-AMS simulations was approximately 0.860% for the optimal controller under single jolt sin wave, and it was 2.642% for the passive system under multiple sin waves with WGN simulation.

V. CONCLUSION

A model of an electrical-hydraulic-mechatronic automatic seating vibration isolation system has been developed in SystemC-A HDL. This has demonstrated the ability of SystemC-A to model highly complex mixed-signal and mixed-domain systems in HDL context. In the case study’s results, several controllers have been used and tested for two different types of stimuli, a single jolt sin wave and a multiple sine waves with WGN. Among the different types of controllers the optimal controller was the best one to attenuate external disturbances of the two types used. Further, SystemC-A simulations were compared to published VHDL-AMS simulations, showing highly comparable numerical figures, which proves that SystemC-A can be compared to well established HDLs.

REFERENCES

TABLE I
SYSTEMC-A AND VHDL-AMS PERFORMANCE FIGURES OF THE SEAT POSITION \( x_{sp-p} \) (CM) FOR THE PASSIVE SYSTEM AND THE SUITE OF CONTROLLERS.

<table>
<thead>
<tr>
<th></th>
<th>( x_{sp-p} ) (CM)</th>
<th>% Relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sin wave</td>
<td>sin waves with WGN</td>
</tr>
<tr>
<td>Passive system</td>
<td>3.052</td>
<td>3.073</td>
</tr>
<tr>
<td>PIC</td>
<td>1.510</td>
<td>1.519</td>
</tr>
<tr>
<td>VSC</td>
<td>0.683</td>
<td>0.687</td>
</tr>
<tr>
<td>OC</td>
<td>0.574</td>
<td>0.579</td>
</tr>
</tbody>
</table>

Fig. 6. Multiple noisy sinwaves stimulus disturbance with simulation responses of the three controllers.


