A Framework for Modeling and Concurrently Simulating Mechanical and Electrical Faults in Verilog-AMS

Francesco Tosoni[*], Nicola Dall'Ora[*], Enrico Fraccaroli[†], Franco Fummi[*]
[*]Department of Computer Science – University of Verona (Italy), name.surname@univr.it
[†]Department of Computer Science – University of North Carolina at Chapel Hill (USA), enrifrac@cs.unc.edu

***Abstract*—There are several languages for modeling a Cyber-Physical System (CPS). One of them is Verilog-AMS, which allows representing a system belonging to the electrical and mechanical physical domains in a single model through different disciplines. A framework for the automatic fault injection in the electrical and mechanical domains is proposed in this context. In particular, starting from a mechanical system, it is possible to represent it as an electrical circuit by exploiting the physical analogies. In the electrical domain, fault modeling and injection techniques are more advanced than in other physical domains. Extending the analogies to fault models makes it possible to apply the electrical fault models in the equivalent circuit to the mechanical system. These yields mechanical-level faulty behaviors, which can be injected into the mechanical domain, resulting in mechanical (physical) faults, depending on the component. It is finally shown an example of execution of this flow through a model of an electric motor, in which mechanical faults are injected. Simultaneously, the equivalent electrical faults are injected into the equivalent electrical circuit.**

***Index Terms*—Cyber-physical systems, Safety, Fault modeling, Fault injection, Verilog-AMS.**

I. Introduction

The industrial field undergoes continuous and rapid evolution, granted by increasing precision of the design and simulation of the models. As the complexity of these models grows, increasing accuracy is required to be effective for industrial production. As part of the functional safety of industrial machinery, simulating these systems is essential to ensure their proper operation. Once the simulated model is built, fault injection is performed to understand how the behavior of the system, or individual component, changes with respect to a given fault. This procedure, called *fault injection*, highlights new vulnerabilities or flaws that could compromise a system's safety. However, in the context of CPSs (*i.e.*, multi-domain systems), how to perform *fault injection* differs depending on the part of the system under analysis. Considering an electro-mechanical system, the fault models injected into the mechanical part will differ from purely mechanical ones. Moreover, techniques change also between different physical domains (e.g., electrical, mechanical, thermal). Overall, *fault injection* in analog and digital electrical models is the state of the practice to ensure functional safety, as mentioned by the ISO 26262 standard for automotive functional safety [1], [2].

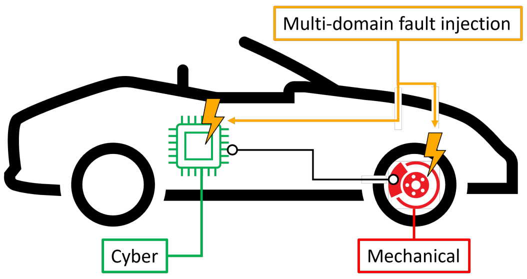

Fig. 1. Cyber-Physical System locations in which the proposed approach can act to automatically inject multi-domain faults by exploiting the potentialities of the Verilog-AMS language.

Therefore, unlike the electrical domain, there is no large use of fault injection techniques in the mechanical one at the state of the practice. Primarily because of the difficulty in categorizing mechanical components and the working conditions of the machinery. Extending the fault injection to the mechanical field could improve the evaluation of safety mechanisms. Fortunately, there are some *analogies* between systems which belong to different physical domains where they use the same differential equations. The following sections show that there are *analogies* between mechanical and electrical systems [3]. This paper aims to harness these analogies to model faulty behaviors of a generic system, regardless of the underlying physical domain. In this way, it would be possible to use the advanced fault injection techniques of the electrical domain in others, *e.g.*, the mechanical one. Figure 1 shows an example of a faulty CPS, where different types of faults can lead to altering the assisted braking functionality of a motor vehicle in many ways.

This paper proposes a flow for fault injection in multi-domain systems using Verilog-AMS:

- Describe a mechanical system as a circuit to inject electrical faults and study its behavior in terms of mechanics;
- Identify mechanical fault models and inject them directly into the mechanical system;
- Verify the equivalence of electrically injected faults to mechanic's faults, forming a taxonomy.

There are many tools to represent and simulate physical systems: one of them is Verilog-AMS, which is a Hardware Description Language that can describe the behavior of a model through differential equations. Such models can be used also for the fault injection, *i.e.*, changing the equation system.

The paper is organized as follows: Section II describes the background and the state-of-the-art. Section III shows how to model mechanical systems and faults by exploiting the analogies. In Section IV faults are derived from the electrical to the mechanical domain to build a mechanical fault taxonomy. Then, in Section V an automatic tool to perform fault injection into Verilog-AMS multi-disciplinary models is proposed. Furthermore, an automatic simulation flow for SPICE-based simulators is presented. Conclusions are drawn in Section VI.

II. Background and state of the art

This section describes the potentialities of Verilog-AMS as a modeling language. Then, it show the state-of-the-art of multi-domain fault injection in electrical and mechanical components. Finally, it describe how to represent a non-electrical model as an electrical equivalent one.

A. Systems Modeling through Verilog-AMS

The Verilog-AMS language is the last extension of the Verilog language. The extension is created for combining the *digital* and the *analog* part. The capabilities of Verilog-AMS are equal to the VHDL-AMS language, and they share the same functionalities [4]. Moreover, the communication between the *digital* and the *analog* part is possible through pre-defined language functions. With this features analog models could communicate with digital designs through apposite functions, *e.g.*, `timer()`, `cross()`. For example, these functions allow to activate a timer inside an analog design or to activate a crossing routine when the monitored analog function crosses the zero value of magnitude.

Furthermore, the Verilog-AMS language defines constructs to model systems from different physical domains. In particular, electrical, mechanical, and magnetic are the principal domains covered. It is also possible to define models that combine multiple domains, *e.g.*, by modeling electro-mechanical systems.Conservative domains are supported by defining the *potential* and *flow* variables. For each physical domain, a *discipline* and several *natures* are defined. For example, the electrical domain defines as natures the voltage and the current, accessible through the functions `V()` and `I()`. If required, it is also possible change the type of the pre-defined *potential* and *flow* variables. The behavior of conservative systems is described by specifying how the natures of each domain vary over time. It is easy to define a conservative network by using the `branch` statement to create a link between a pair of nodes. Commercial SPICE-based simulators can simulate behavioral Verilog-AMS models efficiently. These simulations can be managed through ad-hoc testbenches written in SPICE-based code, *e.g.*, Eldo and Spectre.

B. Multi-domain fault modeling and simulation

Faults represent wrong behaviors of a system that can happen for multiple reasons like device aging, the breaking of an internal component, a digital failure, or a production defect. The functional safety studies how it is possible to guarantee the functionalities of machinery in the presence of failures, by exploiting different techniques. One of these techniques is *fault injection*, formally described in the Standard ISO 26262 [1] for the electrical domain [5]. A *fault* is a *saboteur* if it consists of a new component that alters circuit behavior by injecting a new set of equations describing the faulty behavior. The fault is a *mutant* when it mutates an existing component, *e.g.*, it changes the nominal value of a parameter or models an abnormal behavior in the presence of faults. In contrast to the mechanical domain, several failure models are part of the state of practice in the electrical world. There are well-defined fault models and fault injection techniques that specify where and how to inject them into a transistor-level circuit [2].

Instead, the fault simulation techniques are not advanced as the electrical ones in the mechanical domain. Fault taxonomies that consider the physical aspect of mechanical systems are listed in [6], [7]. However, it is possible to model these physical faults only if the model considers the geometry of the system and the behavior around it. By abstracting from the physical level to the behavioral level in the mechanical domain, the number of works that analyze multi-domain faults remains limited. It is complex to represent faults suitable for any physical model and any level of abstraction. Moreover, a fault modeled in a simulated environment is unlikely to cover all possible physical variations in the real world. The accuracy of these fault injection techniques concerning real-world physics depends on the model's level of detail. For example, a minimum subset of details to model a mechanical rotational model considers the force and the angular velocity. There are studies describing the injection of mechanical failures using specific simulation tools [8]–[10]. All these works propose pre-defined libraries with different classes of faults modeled as blocks. The main limitation of previous approaches is the difficulty of reproducing the alterations with pre-defined blocks, like the Modelica libraries. It is not easy to systematically change the internal dynamics with a procedure, and thus it is challenging to alter with faults. The proposed framework injects different classes of faults into models described through differential equations by exploiting the potentialities of the Verilog-AMS language.

C. Systems modeling through the electrical analogies

Equivalent electrical circuits have been used increasingly over the years in both industry and research to represent the behavior of complex systems. For this reason, several analogies have formed between physical domains like thermal, magnetic, and hydraulic, and especially with the electrical one [11]. Researchers have used electrical circuits to mimic other physical systems for a while now. For instance, to model characteristic parameters variability in automotive batteries [12], internal combustion engine [13], ocean wave power takeoff [14], partial differential equations [15] and so forth.

This approach can also represent nonlinear dynamical systems by approximating their behavior. Hence, analogies are useful to model a translational/rotational mechanical system as an electrical circuit to exploit its characteristics. Through analogies, it is possible to link physical quantities of a specific domain to the ones of another. Analogies are founded on the concepts of conservation of energy laws [16], [17]. In particular, the cited analogies between the electrical and mechanical domains are the *force-voltage* and the *force-current* analogy [18]. The *force-voltage* analogy is considered the easiest to use, while the *force-current* analogy is more conservative with respect to system structure. Once a system is transformed into its electrical equivalent, fast and accurate model simulations are achievable with a modern simulator (*e.g.*, Spectre and Eldo). Alternatively, it is possible to simulate these models through multi-physics simulators (*e.g.*, Matlab Simulink/Simscape, Siemens AMEsim). However, using an Hardware Description Language (HDL) simulator allows the combination of the equivalent electrical model with accurate digital systems without doing co-simulation using standards like Functional Mock-up Interface (FMI). In the following section, the electrical analogies will be exploited to relate the mechanical to the electrical domain and analyze the effect of the faults.

III. Modeling mechanical models and faults through analogies

The physical analogies introduced in the previous section have become important also because of the rising knowledge of the electrical world. The goal is to exploit the mastery of electrical principles for other scientific domains like the mechanical one. In particular, they are widely used in the mechanical field because it is easier to describe and manipulate an electrical description than a mechanical one [3]. The same applies to simulation. These analogies are defined at the mathematical level: an analogy can be established between two domains if the variables and the mathematical laws that define them are mathematically equivalent. Therefore, two systems belonging to different domains can be said to be equivalent if the differential equations defining them are equivalent. For instance, it is possible to describe a mechanical system through an electrical circuit simply by transforming the equations of the mechanical system into electrical ones. This is accomplished by translating the variables of the mechanical domain into electrical variables, following the relationships between physical quantities defined by analogy itself. Over the years, simulations of physical systems, especially electrical circuits, have been performed via SPICE-based simulators. In this domain, each component, defined on a branch in Verilog-AMS, is described through its own differential equation. Combining all branches using nodes, it is possible to simulate all equations that make up a circuit by exploiting Kirchhoff's laws. Over time, other simulators have been created for the other physical domains although SPICE-based one is still more efficient for the electrical domain.

Now, let us see how the analogies between the electrical and mechanical domains are defined. Focusing on the mechanical domain, the two main analogies between this domain and

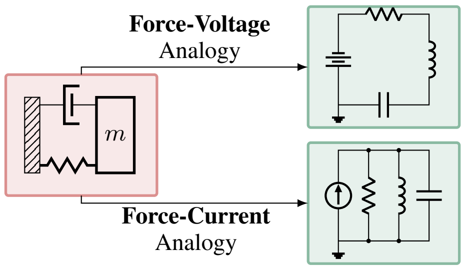

Fig. 2. Analogies between mechanical (left) and electrical (right) system.

the electrical domain are the *force-voltage* (also known as impedance analogy) and the *force-current* (also known as mobility analogy) [17]. Figure 2 shows the two types of analogies between mechanical and electrical domains. The *force-voltage* analogy is considered a "direct" analogy since it is more intuitive than the other. Nevertheless, it still requires an intermediate representation (mechanical network) to go from the mechanical domain to the electrical domain. However, no analogy is superior to the other: the choice to use one instead of the other is arbitrary because the results are consistent in both cases. The differences are in the association of the variables that control each domain: in the *force-voltage*, force is associated with voltage, while velocity is equivalent to electric current. These physical variable relationships are shown in Table I. In the *force-current* analogy, as the name suggests, the opposite holds. The *force-voltage* analogy preserves the analogy between electrical impedance and mechanical impedance, whereas the other analogy does not. On the other hand, the *force-current* analogy preserves the topology of the mechanical system when transferred to the electrical domain, whereas the *force-voltage* analogy does not. However, all laws of circuit analysis, such as Kirchhoff's circuit laws, apply in the electrical domain to both these analogies. These analogies become even more effective when analyzing electro-mechanical systems, such as a DC motor, modeled through mechanical and electrical parts. The electrical part controls the motor in voltage, and through an Electromotive Force (EMF) we can move the shaft, *i.e.*, the mechanical part of the motor. If the whole system is represented electrically by analogy, it would be sufficient to use an all-electrical simulation instead of having both an electrical and mechanical simulation. The work proposed in this paper is based on the *force-voltage* analogy because it turns out to be the most intuitive.

The *force-voltage* analogy is used with the aim of representing a mechanical system through an electrical circuit: since each variable of a domain is mapped one to one in the other, each mechanical component corresponds to an electrical one. These relationships, described in Table I, are obtained mathematically, starting from the variables of effort and flow, respectively mechanical force and speed, electrical voltage and current. For example, a damper is the equivalent of a resistor since both represent a loss of energy flow in their domains. Other relations are that the mass is equivalent to an inductance, a capacitor to a spring *etc*. All of these relationships highlight how it is possible to describe a multitude of different mechan-

TABLE I
RELATIONSHIP BETWEEN MECHANICAL AND ELECTRICAL VARIABLES IN THE FORCE-VOLTAGE (IMPEDANCE) ANALOGY.

Type	**Mechanical translation**	**Mechanical rotation**	**Analogous electrical**
Effort Variable	Force	Torque	Voltage
Flow Variable	Velocity	Angular velocity	Current
Elements	Damping	Rotational resistance	Resistance
	Mass	Moment of inertia	Inductance
	Compliance	Rotational compliance	Capacitance
	Mechanical impedance	Mechanical impedance	Electrical impedance

ical systems. Hence, a 1D mechanical system can be modeled with passive electrical components by exploiting the analogy. Let us now consider a real electro-mechanical system: the DC motor.

A. Running example: DC motor with connected wheel

The DC motor is an *electromechanical system*, composed by the electrical actuation part and the mechanical part. It is choosen as running example because is widely used in the automotive and factory fields. The system considered as running example is showed in Figure 3. The DC motor can be modeled with an equivalent circuit of the armature and the free-body diagram of the rotor. In details, the electrical part is composed by a voltage source that drive the current flow, a resistance connected in series with an inductance that model the internal characteristics of the internal motor coils. Then an electromotive force (emf) is used to model the conversion of the energy from the electrical to the mechanical domain. The mechanical rotational part model a shaft with own inertia connected to a wheel with a fixed inertia.

The following constitutive relations model the dynamic of the motor connected to a wheel:

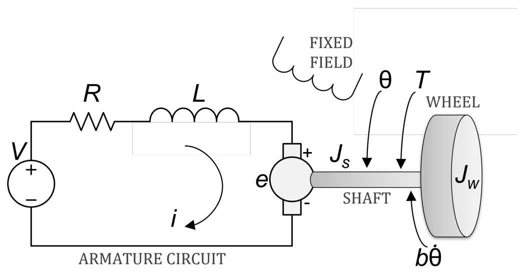

Fig. 3. DC motor: the left part show the electrical circuit used to control the motor with a potential source, while the left part shows the shaft connected to a wheel. The EMF convert the energy from the electrical to the mechanical domain.

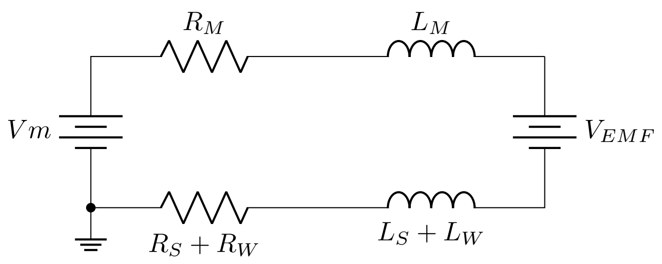

Fig. 4. Full electric description of the DC motor plus one wheel.

$$v = K_M \cdot \omega + R_M \cdot i + L_M \cdot \frac{di}{dt}, \quad (1)$$

$$\tau = K_T \cdot i - (d_S \cdot \omega + d_W \cdot \omega) - (j_S \cdot \frac{d\omega}{dt} + j_W \cdot \frac{d\omega}{dt}), \quad (2)$$

where v is the input voltage source applied to the motor's windings to control the velocity of the rotor; variable i represents the current through the windings; variable ω is the angular velocity of the shaft; variable τ is the torque on the shaft. K_M and K_T are motor coefficients used to dimension the size of the motor: coefficient K_M is used to compare motors and to calculate temperature rise based on the dissipated power, while coefficient K_T is used to calculate the required current based on the required torque. The rotor and the shaft are assumed to be rigid, and the friction torque is proportional to the shaft angular velocity. According to the analogy, the inertia (j_S and j_W) and friction (d_S and d_W) of the shaft and the wheel in Equation (2) are equivalent to a capacitor (L_S and L_W) and resistance (R_S and R_W) respectively.

In order to represent the whole system in the electrical domain, it can be seen as a mechanical network [19], [20]. This intermediate representation between the two domains is necessary because the *force-voltage* analogy is used: in fact, it does not preserve the topology of the system by constructing the equivalent circuit. The mechanical network consists of the mechanical components embedded in the branches, while the nodes represent the displacements. If a component is subject to a certain displacement, then it will be connected to that node. Doing so represents a mechanical system in a way much more similar to an electrical circuit. Indeed, it is much easier to construct the equivalent electrical circuit by applying the analogy from this representation than from the original system. The electrical circuit obtained from the Equation (1) and (2) through the analogy (shown in Figure 4) consists of the same electrical section of the system, while the mechanical part is translated into its electrical equivalent. The motor has been described with an electromotive force opposite to the supply force named V_{emf} (exerted by the motor itself), a resistance (R_S) and an inductance (L_S), which are due to shaft connection. The latter two components were increased because of the friction and inertia also exerted by the wheel (R_W and L_W).

TABLE II
MECHANICAL FAULT TAXONOMY.

Fault	Effect
Galling, Seizure	Surfaces are damaged by sliding over each other due to different speeds and temperatures. Seizing: the two surfaces no longer separate
Creep	Plastic deformation due to prolonged stress and/or temperature changes
Spalling	Splintering of a surface, deteriorating the function of the component
Rupture	Multi-part component breakage
Backlash	System motion conditions modified by gaps between parts
Impact	High force or shock applied over a short time period when two or more bodies collide
Fatigue	Initiation and propagation of cracks in a material due to cyclic loading or impact
Buckling	Deformation due to a load or force exerted parallel to the axis of the body
Wear	Damage, gradual removal or deformation of material on solid surfaces. Causes of wear can be mechanical or chemical

TABLE III
MECHANICAL FAULT TAXONOMY INFERRED FROM THE ANALOG FAULT INJECTION IN THE EQUIVALENT MODEL.

Mechanical Behavior	Mechanical Effect	Mechanical Fault	Electrical Fault Equivalent
Brake/ friction	Added/increased braking force on failed component	Galling/Seizure, Creep, Spalling, Wear/Corrosion	Open
Disconnection	The failed component detaches from the system	Rupture, excess of Backlash	Short
External force	An abnormal (external) force affects the component	It can cause deformations or cracks or ruptures (from impact of fatigue)	Voltage source
Limited movement	The direction of displacement of a component is abnormally modified	Rupture, Deformation, Wear	Current source
Parametric	Intrinsic characteristics of the failed component altered	Wear (component aging) and all the parameters changes	Parametric

IV. MECHANICAL FAULTS

Now that the *force-voltage* analogy has been illustrated, let us proceed to discuss fault injection. For several years, the list of widely accepted fault models for the electrical domain was restricted to open circuit, short circuit, source, and parameter deviation. However, a few years ago, a working group started drafting what today is known as the IEEE P2427 standard [5]. This working group is trying to standardize defect modeling, simulation, and coverage metrics for analog and mixed-signal circuits. Specifically, the IEEE P2427 standard suggests that analog faults must be injected one at a time at a single point in the circuit to determine the behavior of each fault. In the mechanical domain, instead, faults are usually classified in relation to the physical characteristics of each component [7]. Usually, in the existing taxonomies a possible cause (*e.g.* an overload or an unexpected shock) and the failure mode (*i.e.* how the component fails) for each fault are specified. One such classification is shown in Table II, where the fault model and its mechanical effect are detailed. It is evident that the effects primarily involve changes in materials, component geometries, shapes, and dimensions.

This paper focuses on the *behavior* of mechanical systems; therefore, here, we investigate a methodology that relies entirely on the *behavioral-level* of abstraction. By representing mechanical systems as mechanical networks and describing their behavior using equations, a fault taxonomy closer to this level was researched. This middle representation is also useful for performing fault injection, and then fault simulation, in terms of the differential equations of each node or branch of the mechanical network. Consider the modeling and fault injection process in the electrical domain: they are very similar to the one required by the proposed procedure, since the injection is performed into single branches in both cases. Moreover, the mechanical network is very similar in construction to an electrical circuit, since both are formed by nodes and branches, which have their own equation. Using analogies for this purpose appears to be a good option, which would allow to have the mechanical behavior described by an electrical circuit.

However, physical analogies do not provide any information about failures: it is not always granted that electrical fault models have a correspondence in the mechanical world. In fact, behavior is mathematically equal across domains, but that doesn't mean it has to be functionally equal. Because of that, in order to define a possible mapping between electrical and mechanical faults, there is a need for testing and comparing their behavior. In particular, it is necessary to simulate the equivalent faulty electrical circuit, using electrical fault injection techniques, and to study the obtained behavior from the mechanical point of view. If the behavior has a meaning from both electrical and mechanical point of view, then there is a correlation between faults belonging to different domains.

The *open circuit* fault represents an arbitrarily valued resistor injected into a line of the circuit. Ideally, with a very high value, the fault simulates a line break. For example, the electrical open fault is equivalent to a mechanical braking agent because both, when injected, significantly reduce the flow variable, *i.e.*, velocity and electric current. We can think of this failure as an increased coefficient of friction on the motion surface due to temperature, wear, or the presence of debris. Similarly, the other electrical faults at the behavioral level have been applied: shorts and voltage and current sources. The *short circuit* fault consists of an unwanted branch between two points of the circuit that originally was not supposed to touch each other. This fault can be assimilated to a disconnection of certain mechanical components from the rest of the system, such as springs or dampers. The separation of a component from the rest of the system could happen if it is affected by excessive backlash or rupture. Voltage and current *sources* represent unwanted changes in voltage and current at a given point in the circuit. This changes might be caused by interference coming from surrounding electrical circuits, or by alpha particles coming from outer space hitting a portion of the circuit. According to the analogy, force is equivalent to voltage: therefore a voltage source can be seen as an unexpected external force on a certain component. Similarly, velocity is equivalent to electric current: therefore, a current source is a variation of the displacement velocity of the component in which it is injected. As for *parametric faults*, they are equivalent in both domains. More details on this fault classification and correlation between the two physical domains can be found at [21], [22]. This extension of the analogy is shown in Table III: in particular, the injected electrical fault models and the detected equivalent mechanical behaviors are shown. In Table II there are some mechanical failures that can be assimilated to the faulty behaviors obtained through the simulation [6], [7].

Table III has been derived mainly from the simulation of several mechanical systems through electrical analogy: a mass-spring-damper, a tuned mass-spring-damper, a double pendulum, and a DC motor, presented in the previous section. The proposed taxonomy has been built to show how mechanical fault behaviors can be derived by simulating faulty electrical equivalent systems. Thus, this fault taxonomy could be extended by simulating more complex mechanical systems. The next section will illustrate the procedure of injection and fault simulation exemplified on the DC motor system.

V. Automatic fault injection and simulation tool

Let us then take the DC motor presented in the previous sections (Figure 3) as an example: as we have seen it is composed of two main parts, one electrical and one mechanical. The electrical part provides power to the electric motor itself, feeding the power supply, while the mechanical section is composed by the shaft and a wheel, directly connected to the motor. In particular, it moves the wheel, which is connected directly to the shaft: these two mechanical parts cause resistances to the torque exerted by the motor, such as rotating friction and inertia. The system has been described through the HDL Verilog-AMS and simulated using the techniques and tools explained in the previous sections. The DC motor plus wheel model is shown in Listing 2.

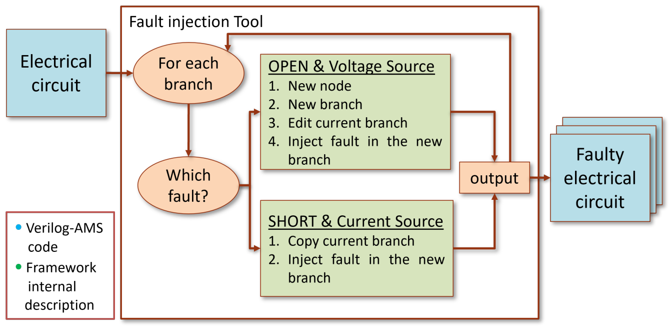

Fig. 5. Framework flow.

As we anticipated, the system was translated to electrical via the *force-voltage* analogy: the already electrical part remained unchanged, while the mechanical part was modeled as shown in Figure 4. This circuit was modeled in Verilog-AMS and is displayed in Listing 3. Note that the branches `Rsw` and `Lsw` represent the combined values of the shaft and wheel resistors and capacitors, according to Equation (2) and Figure 4.

Once the electrical equivalent is obtained, we are ready to inject the electrical faults into the new circuit. As already explained, faults are injected one at a time, at only one point in the circuit, following the P2427 standard. An example of an open fault injected into the DC motor is shown in Listing 4. Faults are automatically injected using HIFSuite tool [23], which allows manipulation of various HDL descriptions, including Verilog-AMS. The flow is presented in Figure 5: the tool is able to take any circuit described in Verilog-AMS and translate it into XML as intermediate modeling for simpler fault injection. Then, depending on which fault we are injecting, there is one procedure for faults injected in series to the branch (*i.e.* open and voltage source) and one for faults injected in parallel (*i.e.* short and current source). Finally, all faulty circuits, according to the electrical fault models, are returned by the tool in their original HDL language. Once the automatic generation of the faulty models is performed, so all the faulty circuits are obtained. Regarding simulation, the process is handled by a testbench module, which instantiates the fault-free model (see Figure 6) and the faulty models through the `alter` command as shown in Listing 1. The comparator receives all the simulation output from the fault-free and faulty models, then it provides as output the fault pattern detected and at what point in the time. In this way,

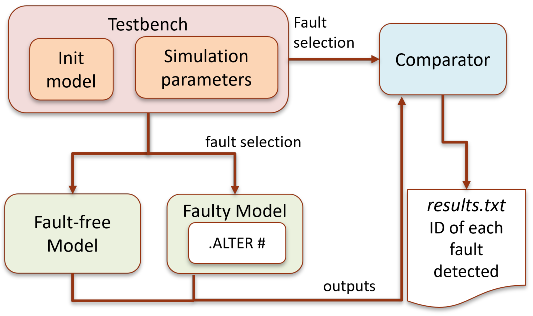

Fig. 6. Simulation flow.

TABLE IV
AVERAGE SIMULATION TIME AND NUMBER OF INJECTED FAULTS IN THREE DIFFERENT SYSTEMS.

Faults Data		RLC	Tuned-RLC	Double Pendulum	DC motor (mixed)	DC motor (full elec.)
Fault-free simulation time		570 ms	300 ms	190 ms	230 ms	240 ms
OPEN	No. injected faults	4	6	6	3	6
	Avg. simulation time	587 ms	241 ms	208 ms	250 ms	250 ms
SHORT	No. injected faults	12	30	-	12	30
	Avg. simulation time	610 ms	309 ms	-	240 ms	240 ms
Voltage Source	No. injected faults	4	6	6	3	6
	Avg. simulation time	4.8 s	1.28 s	1.5 s	450 ms	530 ms
Current Source	No. injected faults	12	30	30	12	30
	Avg. simulation time	1.5 s	2.1 s	1.4 s	380 ms	550 ms

```
.model motor macro lang=veriloga
* Instantiate Verilog-AMS motor
ymotor motor shaft p n
* Instantiate load resistor
rload n 0 1e09
* Initialize transient simulation
.tran 1u 1sec
* Define circuit alteration
.alter
ymotor motor_1 shaft p n
* List of all the faulty circuits
.end
```

Listing 1. Sketch of SPICE code to control the simulations.

the simulation is run only once for all comparisons, writing all results to the dedicated file.

Table IV shows the injection and simulation results derived from the DC motor model introduced and illustrated in this paper as a running example. Both the already presented versions of the motor are reported: the original electro-mechanical description and the analogous circuit. The table also shows the results of the same operations on other test cases, such as the RLC circuit, the Tuned-RLC (i.e., the electrical equivalent of the Mass-Spring-Damper) [19] and from the electrical equivalent of a double pendulum [24]. Specifically, the results refer to injection regarding the number of different faults (by position) injected into the circuit for each fault model. The results in the table also cover simulation: the time values reported are of both faulty and fault-free circuits. These times refer to a 30-seconds simulated behavior, and the circuit receives always the same input signal. Simulating faulty systems usually takes longer because faults add more overhead to system behavior.

VI. CONCLUSIONS

The article proposes techniques to support the *functional safety* in a CPS related to fault modeling and injection through physical analogies. Specifically, an automatic flow has been proposed which can inject faults into multi-domain models and create an efficient simulation environment. The mechanical part of these models is translated to its electrical equivalent through analogy in order to simulate the entire system as a full electrical network. Then, by simulating the full electrical network enriched with behavioral electrical faults, a mechanical fault taxonomy has been inferred by systematically studying the faulty behaviors.

The future development of this work is to make this methodology scalable: a system can be divided into parts to simplify the application of the proposed methodology. Each one of these subsystems can be manipulated individually and then reassembled during simulation based on SPICE simulators.

```
`include "disciplines.vams"
`include "constants.vams"
`timescale 1us / 1us
module motor(shaft, p, n);
  // PARAMETERS ------------------------
  // Motor constant (V-s/rad)
  parameter real km = 4.5;
  // Flux constant (N-m/A) = Kt
  parameter real kf = 4.5;
  // Inertia of the shaft (N-m-s2/rad)
  parameter real js = 1.2;
  // Drag rotating friction of the shaft (N-m-s/rad)
  parameter real ds = 0.1;
  // Motor winding resistance (Ohms)
  parameter real r = 5.0;
  // Motor winding inductance (H)
  parameter real l = 0.02;
  // Inertia of the wheel (N-m-s2/rad)
  parameter real jw = 0.005;
  // Drag rotating friction of the wheel (N-m-s/rad)
  parameter real dw = 0.1;
  // PORTS ----------------------------
  output shaft;
  input p, n;
  // NODES ----------------------------
  electrical p, n;
  // Internal nodes.
  electrical n1, n2;
  rotational_omega shaft, rgnd;
  // Reference nodes.
  ground rgnd;
  // BRANCHES -------------------------
  branch (p, n1)  Vm;
  branch (n1, n2) R1;
  branch (n2, n)  L1;
  branch (shaft, rgnd) bshaft;
  // BEHAVIOR -------------------------
  analog begin
    // Electrical model of the motor winding.
    V(Vm) <+ km * Omega(bshaft);
    V(R1) <+ r * I(R1);
    V(L1) <+ l * ddt(I(L1));
    // Physical model of the shaft.
    Tau(bshaft) <+ +kf * I(Vm);
    Tau(bshaft) <+ -(ds + dw) * Omega(bshaft);
    Tau(bshaft) <+ -(js + jw) * ddt(Omega(bshaft));
  end
endmodule
```

Listing 2. Original DC motor model, fault-free.

```
`include "disciplines.vams"
`include "constants.vams"
`timescale 1us / 1us
module motor_el(shaft, p, n);
  // PARAMETERS ------------------------
  // parameters equivalent to the first model
  // PORTS ---------------------------
  output shaft;
  input p, n;
  // NODES ---------------------------
  electrical p, n;
  electrical n1, n2, n3, n4, n5;
  rotational_omega shaft, wheel;
  ground n;
  // BRANCHES ------------------------
  branch (p, n1)  Vm;
  branch (n1, n2) Rm;
  branch (n2, n3) Lm;
  branch (n3, n4) Vback;
  branch (n4, n5) Lsw;
  branch (n5, n) Rsw;
  // BEHAVIOR ------------------------
  analog begin
    V(Vm) <+ km * I(Vm);
    V(Rm) <+ r * I(Rm);
    V(Lm) <+ l * ddt(I(Lm));
    V(Vback) <+ - kf * I(Vback);
    V(Lsw)   <+ (ls - lw) * ddt(I(Lsw));
    V(Rsw)   <+ (rs - rw) * I(Rsw);
  end
endmodule
```

Listing 3. Full electrical DC motor model, fault-free.

```
`include "disciplines.vams"
`include "constants.vams"
`timescale 1us / 1us

module motor_el(shaft, p, n);
  // PARAMETERS ------------------------
  // parameters equivalent to the first model
  // PORTS ---------------------------
  output shaft;
  input p, n;
  // NODES ---------------------------
  electrical p, n;
  electrical n1, n2, n3, n4, n5, x;
  rotational_omega shaft, wheel;
  ground n;
  // BRANCHES ------------------------
  branch (p, n1)  Vm;
  branch (n1, x)  Rm;
  branch (x, n2)  fault;
  branch (n2, n3) Lm;
  branch (n3, n4) Vback;
  branch (n4, n5) Lsw;
  branch (n5, n)  Rsw;
  // BEHAVIOR ------------------------
  analog begin
    V(Vm) <+ km * I(Vm);
    V(Rm) <+ r * I(Rm);
    V(Lm) <+ l * ddt(I(Lm));
    V(Vback) <+ - kf * I(Vback);
    V(Lsw)   <+ (ls - lw) * ddt(I(Lsw));
    V(Rsw)   <+ (rs - rw) * I(Rsw);
    V(fault) <+ I(fault) * 1e09;   // open fault
  end
endmodule
```

Listing 4. Full electrical DC motor model, open failed.

References

[1] *ISO 26262 – Road vehicles – Functional safety*, ISO, 2011.

[2] A. Nardi, S. Camdzic, A. Armato, and F. Lertora, "Design-for-safety for automotive IC design: Challenges and opportunities," in *Proc. of IEEE Custom Integrated Circuit Conference (CICC)*. IEEE, Apr. 2019.

[3] P. Gardonio and M. J. Brennan, "On the origins and development of mobility and impedance methods in structural dynamics," *Journal of Sound and vibration*, vol. 249, no. 3, pp. 557–573, 2002.

[4] F. Pecheux, C. Lallement, and A. Vachoux, "Vhdl-ams and verilog-ams as alternative hardware description languages for efficient modeling of multidiscipline systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 204–225, 2005.

[5] IEEE, *IEEE-SA Standards Board P2427/D0.13 Draft Standard for Analog Defect Modeling and Coverage*. IEEE, 2019. [Online]. Available: https://standards.ieee.org/project/2427.html

[6] E. Balaban, P. Bansal, P. Stoelting, A. Saxena, K. F. Goebel, and S. Curran, "A diagnostic approach for electro-mechanical actuators in aerospace systems," in *2009 IEEE Aerospace conference*. IEEE, 2009, pp. 1–13.

[7] S. J. Uder, R. B. Stone, and I. Y. Tumer, "Failure analysis in subsystem design for space missions," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 46962, 2004, pp. 201–217.

[8] C. Chen *et al.*, "Research on mechanical fault injection method based on ADAMS," in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, Oct. 2019.

[9] A. Kolesnikov, D. Tretsiak, and M. Cameron, "Systematic simulation of fault behavior by analysis of vehicle dynamics," in *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. Linköing University Electronic Press, Feb. 2019.

[10] J. Gundermann, A. Kolesnikov, M. Cameron, and T. Blochwitz, "The fault library - a new modelica library allows for the systematic simulation of non-nominal system behavior," in *Proc. of the 2nd Japanese Modelica Conference Tokyo*, 2019.

[11] P. Kroes, "Structural analogies between physical systems," *The British Journal for the Philosophy of Science*, vol. 40, no. 2, pp. 145–154, 1989.

[12] R. R. Thakkar, "Electrical equivalent circuit models of lithium-ion battery," *Management and Applications of Energy Storage Devices*, 2021.

[13] F. Grossi, A. Palladino, R. Zanasi, and G. Fiengo, "The pog technique for modelling engine dynamics based on electrical analogy," in *2009 American Control Conference*, 2009, pp. 2057–2063.

[14] C.-A. Chen, X. Li, L. Zuo, and K. D. Ngo, "Circuit modeling of the mechanical-motion rectifier for electrical simulation of ocean wave power takeoff," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3262–3272, 2020.

[15] Y. Save, N. Harihar, and S. Patkar, "Solution of partial differential equations by electrical analogy," *J. Comput. Science*, vol. 2, pp. 18–30, 03 2011.

[16] J. Bougie and A. Gangopadhyaya, "Conservation laws and energy transformations in a class of common physics problems," *American Journal of Physics*, vol. 87, no. 11, pp. 868–874, nov 2019.

[17] N. Hogan and P. C. Breedveld, "The physical basis of analogies in physical system models," in *The mechatronics handbook*. CRC Press/Balkema, 2002.

[18] P. Gardonio and M. J. Brennan, "Mobility and impedance methods in structural dynamics: an historical review," *ISVR Technical Report 289*, 2000.

[19] J. López-Martínez, J. C. Martínez, D. García-Vallejo, A. Alcayde, and F. G. Montoya, "A new electromechanical analogy approach based on electrostatic coupling for vertical dynamic analysis of planar vehicle models," *IEEE Access*, vol. 9, pp. 119 492–119 502, 2021.

[20] M. Smith, "Synthesis of mechanical networks: the inerter," *IEEE Transactions on Automatic Control*, vol. 47, no. 10, pp. 1648–1662, 2002.

[21] N. Dall'Ora, F. Tosoni, E. Fraccaroli, and F. Fummi, "Inferring mechanical fault models from the electrical domain," in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2022, pp. 01–08.

[22] N. Dall'Ora, E. Fraccaroli, S. Vinco, and F. Fummi, "Multi-discipline fault modeling with verilog-ams," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2021, pp. 237–243.

[23] N. Bombieri, G. D. Guglielmo, L. D. Guglielmo, M. Ferrari, F. Fummi, G. Pravadelli, F. Stefanni, and A. Venturelli, "HIFSuite: Tools for HDL code conversion and manipulation," in *2010 IEEE International High Level Design Validation and Test Workshop (HLDVT)*. IEEE, Jun. 2010.

[24] T. Yaren, S. Kizir, and A. B. Yildiz, "Electrical equivalent circuit based analysis of double pendulum system," in *2019 6th International Conference on Electrical and Electronics Engineering (ICEEE)*, 2019, pp. 258–262.