Cyber-physical systems design flow to manage multi-channel acquisition system for real-time migraine monitoring and prediction

Kevin Henares, José L. Risco-Martín, Josué Pagán, Carlos González, José L. Ayala and Román Hermida

Abstract Chronic diseases represent the major health problem of the 21st century. These diseases kill 41 million people each year, equivalent to 71% of all deaths globally. The major chronic diseases listed by the World Health Organization are cardiovascular diseases, cancer, chronic respiratory diseases, diabetes mellitus, and neurodegenerative disorders. Monitoring and maintaining normal values for key health metrics play a primary role in reducing chronic disease risk. Powerful mechanisms based on prevention to combat the chronic disease crises are currently present and continue to evolve. A new healthcare delivery model is needed to implement these mechanisms effectively. This model implies the utilization of wearable devices connected to the Cloud, allowing continuous monitoring and prevention of chronic disease crises. Standard Modeling and Simulation (M&S) methodologies created to design Cyber-Physical Systems (CPS) and deploy them into the Cloud can help design and implement complex scenarios. In this chapter, we show the automatic CPS implementation process of a robust migraine prediction system that allows the generation of alarms before the appearance of new pain episodes. This method is used to implement the device in an FPGA and to study the scalability of the proposed infrastructure, the integration of the designed device into an Internet of Things (IoT) ecosystem is demonstrated.

K. Henares, J. L. Risco-Martín, C. González, J. L. Ayala and R. Hermida
Department of Computer Architecture and Automation, Complutense University of Madrid, C/Prof. José García Santesmases 9, 28040 Madrid, Spain, e-mail: khenares,jlrisco,carlosgo,jayala,rhermida@ucm.es

J. Pagán
Department of Electronic Engineering, Technical University of Madrid, Avda. Complutense 30, 28040, Madrid, Spain, e-mail: j.pagan@upm.es

1 Introduction

Migraine causes recurrent headaches and it is one of the most disabling neurological diseases. It affects around 10% of population worldwide [9] and 15% in Europe [21]. Besides, in Europe it has a cost of 1,222 € per patient per year [9], taking into account direct and indirect costs. They include both medical costs and the ones caused by the decline in the productivity at work or school.

Migraines are characterized by recurrent headaches of medium and high intensity. Nevertheless, the pain is not the only symptom of a migraine. Some of them are prodromes, auras and postdromes. Premonitory or prodromic symptoms may occur from three days to hours before the pain starts [3]. They are subjective, varied and include changes in mood, appetite, sleep, etc. Auras occur in one-third of the cases [11] and appear within 30 minutes before the onset of pain. It consists in a short period of visual disturbance. Postdromes are symptoms that occur after the headache. Some of the most common are tiredness, head pain or cognitive difficulties. They are present in the 68% of the patients and they have an average duration of 25.2 hours [8].

The medicines used to neutralize the migraine-related pain do not have an immediate response but rather its constituents are slowly absorbed by the organism. Also, most migraine patients tend to wait until the onset of pain to take their medications (since the prodromic symptoms do not allow them to know for sure when will the pain start and the auras have too short durations). This delayed intake reduces the effectiveness of the treatment. In this way, the prediction of the beginning of the pain episode can help the patients to reduce or even cancel their pain episodes.

Migraine prediction has been modeled in our previous works [13] [14]. These predictive models based their pain levels estimations in a set of hemodynamic variables. These variables are controlled by the Autonomous Nervous System (ANS) and are altered when a migraine occurs. Specifically, four variables were analyzed: skin temperature (TEMP), electrodermal activity (EDA), oxygen saturation (SpO2) and heart rate (HR). In [12], the Discrete Event System Specification (DEVS) formalism was used to describe the prediction and define fault tolerance mechanisms for the system. We also proved in [5] that the DEVS model could be easily translated into a hardware specification language like VHSIC [1] Hardware Description Language (VHDL).

This chapter describes the prediction system developed so far in the literature and shows the details about the final implementation of a predictive device in a Field-Programmable Gate Array (FPGA). The FPGA technology facilitates the translation of the conceptual model into a hardware platform. An FPGA is an integrated circuit that can be programmed or reprogrammed to the required functionality or application after manufacturing. Physical sen-

[1] Very High Speed Integrated Circuit

sors and an additional control board complement the FPGA design. Finally, a model of this device is incorporated into a novel Modeling and Simulation (M&S) framework, designed to analyze the deployment of the final device within an IoT environment, where a huge set of monitoring devices are continuously used for predicting migraine crises. When one of the devices accumulates an error that is not acceptable, the predictive model is re-trained at the Cloud that stores the patient's entire dataset. After that, the updated model is sent back to the physical device. Although the prediction methodology used in this chapter is applied to the migraine disease, it can be applied to other diseases with symptomatic crisis [15], developing equivalent prediction systems. Some examples would be strokes, epileptic seizures or Parkinson. Since the whole design of the cyber-physical system is conducted through the DEVS M&S formalism [24], this chapter defines a reproducible design flow for the design and deployment of these monitoring devices.

This chapter is organized as follows: the set of technologies involved in the design flow is presented in Section 2. The hardware implementation of the migraine prediction system is shown in Section 3. The framework implemented to incrementally deploy the resultant IoT system is described in Section 4. Finally, the chapter is concluded in Section 5.

2 Technologies involved in the design flow

In this Section, we explain the main concepts behind the methods and hardware components used to implement the monitoring device, as well as those methods used to simulate and analyze the impact on scalability. Firstly, we provide an introduction of FPGAs, why we have selected this development platform and its utility as a prototyping architecture in the healthcare domain. Next, we explain the DEVS M&S formalism, used to conduct the whole design of the hardware device on the one hand and the software platform to analyze scalability on the other. Finally, we also briefly introduce some basic concepts used to build and train our predictive models that will be later recorded in the physical device.

2.1 FPGAs and Healthcare Monitoring Systems

There is a wide range of applications for Healthcare Monitoring System (HMS) in supporting medical and healthcare services. By attaching portable devices to patients, vital healthcare data can be automatically collected, which is then forwarded to a nurse center for patient state monitoring. The benefit of this scenario is that it can reduce the working load of physicians and result in increased efficiency in patient management. For the patient, the

most important qualities of HMS must be small size, easy-to-use, lightweight and portable.

For the development of a successful HMS, the cost-effectiveness of the solution is one of the main driving factors. The development of a system that only has the necessary functions will help to reduce expenses on the design. Another aspect that can reduce costs is using easily accessible, widely used and fully configurable components. Using programmable components removes the likelihood that an inconvenient, non-cost-effective device be chosen for the system. The best solution is to build an HMS based on FPGA. It is an integrated circuit that is programmable by developers or customers after production. This is why it is called field-programmable. FPGA is designed to be programmable by changing the functional logic of the principal circuit using a hardware description language. An HMS using this technology would contain a low-cost, Analog-to-Digital Converter (ADC), which is used to transform an analog signal into a digital one. Digitization allows users to connect the FPGA to the entire system. The main advantage of the FPGA is the ability to reconfigure it after it has been manufactured. This helps fix bugs easier and more quickly. Moreover, the array takes less time to go from the drawing board to the market. FPGA also has lower non-recurring engineering costs. This means that manufactures only pay for research, design, building, and testing once.

Another major challenge in these applications is power consumption. All these systems require a high level of integration due to the necessary portability. High power efficiency is needed for systems that are truly portable and thus battery operated. Here, lower power consumption will increase the operation time of the device without recharging or replacing batteries. FPGAs implement optimized data paths for each computational task. Using optimized data paths not only improves the performance but also reduces the power and energy consumption when compared with General Purpose Processors (GPPs). Executing a given task in GPPs involves adjusting the required computations to the GPP instruction set and carrying out an expensive instruction decoding process, which leads to important power and energy overheads. In contrast, the power and energy efficiency of FPGAs has significantly improved during the last decade. FPGA vendors have achieved this goal by improving the FPGA architectures, including optimized hardware modules, and taking advantage of the most recent silicon technology. For instance, Xilinx reports a 50% reduction in the power consumption when moving from their previous Xilinx 6 FPGAs (implemented using 40-nm technology) to their most recent Xilinx 7 FPGAs (a new architecture implemented using 28-nm technology).

For these reasons, and those discussed below, the trend is to implement these HMS in FPGAs instead of microprocessors as traditionally done [1] [17] [2]. The first systems were simpler and the number of input modules provided by the microcontrollers was sufficient [23]. Unfortunately, simply switching to a bigger microcontroller only helps to a certain extent as

the focus is usually on more memory and general-purpose pins rather than I/O modules. Adding more microcontrollers complicates the overall system design due to the communication overhead. FPGAs, in contrast, do not have these limitations. Further, the microprocessors are capable of executing only one instruction per cycle while in FPGAs, due to a large number of available resources, it is also possible to execute several tasks in parallel to achieve further speedups.

2.2 Discrete Event System Specification (DEVS)

DEVS is a general formalism for discrete event system modeling based on Set theory [24]. The DEVS formalism provides the framework for information modeling which gives several advantages to analyze and design complex systems: completeness, verifiability, extensibility, and maintainability. Once a system is described in terms of the DEVS theory, it can be easily implemented using an existing computational library.

The parallel DEVS formulation (PDEVS), which is the one used in our approach, enables the representation of a system by three sets and five functions: input set (X), output set (Y), state set (S), external transition function (δ_{ext}), internal transition function (δ_{int}), confluent function (δ_{con}), output function (λ), and time advance function (ta).

DEVS models are of two types: atomic and coupled. Atomic models are directly expressed in the DEVS formalism specified above. Atomic DEVS processes input events based on their model's current state and condition, generates output events and transition to the next state. The coupled model is the aggregation/composition of two or more atomic and coupled models connected by explicit couplings. Particularly, an atomic model is defined by the following equation:

$$A = \langle X, Y, S, \delta_{\text{ext}}, \delta_{\text{int}}, \delta_{\text{con}}, \lambda, ta \rangle \tag{1}$$

where:

- X is the set of inputs described in terms of pairs port-value: $\{p \in IPorts, v \in X_p\}$.
- Y is the set of outputs, also described in terms of pairs port-value: $\{p \in OPorts, v \in Y_p\}$.
- S is the set of states.
- $\delta_{\text{ext}} : Q \times X^b \rightarrow S$ is the external transition function. It is automatically executed when an external event arrives to one of the input ports, changing the current state, if needed.
  - $Q = (s, e)s \in S, 0 \leq e \leq ta(s)$ is the total state set, where e is the time elapsed since the last transition.
  - X^b is the set of bags over elements in X.

- $\delta_{\text{int}} : S \rightarrow S$ is the internal transition function. It is executed right after the output (λ) function and is used to change the state S.
- $\delta_{\text{con}} : Q \times X^b \rightarrow S$ is the confluent function, subject to $\delta_{\text{con}}(s, ta(s), \emptyset) = \delta_{\text{int}}(s)$. This transition decides the next state in cases of collision between external and internal events, i.e., an external event is received and elapsed time equals time-advance. Typically, $\delta_{\text{con}}(s, ta(s), x) = \delta_{\text{ext}}(\delta_{\text{int}}(s), 0, x)$.
- $\lambda : S \rightarrow Y^b$ is the output function. Y^b is the set of bags over elements in Y. When the time elapsed since the last output function is equal to $ta(s)$, then λ is automatically executed.
- $ta(s) : S \rightarrow \Re_0^+ \cup \infty$ is the time advance function.

The formal definition of a coupled model is described as:

$$M = \langle X, Y, C, EIC, EOC, IC \rangle \tag{2}$$

where:

- X is the set of inputs described in terms of pairs port-value: $\{p \in IPorts, v \in X_p\}$.
- Y is the set of outputs, also described in terms of pairs port-value: $\{p \in OPorts, v \in Y_p\}$.
- C is a set of DEVS component models (atomic or coupled). Note that C makes this definition recursive.
- EIC is the external input coupling relation, from external inputs of M to component inputs of C.
- EOC is the external output coupling relation, from component outputs of C to external outputs of M.
- IC is the internal coupling relation, from component outputs of $c_i \in C$ to component outputs of $c_j \in C$, provided that $i \neq j$.

Given the recursive definition of M, a coupled model can itself be a part of a component in a larger coupled model system giving rise to a hierarchical DEVS model construction.

2.3 Predictive models

2.3.1 Migraine predictive modeling

The migraine predictive system presented by the authors in [16] and [12] included the possibility of using different types of predictive mathematical models such as Grammatical Evolutionary algorithms and state-space models respectively. In this work we have considered the later, a Subspace State Space System Identification (N4SID) model to generate the prediction of the characteristics of the acute migraine attack, including time of onset, time to peak, duration, intensity, quality, etc.

N4SID is a state-space based algorithm [22]. It describes immeasurable states and specifies differential equations that relate future outputs with current and past inputs. It is formally described in 3 and 4.

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3}$$

$$y_k = Cx_k + Du_k + v_k \tag{4}$$

where:

- u_k are our $U = 4$ hemodynamic inputs—body temperature, sweating, heart rate, oxygen saturation—at time k.
- y_k is the output at time k. In this project it will be the predicted pain level.
- A, B, C, and D are the state-space matrices.
- v_k and w_k represent white immeasurable noises. More details can be found in [13].

2.3.2 History-based signal repair

If a sensor breaks, prediction worsens. Being able to temporarily repair a signal keeps the predictive system active until the error is fixed. A DEVS-based *Sensor Status Detector* with a subsystem for signal repairing based on Gaussian Process Machine Learning (GPML) has already been reported [18] [13]. Due to the complexity of a procedure like GPML, an FPGA would consume numerous resources to replicate it. In this chapter, we have explored simpler ways to optimize this task. As a result, an Auto-Regressive model with eXogenous inputs (ARX) model is used to repair the signal (in case of sensor failure).

Auto Regressive (AR) models assume that current values of a variable depend on a polynomial combination of its own past values. In addition, ARX models consider the influence of past data of exogenous (external) variables as well. As our hemodynamic variables relate each other through the ANS, we contemplate these polynomial models as eq. 5 shows:

$$y_k + a_1 \cdot y_{k-1} + \ldots + a_{n_a} \cdot y_{k-n_a} = b_1 \cdot u_{k-n} + ... + b_{n_b} \cdot u_{t-n_b-n+1} + e_k \tag{5}$$

where:

- y_k is the output—one hemodynamic variable—at time k.
- n_a and n_b are the number of poles and, zeros plus one, of the polynomial.

- n is the number of input samples that occur before the input affects the output.
- $y_{t-1}, \ldots, y_{t-n_a}$ are the previous outputs on which the current output depends.
- $u_{k-n}, \ldots, u_{k-n-n_b+1}$ are the previous exogenous inputs on which the current output depends—the remaining non-damaged hemodynamic variables.
- e_k is a white-noise disturbance value.

3 System I: Migraine predictive device

In this section, we show the details about the implementation of the VHDL system in the FPGA and the sensor management board used to group the input. Moreover, it shows the options proposed for decimal numbers representation and a study of the precision loss resulting from the use of fixed-point decimal numbers.

3.1 FPGA implementation

The earlier developed DEVS-model [12] validated the design and behavior of the system. As both share a hierarchical nature, the transformation from DEVS to VHDL was a straightforward process.

After validating the system in the DEVS environment, an HW implementation using VHDL was implemented in an FPGA. A general view of this system can be seen in Figure 1 (coupled models are TEMP SSD, EDA SSD, HR SSD, SpO2 SSD and Predictor). It has the following main components:

- `Drivers`: Two of these modules were specified. One of them collects oxygen saturation and heart rate data. It receives that data from a NONIN OEM III module [2] using serial communication. The other one receives the data relative to electrodermal activity and body temperature measures. As the sensors used for this purpose generate an analog output, an Analog to Digital Converter (ADC) module has been used. Specifically, a Pmod AD2 has been selected [3]. It counts with 4 channels, 12 bits of precision and communication through the Inter-Integrated Circuit (I2C) protocol.
- Synchronizer (`Sync`): Packs the input values of the different sensors into a unified data structure. As the sensors can have different sampling frequency, the synchronizer has to decide how to process the situations where there is sparse data. In this way, the values received by the synchronizer

[2] http://www.nonin.com/OEM-III-Module

[3] https://reference.digilentinc.com/reference/pmod/pmodad2/start

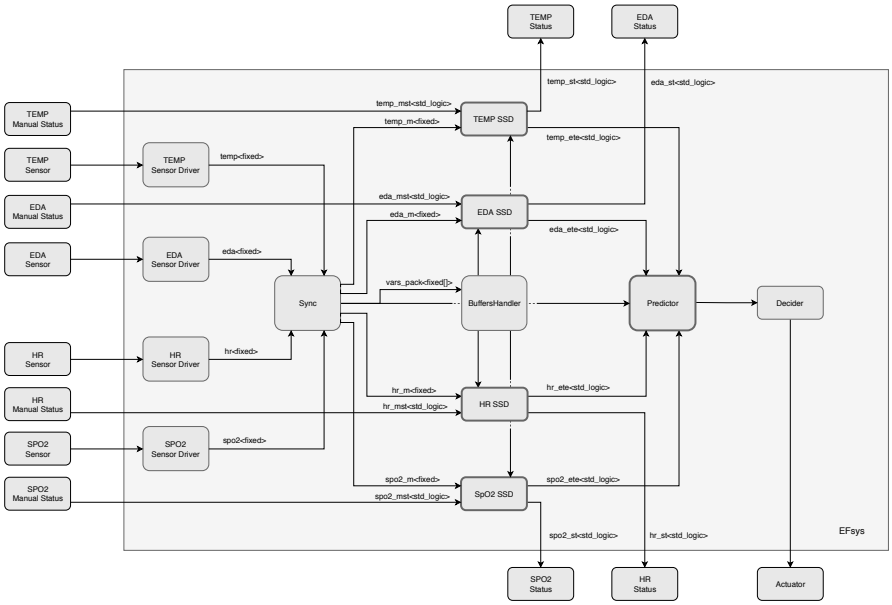


Fig. 1 Root component of the migraine prediction system implemented in an FPGA with VHDL.

are grouped into accumulators. Once per minute, a pulse is raised and a new data packet is created. It contains the average of all the values received in each one of the variables. When that happens, the accumulators are reset to store the data of the next packet.

- `Sensor Status Detectors` (`SSDs`, Figure 2): They check if different error types are present in the input signal (saturation, fall or noise). Where one of them is detected, the status signal related to the controlled variable is raised, so that the patient can restore the sensor. In the meantime, a module to regenerate the signal is activated (`ARX`). It generates input estimations using previous samples of both the controlled variable and the exogenous ones. When too much time since the error detection passes, an Elapsed Time Exceeded (ETE) signal is raised. That signal points out that the variable implied can not be used reliably to generate predictions, so the Predictor module will discard it.
- `BuffersHandler`. This was added to the implementation because all the `SSDs` use the information of all the biomedical variables of the system (in their `ARX` modules). Therefore, it is convenient to centralize the management of these data.
- `Predictor` (Figure 3): Generates a numeric indicator that predicts the probability of occurrence of a new pain episode. For that, it contains five sets of of three N4SID models each, 12 models in total. Each set is trained using a subset of the four biomedical variables. There is one set of mod-

els for the situation in which all the variables are operational and four more sets for the cases where one of the variables fails. When there are failures in two or more variables the prediction is not considered representative and is not supported by the system. The three models generated for each combination of variables are trained taken into account different prediction horizons. There are models specialized in generating short-term, medium-term and long-term predictions. By combining these predictions more accurate predictions are generated. In this way, when the group of variables is used to predict change, the corresponding set of models is loaded. This happens when the number of damaged variables changes (an additional error is found or one of the damaged variables is regenerated).

- `Decider`: Activates the alarm when the output generated by the `Predictor` module exceeds a certain threshold (previously trained with several hours of data).

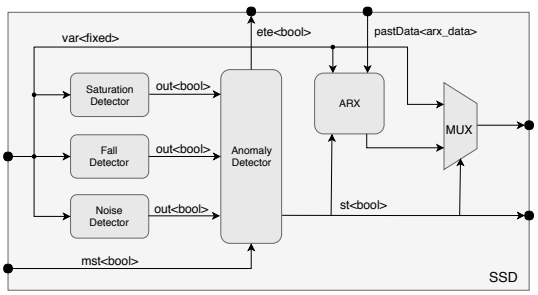


Fig. 2 `SSD` component. It detects errors in the signal.

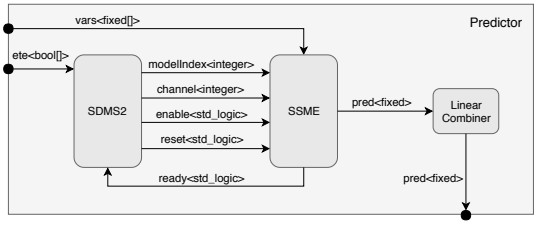


Fig. 3 `Predictor` component. It generates predictions about the probability of a migraine occurring.

Among these components, the `SSD` and the `Predictor` deserve a detailed explanation. Both contain several sub-components, as shown in Figures 2 and 3.

The SSD, as discussed previously, aims to detect the possible errors present in the signal and to temporarily fix them. For that, it is composed of the following components:

- Noise Detector: Detect noise present in the input signal following the mean squared error.
- Fall Detector: Establish the existence of a fall when the average of the values of a sensor is lower than a certain threshold.
- Saturation Detector: Establish the existence of a saturation when the average of the values of a sensor exceeds a threshold.
- Anomaly Detector: Raises a status signal when one of the three errors previously explained is detected. In this way, the patient can check it and replace it, if necessary. From that moment the GPML module will be activated to estimate the signal temporally. Moreover, when this situation continues for a certain time, the Estimation Time Exceeded (ETE) signal is raised, reporting that the input signal can no longer be estimated.
- ARX: Estimates the signal based on previous samples. This estimation is generated using an Auto-Regressive model with eXogenous inputs (ARX) model.

The Predictor estimates the probability of a migraine attack and has the following components:

- Sensor Dependant Model Selection System (SDSM2): The SDMS2 module is responsible for selecting the correct models and requesting the generation of the three predictions, which are carried out sequentially. This has been done to reduce the resources consumed by the system.
- Sensor Dependant Model Selection System (SSME): This generate the predictions based on state-space models. Since this process is sequential, it has three channels to select which one of the predictions is being calculated. In this way, each one of the channels is associated with one state. When the set of variables used to predict changes, the SDMS2 detects it and informs the SSME through a reset pulse. Then, the SSME module loads the three models related to the new set of operational variables and resets the states that were being used to generate predictions in the previous conditions.
- Linear Combiner: performs the average of each one of the groups of three predictions generated by the SSME module.

System synchronization is controlled by several pulses and clock signals. The used FPGA has a base frequency of 125MHz. It is used to manage the communication protocols and to generate the operational clock: a clock is used to control the operation of the components of the system. Since the system does not have high time requirements it has been set to 100KHz. This allows reducing system resource consumption. Using the operation clock, several pulses are generated to inform the modules of the existence of newly available data. Extra details about how synchronization works are presented in [5].

3.2 HW set-up

In this work, both the N4SID and the ARX algorithms have been computed using the System Identification Toolbox of the MATLAB software[4].

For the implementation of the DEVS simulator the framework xDEVS has been used [20].

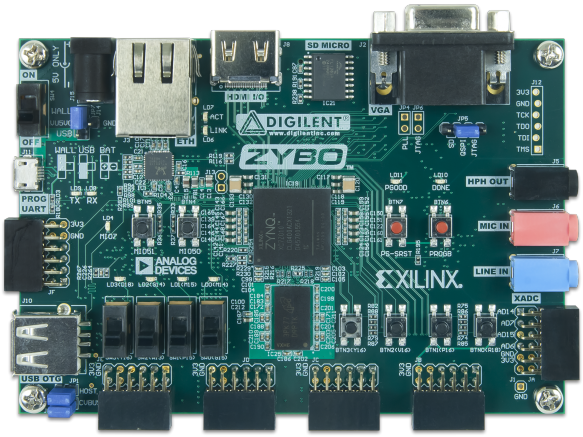


Fig. 4 Figure 7: Zybo Zynq 7000: Board used to test the migraine prediction system.

To deal with decimal numbers in VHDL the FLOAT32 data type was first used. It is included in the FLOAT_PKG package of the IEEE_PROPOSAL library. However, when the system is synthesized it needs many resources to handle the operations with this data type (especially, if the used FPGA does not have dedicated circuits to deal with floating-point numbers). For this reason, a fixed 20-bit precision data type was used instead.

To develop the VHDL implementation of the migraine prediction system the design software Xilinx ISE 14.7 has been used. Firstly, it was loaded in a Xula2-LX9, but due to memory limitations was ported to a Zybo Zynq 7000 development board (Figure 4). Therefore, a Zynq 7000 has been established as target FPGA (XC7Z010 device, CLG400 package). This FPGA has 240 KB of Block RAM, 28000 logic cells and 80 DSP slices.

Finally, for the integration of the sensors with the FPGA an additional board was designed as an interface. It has a DB9 connector (for the oxygen saturation input), a 3.5mm Jack connector (for the temperature sensor) and two Snap connectors (for the entry of electrodermal activity data). To obtain electrodermal activity values, the conductance between the two electrodes

[4] MATLAB 2015. version 8.5.0.197613 (R2015a). Natick, Massachusetts, The MathWorks Inc.

must be measured. To do this, a Wheatstone bridge is used. It allows us to read a potential difference dependent on the impedance variation of a sensor.

3.3 Validation

To check the correct functioning of the system two testing phases were implemented. On the one hand, a software emulation was carried out using the ISE simulator (ISim), to check the operation of all system modules. On the other hand, after the synthesis of the system, data was generated directly in a Zybo Zynq 7000 FPGA and compared with those obtained in the DEVS simulation. To perform these tests real data was used. The data was obtained monitoring the activity of migraine patients in ambulatory conditions with a wireless body sensor network (WBSN) [13].

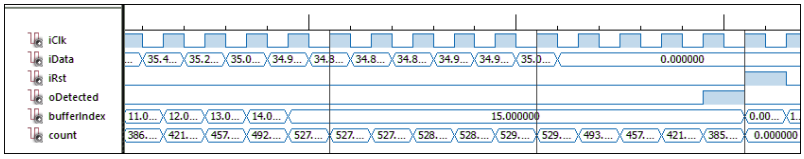


Fig. 5 Fall Detector emulation. In it we can see how a fall is detected and how it restarts the operation when the reset button is pressed.

Figure 5 corresponds to a `FallDetector` module. It can be seen how the error detection works. The data present in the input on each rising edge are stored in a buffer. When the buffer is full, the error detection goes into operation, calculating the mean of the last values and checking if it is lower than a threshold. When an error is detected the `oDetected` signal is raised. It will remain in this state until either data in the buffer is considered valid or the associated reset button is pressed. When that button is pressed the operation is restarted and the buffer is emptied.

`SaturationDetector` and `NoiseDetector` modules work similarly, raising their output in the situations explained above.

Those three detectors are connected with the `AnomalyDetector` module. That module manages two output signals. `oSensorStatus` is activated when at least one of the detectors reports an error. `oETE` is activated when `oSensorStatus` is in high state during a certain time. Figure 6 shows the aforementioned situation.

Finally, Figure 7 shows the emulation of the root component of the prediction system. It can be seen the four input variables corresponding to a real episode, that was previously stored, and the prediction generated by the system. Processing that information the system generates predictions and raises

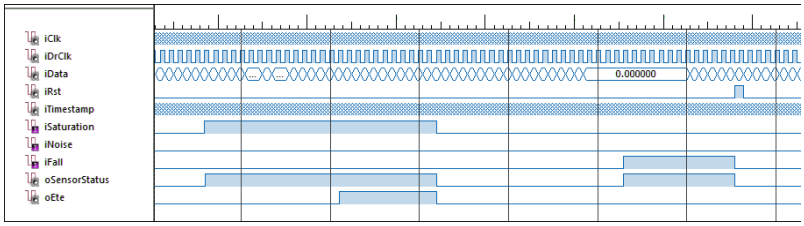


Fig. 6 Anomaly Detector emulation. It groups the errors of the three detector modules.

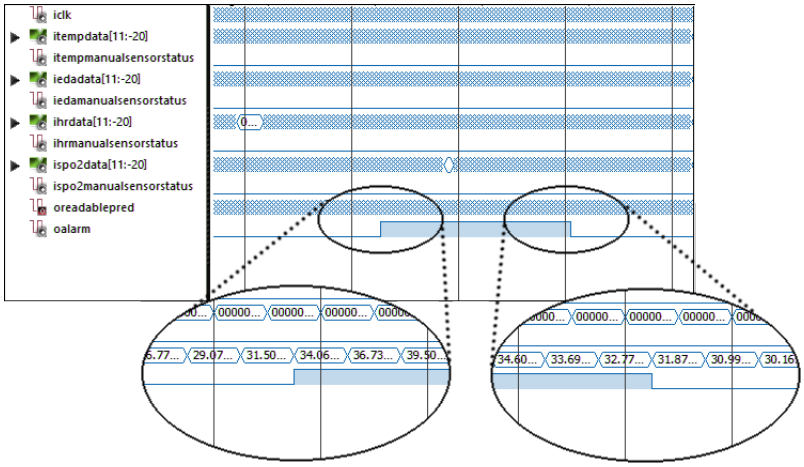


Fig. 7 Root emulation. We can see how the alarm raises when the prediction exceeds a threshold.

the `oAlarm` signal when it exceeds a threshold (32 in the shown figure). This signal will return to low level when the predictions are back below the limit.

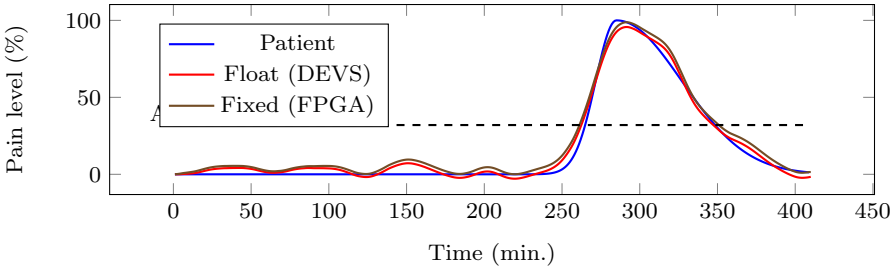


Fig. 8 Comparisons between the patient curve (generated with its own subjective pain level information) and the predictions generated by both the DEVS simulator (RMSE: 3,6%) and the FPGA implementation (RMSE: 4,94%). The alarm raises when the prediction raises the 32 threshold.

Once all the components of the system were emulated, patients' data were injected directly on the VHDL system (implemented in an FPGA). Figure 8 depicts the comparison between the pain levels relative to a real migraine episode. The blue curve corresponds to the one generated with the actual patient, with its subjective data. The red curve represents the predictions generated by the DEVS simulator using the IEEE754 standard. Finally, the brown one represents the output of the `Predictor` module in the FPGA (using fixed-point data types).

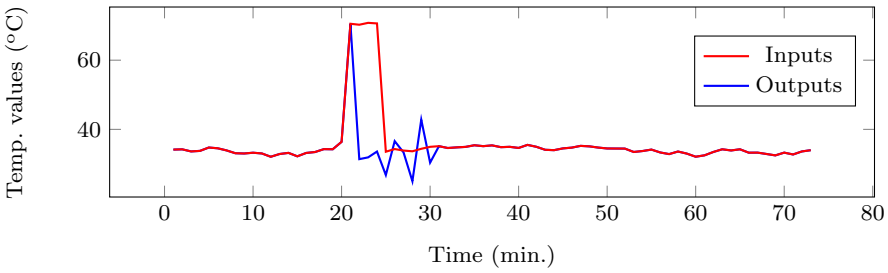


Fig. 9 `ARX` response after the appearance of a saturation error in a temperature input.

In Figure 9 it can be seen the appearance of saturation in the input of a `SSD` that manages temperature values (red line). The saturation is detected by the `SaturationDetector` module and the `AnomalyDetector` module activates the `ARX` to start recovering the signal. From this moment, the `ARX` module generates temporal estimations adapted to the previous trend of the input (blue line). To this end, it uses the previously registered values of that variable.

Component	SSD module	Predictor module	Complete system
# DSP48E1s	28	24	80
# slice registers	1530	1743	9618
# slice LUTs	1549	2705	13056
# LUT flip flop pairs	2648	4172	20059
Percentage of total	21	24	71
Maximum frequency (MHz)	41	64	40

Table 1 Summary of Resource Utilization for the FPGA-Based Implementation of the migraine prediction system on a Zynq XC7Z010-CLG400 FPGA.

Table 1 shows the percentage of hardware utilization for the main modules and the complete system. The percentage of total hardware utilization is 71%, so we have still leave room in the FPGA for additional algorithms.

4 System II: DEVS-based framework to deploy cyber-physical systems over IoT environments

4.1 Framework design

Once the nodes have been developed, it is time to think about how to manage the generation of models and where to perform the simulations and analyses i.e. the M&S deployment infrastructure. In some situations, models are generated statically with datasets and configurations good enough to cover the needs of the system to develop. As a consequence, they are trained offline once and do not have to be retrained. However, it is also common to evolve the models periodically with new data so that they can learn progressively how to generate the best outcomes. This happens in a large variety of scenarios. One example is when medical events or features of some disease are determined based on the data generated through the use of monitoring devices, especially in the ones that present different symptoms by each patient. This is the situation in the migraine scenario presented before. At first, grouped data of other patients with the same disease can be used to generate a general model. After that, to improve the outcomes in individual patients, their specific events are taken into account to improve the models gradually.

All these operations can be performed in different layers, depending on the power consumption constraints and the selected maximum latency. These layers are usually the edge, the fog, and the cloud. Edge devices usually have limited processing capabilities and battery constraints. On the other hand, the use of data centers implies the acceptance of higher delays. The use of the fog layer to perform the training, as long as they do not require large computing loads, usually offer a good trade-off between these two layers.

Before training and evaluating, data needs to be filtered, formatted and information recovery operations must be performed. The use of Model-Based Systems Engineering (MBSE) allows better management of all these operations. In this way, they can be easily reused and combined and brings greater scalability to the system. Also, as these operations usually imply similar tasks, they can be encapsulated so that only a parameterization is needed. In this way, a model training workflow could be specified using by describing and parameterizing individual predefined modules and the connections among them. After that, these modules can be instantiated and tuned to deploy a functional system based on this description. This methodology reduces development times and facilitates the generation of predictive models. Also, a formalism like DEVS can be used as a background to validate the operation of the system while controlling its operation. In DEVS, modeling and simulation layers are independent. The modular and hierarchical structure makes it a great option to support this methodology.

The resulting parameterized modules can easily communicate with each other. Moreover, this is still possible even if the modules are configured and

instantiated in different devices across the edge, fog, and cloud layers. An easy way to achieve this is using a DEVS library supporting distribution. In this way, couplings between local and remote modules can be specified, sending the intermediate values properly codified over the network.

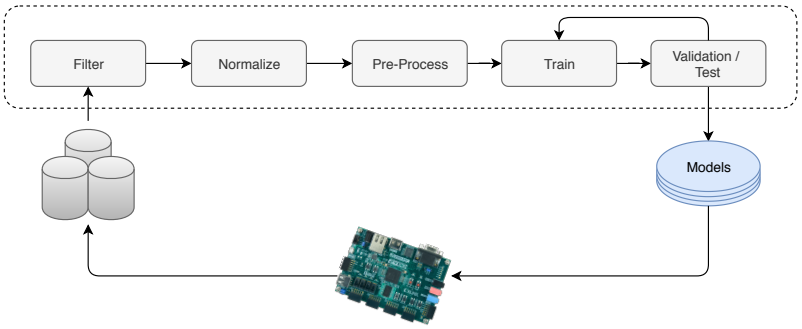


Fig. 10 Centralized representation of the framework. This framework is able to (re-)train predictive models based on data gathered from monitoring devices.

We have developed a DEVS-based M&S framework to perform the automatic creation of predictive models [7]. Figure 10 shows the common phases followed in this process. First, data from all the monitoring devices are collected and stored in the database (this process can be performed online or offline). In the `Filter` module, these data are filtered following several criteria, to assure quality discarding wrong or incomplete records. Next, normalization operations are applied in the `Normalize` module, unifying units of the variables, or performing categorization tasks and adjustments in the implied values or distributions. In the `Pre-Process` module, some transformations are applied to adapt the data format to the use case. For example, single samples coming from a patient monitoring device can be too granular to be directly related to the event to predict. In this situation, it is common to split clinical data into time-windows of a predefined size so that a certain number of consecutive samples matches the class to predict. Finally, the `Pre-Process` module separates the input data in several datasets: train datasets, that are used to fit the model, and validation/test datasets, that are used to adjust the models and select the best ones. Once the separation is done, different sets of models are generated with the training dataset using the suitable machine learning algorithms in the `Train` module. The set of algorithms to use depends as much on the problem characteristics as on the data types. Once trained, models are verified and compared using the previously separated validation/test datasets in the `Validation/Test` module. The selected models will be subsequently evaluated to generate valuable information about the area to be modeled. New samples can be used both to

evaluate the models and to feed the data sources. This allows us to generate new models periodically so that its accuracy can be increased over time.

This framework has been implemented using xDEVS [20], a DEVS M&S library. Particularly, the Python branch of xDEVS allows us to deploy a DEVS execution over the network. Hence, both FPGAs and their corresponding DEVS models can be integrated into the distributed co-simulation framework and analyze the impact of (i) scalability as the number of monitoring devices is increased and (ii) performance optimization, distributing the computation power of the framework between the fog and the cloud layers.

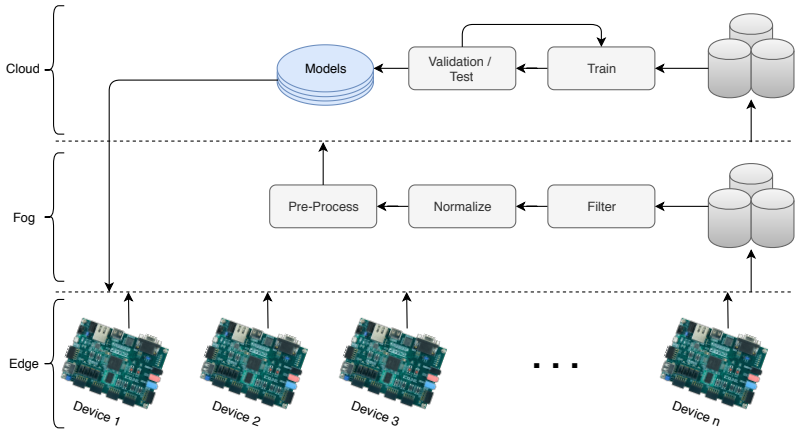


Fig. 11 Example of a combined deployment with VHDL and DEVS subsystems. The same two DEVS subsystems are instantiated in the fog and cloud layers.

We can use this framework with actual devices, simulated devices, or both. Co-Simulation of physical devices and software models can be easily performed [19]. Figure 11 shows an example of a deployment that uses the three layers aforementioned. In the edge layer, we may find the monitoring devices. Data collected are sent to a database located in the fog layer. This database represents the reference hospital of each patient. Data are filtered, normalized and pre-processed at the fog layer. Resulting sets are sent to the cloud, where more computational power is needed to train (or re-train) the existing models. This DEVS framework, currently a preliminary version developed in Python, has been tested both in a migraine and stroke predictive IoT system [6], [7].

4.2 From sensors to the cloud: scalability issues

IoT frameworks like the one presented in this chapter (Fig. 11) combines three computing layers, i.e., edge, fog and cloud computing. Large monitoring networks using this paradigm are a reality nowadays. In the sports area, for instance, the monitoring is being used to generate predictions in team sports [4]. In the medical area, applications like remote diagnosis, disease alarms generation, or prediction of atrial fibrillation are also following this trend [10].

This IoT architectures typically consist of three main parts: the sensing nodes, the coordinators and the data centers. Each one of the three network elements can operate in various modes. Sensing nodes can collect, transmit and process data. The controller can receive, transmit and process data, and perform predictions, if necessary. The data center can process data and perform predictions. The energy efficiency policies take these possibilities into account to minimize the power consumption of the whole system and to maximize the accuracy of the prediction.

Hence, sensor devices can collect and transmit data, coordinators can receive and transmit data and data centers can process data and perform predictions. Another possible scenario would be: sensors to collect and transmit data, coordinators to receive, process data, perform predictions and transit data, and data centers would not be necessary here, with a possible loss of accuracy. Further details about these features and the tasks to be performed are available in [6].

As can be seen, these IoT frameworks pose important challenges because of the large volumes of data that must be gathered and analyzed. Among all the possible IoT applications, population monitoring in e-health include important constraints. These applications also demand intelligent strategies to develop massive healthcare solutions in massive population scenarios, to minimize energy consumption while maximizing models' accuracy, ensuring reasonable levels of scalability.

5 Conclusions

This chapter describes a design flow to conceive, design and implement cyber-physical systems oriented to monitoring neurological diseases and predict possible attacks or crises.

The first step consists of defining a DEVS model of the whole device. The use of the DEVS formalism allows us to quickly debug our design and make decisions to optimize performance and accuracy. Once the model has been verified and validated, the second step consists of automatically translate the DEVS model into a VHDL description, which is a straightforward operation. This VHDL description can be synthesized into an FPGA to test the real-

world device in the field. We have discussed the multiple benefits of using FPGA as Health Monitoring Systems.

We have tested this design flow in a real neurological disorder, the migraine disease. Unfortunately, the floating-point representation has occupied too much space inside the FPGA, so we have implemented an ad-hoc fixed point model. We compared both the DEVS floating-point approach against the fixed-point approach, increasing the number of bits used in the decimal part until obtaining a low error. Future work includes the pre-computation of the design size so that this issue can be fixed from the DEVS model.

Finally, the DEVS device model has been integrated into a novel M&S framework. This framework has been implemented to analyze the scalability of the monitoring device when it is used in an IoT environment. It is used to simulate the coexistence of a huge set of health monitoring system, analyzing the effects of re-training the models saved inside the devices when the prediction error is increased.

Future work includes the automation of the whole design flow since currently many design steps must be performed manually. We are also considering the use in our design flow of other monitoring devices currently present in the market, like smart wristband or watches.

References

1. K. Arshak, E. Jafer, and C. S. Ibala. Fpga based system design suitable for wireless health monitoring employing intelligent rf module. In *2007 IEEE Sensors*, pages 276–279, Oct 2007.
2. C. C. Chou, W. C. Fang, and H. C. Huang. A novel wireless biomedical monitoring system with dedicated fpga-based ecg processor. In *2012 IEEE 16th International Symposium on Consumer Electronics*, pages 1–4, June 2012.
3. NJ Giffin, L Ruggiero, RB Lipton, SD Silberstein, JF Tvedskov, J Olesen, J Altman, Peter J Goadsby, and A Macrae. Premonitory symptoms in migraine an electronic diary study. *Neurology*, 60(6):935–940, 2003.
4. Benjamin H Groh, Samuel J Reinfelder, Markus N Streicher, Adib Taraben, and Bjoern M Eskofier. Movement prediction in rowing using a dynamic time warping based stroke detection. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6. IEEE, 2014.
5. Kevin Henares, Josué Pagán, José L. Ayala, and José L. Risco-Martín. Advanced migraine prediction hardware system. In *Proceedings of the 50th Computer Simulation Conference*, SummerSim '18, pages 7:1–7:12, San Diego, CA, USA, 2018. Society for Computer Simulation International.
6. Kevin Henares, Josué Pagán, José L. Ayala, Marina Zapater, , and José L. Risco-Martín. *Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy*, chapter Cyber-Physical Systems Design Methodology for the Prediction of Symptomatic Events in Chronic Diseases. 2019.
7. Kevin Henares, José L. Risco-Martín, Román Hermida, and Gemma Reig-Roselló. Modular framework to model critical events in stroke patients. In *Proceedings of the 2019 Summer Simulation Conference (SummerSim'19)*, 2019.

8. L Kelman. The postdrome of the acute migraine attack. *Cephalalgia*, 26(2):214–220, 2006.
9. M Linde, A Gustavsson, LJ Stovner, TJ Steiner, J Barré, Z Katsarava, JM Lainez, C Lampl, M Lantéri-Minet, D Rastenyte, et al. The cost of headache disorders in europe: the eurolight project. *European journal of neurology*, 19(5):703–711, 2012.
10. Jelena Milosevic, Andreas Dittrich, Alberto Ferrante, Miroslaw Malek, Camilo Rojas Quiros, Rubén Braojos, Giovanni Ansaloni, and David Atienza. Risk assessment of atrial fibrillation: a failure prediction approach. In *Computing in Cardiology Conference (CinC), 2014*, pages 801–804. IEEE, 2014.
11. Headache Classification Subcommittee of the International Headache Society et al. The international classification of headache disorders. *cephalalgia*, 24(1):1–160, 2004.
12. Josué Pagán, José M. Moya, José L. Risco-Martín, and José L. Ayala. Advanced migraine prediction simulation system. In *Summer Computer Simulation Conference (SCSC)*, 2017.
13. Josué Pagán, De Orbe, M Irene, Ana Gago, Mónica Sobrado, José L Risco-Martín, J Vivancos Mora, José M Moya, and José L Ayala. Robust and accurate modeling approaches for migraine per-patient prediction from ambulatory data. *Sensors*, 15(7):15419–15442, 2015.
14. Josué Pagán, José L. Risco-Martín, José M. Moya, and José L Ayala. Grammatical evolutionary techniques for prompt migraine prediction. In *Genetic and Evolutionary Computation Conference, 2016*. ACM, 2016.
15. Josué Pagán, José L Risco-Martín, José M Moya, and José L Ayala. Modeling methodology for the accurate and prompt prediction of symptomatic events in chronic diseases. *Journal of biomedical informatics*, 62:136–147, 2016.
16. Josué Pagán, José L. Risco-Martín, José M. Moya, and José L. Ayala. A real-time framework for a devs-based migraine prediction simulator system. In *MAEB 2016*, 2016.
17. Chandrakant S. Ragit, Sameer Subhash Shirgaonkar, and Dr. Sanjay Badjate. Reconfigurable fpga chip design for wearable healthcare system. *International Journal of Computer Science and Network*, 4(2):208–212, 2015.
18. Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
19. José L. Risco-Martín, Saurabh Mittal, Juan Carlos Fabero, Pedro Malagón, and José L. Ayala. Real-time Hardware/Software Co-Design Using DEVS-based Transparent M&S Framework. In *Proceedings of the 2016 Summer Simulation Multiconference (SummerSim 2016)*, 2016.
20. José L. Risco-Martín, Saurabh Mittal, Juan Carlos Fabero, Marina Zapater, and Román Hermida. Reconsidering the performance of DEVS modeling and simulation environments using the DEVStone benchmark. *SIMULATION*, 93(6):459–476, 2017.
21. Lars Jacob Stovner and Colette Andree. Prevalence of headache in europe: a review for the eurolight project. *The journal of headache and pain*, 11(4):289–299, 2010.
22. Peter Van Overschee and Bart De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.
23. S. Warbhe and S. Karmore. Wearable healthcare monitoring system: A survey. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pages 1302–1305, Feb 2015.
24. Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation. Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2 edition, 2000.