# Development of a Design Methodology for Digital Measurement Instrumentation Using Complex Programmable Logic Devices

**Jorge A. VALERIANO, Felipe R. LARA, Norma R. CHAVEZ**
**Departamento de Computación, Facultad de Ingeniería, Universidad Nacional Autónoma de México. assem@servidor.unam.mx**
**04510 / DF, México**

## ABSTRACT

In this work, we present the development of a methodology that is a practical guide for the design and implementation of digital measurement instruments, using Programmable Logic Devices (PLD´s). Now days the programmable logic technology is a real alternative in the design of a digital systems. This is because of the growing complexity, variable specifications in the development cycle, shorter development time, and the necessity of lower costs. Additionally there are some requirements like confiability and testability. The general goal is to have a universal solution. Objectives of this methodology are: first, reduce time since the problem definition until the manufacturing of the final product; second, define the best use of hardware description language (HDL) to describe behavior.

**Keywords**: CPLDs, FPGAs, HDLs, EDA.

## 1. INTRODUCTION

The available software tools in the market for design using programmable logical devices presents a typical and widespread methodology to design any digital system. Most of these tools support designs using different strategies of modeling, but they never indicate which is the more ones or the less appropriate one, since this gives freedom to the designer of using the one that better it dominates or even a hybrid one of all them.

Some other tools that use some Hardware Description Language to model the digital system, present different levels of modeling, as well as some advantages and comparative disadvantages; however they are not a guide that facilitates the election of the most appropriate technique. A bad election of the method of modeling brings a considerable increment in the time and effort required to complete the development of the system, therefore this election is very important in any design. After having the modeling system, the rest of the process involves actions like the compilation, synthesis, simulation and device programming that are quite similar in all the tools. In this work we present a useful methodology for the design of digital measurement instruments. The essential objectives of this methodology are:

- Ø To reduce the time and effort from the specification of the instrument until their operation like final product, taking advantage of certain techniques of modeling.
- Ø To show considerations and guides to the moment to design types blocks that are common among instruments, like they can be the modules of interfaces like input - output.

## 2. PROPOSED METHODOLOGY

In our environment a design methodology is constituted by the following: Group of tools + Restrictions + Flow Design), where:

**Group of tools.** They are the applications offered by the software like graphical capture until programming tools.
**Restrictions.** They are limitations imposed by the used technology (CPLDs, FPGAs) like for the specifications of the system to design.
**Design flow.** It is the algorithm that constitutes the dorsal thorn of the methodology; it is determined partly by the configuration of the software used, and essentially for the orientation given by the designer. Our methodology is summarized in this section.

In parallel to the presentation of the methodology will go it applying to a practical case that it consists on designing a digital frecuencimeter to measure frequencies from 1Hz to 10 MHz.

### 2.1. Specification and Analysis of the Instrument

The methodology leaves from the specifications of the instrument to build and therefore before beginning with any stage of modeling we have to evaluate the factibility of the requested specifications.

*Specification of the Instrument.*

Aspects to consider when specifying the instrument or to evaluate the specifications given by the client are:

- Ø **Instrument operation speed (Bandwide).**

This characteristic depend of the variable or physical variables that is necessary to measure, we can be speaking of very quick variables or of very slow variables. It is clear that this characteristic is function of the device cost, therefore it is very important maybe to request a re evaluation of the instrument specifications with the objective of avoiding requirements that increase the cost of the instrument.

- **Resolution of the input variables.**

It is clear that the real measurement instrument is not 100% digital, this is the "core" of the instrument it can be 100% digital. However as the objective of the mensurement instrument it is to measure real variables, these almost always consist of analogical signs, this way it is necessary to evaluate the output resolution required by the instrument, for this way to be able to determine the resolution required in the input variables. In a directly proportional way to the number of input variables and of the resolution required by each one of them it is the size required by the logical device in relation to the number of input - output pines.

- **Precision, accuracy and tolerance of the instrument.**

The specifications of the instrument should also be made to three basic and well-known characteristics in the instrumentation. It should have specified the precision of the instrument (repetibility) of the mensurement. It should specify the accuracy (the proximity that provides the instrument in their mensurement respect to the real value). Additionally it should specify the tolerances.

*Analysis of the Instrument*

The analysis stage has the responsibility of separating the "core" of the instrument of the rest of the architecture.

- ***To have clear the instrument that we want to build.*** This means on one hand to know and to understand all the functional specifications of the instrument, and in second place to dominate the principle and the basic theory of the instrument that we are designing.
- ***Definition of the "core" of the instrument.*** The "core" of the instrument represents the heart that will be programmed in the programmable logical device and that it constitutes the medullary part of the digital instrument. The "core" can be constituted by an algorithm specified in high level and that finally it should be implanted in hardware.
- ***Definition of the input interfaz.*** Of all the mensurement instruments we can extract a well-known common denominator called input interfaz through which the information of the variable or variables to measure are provided to the "core" of the application.
- ***Definition of the output interfaz.*** Another common denominator in the digital mensurement instruments is the output interfaz, through the one which the information and the results of the "core" are showed to the user.

### 2.2. Instrument Modelling

This stage seeks to model three fundamental parts of any digital instrument

- ***Core of the instrument.*** It constitutes the heart of the application, we speak of a nucleus 100% digital that will be modeled to be implemented in programmable logical devices.
- ***Input Interfaz.*** The instrument will measure input variables so that the input interfaz accept the entrance variables and provide them in the form that the "core" requires them.
- ***Output Interfaz.*** The results of the mensurement makes by the "core" thave for objective to be visualized as a result of the process, more than to be used as control variables, so that we require of an interfaz that shows the results that in general it is 100% digital and it can be built totally in the programmable logical device.

To make the modeling of these three stages exist diverse technical; our methodology proposes the employment of standardized languages of hardware description specially VHDL and Verilog, using a behavior description style and possibly of data flow. The reasons to use this form of modeling are the following ones:

- The behavior description is a description of high level.
- The behavior description gives clarity to the design.
- VHDL and Verilog, are languages standardized by IEEE and using a subset of them guarantees portability among the diverse platforms of existent programmable logical devices in the market.
- VHDL and Verilog allow to describe complex systems using behavior description in a simple way.

This type of modeling consists on describing the behavior of the digital circuit, using some hardware description language. It is very important to highlight that structure is not described but alone the behavior of the system that is sought to model. This description one can make with different degrees of abstraction, in general a description with more degree of abstraction, is best for the user and it describes much more things, allowing to reduce times of design. The different degrees of abstraction of this domain are:

- **Differential equations.** It is a mathematical model of the behavior of the modeling entity, for example of a transistor. To model at this detail level is not necessary in most of the applications.
- **Logical equations.** It is a mathematical model using boolean algebra that describes the behavior of the modeling entity. The degree of abstraction is enough to model some digital systems.
- **Registers Transfer Language.** It is a form of modeling indicating the flow of the data in the design from a point to another by means of sentences of high level. To model this way allows to obviate non necessary details of digital components used with a lot of frequency in the design of digital systems.
- **Algorithms.** It is the most abstract form of modeling a digital system, because we are describing through an algorithm like we make it in the programming languages like "C" or Pascal, the behavior and procedure carried out by the digital system that we seek to model. This type of modeling allows to optimize design time and effort; however a bad

algorithms although easy to generate it can produce not very good hardware and the solution can be expensive in space and therefore in cost.

The next code is shown in VHDL the core and the input – output interfaces, for the example instrument.

***Core of the Frecuencimeter***

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY frecuencimetro IS
   PORT(
        reloj, desconocida          :IN
          STD_LOGIC;
        dividido                    : OUT   STD_LOGIC;
        m0, m1, m2, m3, m4, m5, m6  : OUT  INTEGER
RANGE 0 TO 9);
   END frecuencimetro;

ARCHITECTURE nucleo OF frecuencimetro IS
   SIGNAL relojbase : STD_LOGIC;
   SIGNAL cero, uno, dos, tres, cuatro, cinco, seis :
INTEGER RANGE 0 TO 9;

BEGIN
          PROCESS (reloj)         -- proceso "A"
          VARIABLE cuenta : INTEGER RANGE 0 TO
8000000;
          BEGIN
            IF (reloj'EVENT AND reloj = '1') THEN
               IF (cuenta < 8000000) THEN
                  relojbase <= '0';
                 cuenta := cuenta+1;
              ELSE
                relojbase <= '1';
                cuenta := 0;
              END IF;
           END IF;
END PROCESS;

dividido <= relojbase;          -- asignación "B"

PROCESS (desconocida, relojbase)      -- proceso "C"
   BEGIN
     IF (desconocida'EVENT AND desconocida = '1')
THEN
               IF (relojbase = '0') THEN
           cero <= cero+1;
        END IF;

        IF (relojbase = '0' and cero=9) THEN
           uno <= uno+1;
           cero <= 0;
        END IF;

        IF (relojbase = '0' and uno=10) THEN
           dos <= dos+1;
           uno <= 0;
        END IF;

        IF (relojbase = '0' and dos=10) THEN
           tres <= tres+1;
           dos <= 0;
        END IF;

        IF (relojbase = '0' and tres=10) THEN
           cuatro <= cuatro+1;
           tres <= 0;
        END IF;

        IF (relojbase = '0' and cuatro=10) THEN
           cinco <= cinco+1;
           cuatro <= 0;
        END IF;

        IF (relojbase = '0' and cinco=10) THEN
           seis <= seis+1;
           cinco <= 0;
        END IF;

        IF (relojbase = '0' and seis=10) THEN
           seis <= 0;
        END IF;

        END IF;
        IF (relojbase'EVENT and relojbase = '1') THEN
           m0 <= cero; m1 <= uno; m2 <= dos; m3 <=
tres; m4 <= cuatro; m5 <= cinco; m6 <= seis;
        END IF;
 END PROCESS;
END nucleo;
```

***Input – Output Interfaz of the Frecuencimeter***

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY interfazsalida IS

             PORT
             (
                          reloj     : IN       STD_LOGIC;
                          valores                          :IN
             STD_LOGIC_VECTOR (0 to 27);
                          salida_bcd                     :  OUT
             STD_LOGIC_VECTOR (0 to 6);
                       selector                          :  OUT
STD_LOGIC_VECTOR (0 to 6)
             );

END interfazsalida;

ARCHITECTURE interfaz OF interfazsalida IS
SIGNAL cuenta : INTEGER RANGE 0 to 6;
SIGNAL temporal : STD_LOGIC_VECTOR (0 to 3);
BEGIN
   PROCESS (reloj)   --proceso "A"
   BEGIN
      IF (reloj'EVENT and reloj='1') THEN
         cuenta <= cuenta + 1;
      END IF;
      IF (cuenta = 7) THEN
        cuenta <= 0;
      END IF;
    END PROCESS;
```

```
  PROCESS (reloj)  --proceso "B"
  BEGIN
    IF (reloj'EVENT and reloj='1') THEN
      CASE cuenta IS
              WHEN 0 => selector <= "0000001";
              temporal <= valores(0 to 3);
              WHEN 1 => selector <= "0000010";
              temporal <= valores(4 to 7);
              WHEN 2 => selector <= "0000100";
              temporal <= valor es(8 to 11);
              WHEN 3 => selector <= "0001000";
              temporal <= valores(12 to 15);
              WHEN 4 => selector <= "0010000";
              temporal <= valores(16 to 19);
              WHEN 5 => selector <= "0100000";
              temporal <= valores(20 to 23);
              WHEN 6 => selector <= "1000000";
              temporal <= valores(24 to 27);
      END CASE;
    END IF;
  END PROCESS;

  PROCESS (temporal)   -- Proceso "C"
  BEGIN
    CASE temporal IS
      WHEN "0000" => salida_bcd <= "0000001";
      WHEN "0001" => salida_bcd <= "1001111";
      WHEN "0010" => salida_bcd <= "0010010";
      WHEN "0011" => salida_bcd <= "0000110";
      WHEN "0100" => salida_bcd <= "1001100";
      WHEN "0101" => salida_bcd <= "0100100";
      WHEN "0110" => salida_bcd <= "1100000";
      WHEN "0111" => salida_bcd <= "0001111";
      WHEN "1000" => salida_bcd <= "0000000";
      WHEN "1001" => salida_bcd <= "000110 0";
      WHEN others => salida_bcd <= "1111111";
    END CASE;
  END PROCESS;

END interfaz;
```

### 2.3. Device CPLD selection

This stage of the methodology is not critical, however it impacts some final aspects considerably, being the most important the cost. To choose the device for an application we should consider the following three essential characteristics.

- *Time restrictions*

All programmable logical devices have time specifications, this is a measure of the maximum speed of operation of the device. This means that we should evaluate the maximum limit of operation speed required in the instrument to discard those devices that don't fulfill the requirement. Our methodology recommends to choose devices that at least they have more than 20% of aditional area with the purpose of possible future expansions.

- *Area restrictions*

The great diversity of available devices in the market, even for oneself company, is due to the area capacity of the device. They exist from those in those that we can program combinationals circuits and sequential simple, until those that can contain the design of a complete and complex microprocessor. The cost of these devices is directly proportional to the capacity of the same one, the unit commonly used to refer to the capacity is for the number of gates that you can to program inside the device, or for the bits number RAM that can be stored in the same one. Our methodology recommends that a device be chosen with 20% of free space for possible modifications, corrections or version expansion after being programmed with the design.

- *Costs Restrictions*

When the design specification implies a cost restriction it is necessary for example instead of using a single device of more capacity, to use two or three smaller devices whose total cost is inferior to one of more capacity. The case is also given when the election of a device of more capacity not increse the cost for the device but for example for the cost of generating the PCB that could require surface technology for example. Our methodology always recommends to leave at least 5% below the cost give in the specification to have a small tolerance in the event of requiring modifications of last hour.

### 2.4. Compilation and Synthesis

This stage of the methodology depends completely on the tool CAD-EDA that we are using in the process of design of our instrument. In the market several software tools exist, for example Altera, Xilinx, etc.

**Compilation.** It receives the description of the design in graphic format or in a hardware description language and analyzing the syntax and semantics to determine if the represented design is free of errors and it represents factible digital circuits of implementing in programmable logical devices.
**Synthesis.** This process has for objective to generate the designed circuit in an appropriate way so that it is possible to implement it in the technology of programmable logical devices supported by the tool. In other words this process translates the design compiled to a representation using the available basic functional blocks in the technology.

### 2.5. Digital simulation

This stage of the methodology depends essentially on two fundamental parts. The first one is dependent directly of the simulation tool. The second depend on the designer's strategies to optimize the simulation process fundamentally in time.

**First part.** It is important to distinguish between functional simulation and digital simulation. In the functional simulation the operation of the design is verified taking into account the ideal behavior of all the used digital devices. In the digital simulation besides considering the operations of all and each one of the digital devices used also takes into account, for example, the propagation times of all the digital components, as well as restrictions characteristic of the technology to use, this way are possible to obtain a very similar simulation to the real operation of the final device.

**Second part.** It depends essentially from the convenient strategies to use to optimize the simulation process, overalls when high times of simulation are required in relation to the frequency of the designed system. Some ideas provided by this methodology are the following ones:

- In the case of requiring high times of simulation in relation to the operation frequency bases of the device, it is convenient to scale the frequency of the device.
- The first simulation should contain only the input and output of the instrument core, that is we should center the effort in verifying the operation of the core, isolating the input - output interfaces.
- If the results of the simulation are not the prospective ones depurating strategies should be used that consist not only visualizing the inputs – outputs result of the core simulations, but also intermediate signs that allow us to determine the cause of the errors.
- We recomend to use sincronous designing.

The Fig. 1 illustrate the results of the simulation process using the software Max+Plus II, for the VHDL code described before.

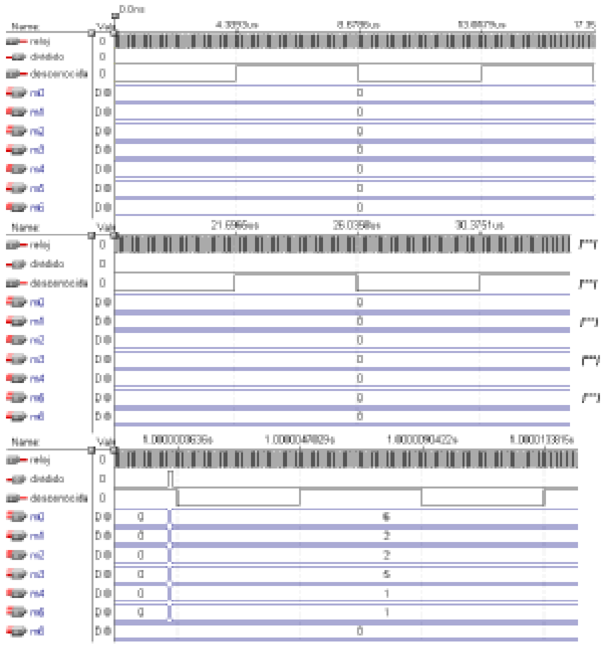

*Fig. 1. Behavior simulation for the instrument core.*

### 2.6. Device Programming.

This stage of the methodology adapts to the tool that we are using as well as to the technology of devices. Most of the tools as a result of the compilation process and synthesis programming generate files that are used directly by the tool to program the device. Some strategies suggested by the present methodology to optimize the pines assignment are the following ones:

- During all the previous stages and in essence until the results of the simulation process are the optimal, a manual assignment of pines it should not be made.
- When the results of the simulation process are satisfactory we can begin to plan manually the reassignment of input – output pins.
- The pines reassignment should begin to place in adjacent form all those signs that are a bus or that they could be organized as such.

### 2.7. Test and Debug

This stage of the methodology has for objective to make tests and debug the real physical device programmed. This process can be made in two ways.

- **Quick prototype.** It consists on already using the device programmed and to either verify their functionality on a protoboard or using some development card sold by the same companies that manufacture the programmable devices as well as the software tools.
- **Prototype PCB.** It consists on manufacturing a printed circuit board to place our programmable device with our design. This form has the advantage of generating a final prototype that is technically already the final product. The disadvantage is that it is a slower process that implies the design and production of the PCB.

The definitive tests are carried out on the real prototype and an operation is expected the same as the one obtained by the simulation stage. Most of the times, the results of this stage will be correct. In the case of problems, some strategies are:

- To determine if the undesired operation is due to a vulnerable condition of the design like the case of an asyncronism, or noise problems induced to our system by external sources.
- To consider all the problems indirectly introduced by any switch as input to our system.
- To consider the problems of noise induced by switches in the periphery of our system.
- Bypassig appropriately all the included digital devices to our programmable logical device

If the result of this stage doesn't provide the results obtained previously with the tips it will be necessary to return to the stage of modeling or even to the re-specification and analysis of the problem. In the Fig. 2 are illustrated the frecuencimeter working .

## 3. CONCLUSIONS

The objective of the methodology is to reduce the development cycle and implementation of digital instrumentation using the technology of the complex programmable logical devices (CPLDs). The methodology that we propose has the following advantages that are clearly during the development of the work:

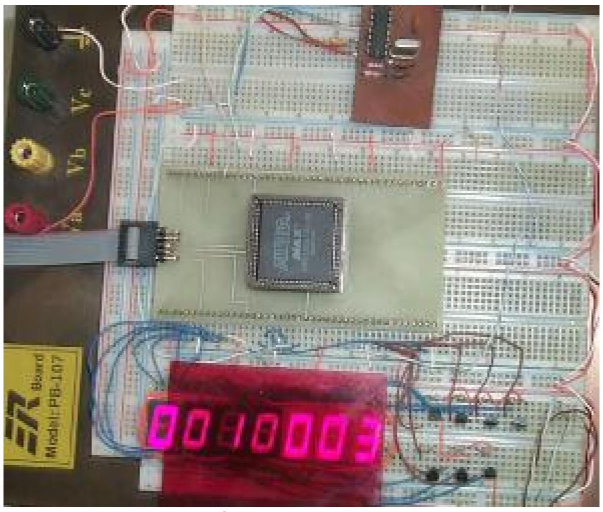

*Fig. 2.The frecuencimeter instrument.*

- It is guided to the design of digital instrumentation using Complex Programmable Logical Devices.
- It subdivides any type of digital instrument in three parts, the first call “core”, the second call “input interfaz” and the third the “output interfaz.”
- The input – output interfaces can be of different types, however they can be the same ones among different instruments. This implies that the interfaz is interchanged.
- The methodology therefore outlines like medullary part the development is the “core” that defines the name of the instrument.
- The hardware descriptions is best using behavior. This allows to minimize the time of design, as well as to facilitate the debug process.
- The inherent simulation to this technology is optimized when using behavior description, because the necessary changes imply modification of code source like in the software programming languages.
- The behavior description use hardware description languages standardized as VHDL or Verilog HDL, because this allows that a design can be migrated to other platforms.
- The behavior description minimizes the cycle of development of an instrument.

## 4. REFERENCES

[1] Valeriano Assem Jorge, Chávez Rodríguez Norma Elva, Haro Ruiz Arturo. *“Lenguaje de Descripción de Hardware Verilog”*, 49 Págs. Facultad de Ingeniería UNAM, División de Ingeniería Eléctrica, Departamento de Ingeniería en Computación e Ingeniería Electrónica. Junio 2001.
[2] Valeriano Assem Jorge, Chávez Rodríguez Norma Elva. *“Lenguaje de Descripción de Hardware VHDL”*, 69 Págs. Facultad de Ingeniería UNAM, División de Ingeniería Eléctrica, Departamento de Ingeniería en Computación e Ingeniería Electrónica. Agosto 2001.
[3] Altera Corporation. *Data Book.* (San Jose California1999).
[4] Fuentes R., Valeriano J. *“Monitor Digital de Presión Sanguínea”*. Memorias SOMI XVI Congreso de Instrumentación. (Querétaro, Querétaro, México, Octubre 2001).
[5] Fuentes R., Valeriano J. *“Implantación Digital de un Cronómetro Múltiple en CPLDs”*. Memorias SOMI XV Congreso de Instrumentación. (Guadalajara, Jalisco, México, Octubre 200).
[6] Chávez Rodríguez Norma Elva, Valeriano Assem Jorge, Haro Ruiz Arturo. *“Manual de consulta entorno de diseño MAX + PLUS II”*, 66 Págs. Facultad de Ingeniería UNAM, División de Ingeniería Eléctrica, Departamento de Ingeniería en Computación e Ingeniería Electrónica. Febrero 2001.
[7] Chávez Rodríguez Norma Elva, Valeriano Assem Jorge. *“Prácticas de Laboratorio en el Entorno de Diseño MAX+PLUS II”*, 29 Págs. Facultad de Ingeniería UNAM, División de Ingeniería Eléctrica, Departamento de Ingeniería en Computación e Ingeniería Electrónica. Agosto 2001.
[8] Cesare Alippi, et.al. *“A Composite System Design Methodology for Instrumentation and Embedded System*“. IEEE. 2000.
[9] Joseph Cerra. *“HDL Methodology Offers Fast Design Cycle and Vendor Independence*”. Actel Corporation. Abril, 1996.
[10] Joannis Papanuskas, et.al. *“A Uniform Design Methodology for Application Specific Digital Integrated Circuits in Automotive Applications”*. IEEE. 1995.
[11] Synthesis Methodology Guide. Actel. Synopsys.
[12] Xilinx. *“Embedded Instrumentation Using XC9500 CPLDs”*. 1997.
[13] Daniel D. Gajski, et.al. *“System Design Methodologies: Aiming at the 100 h Design Cycle”*. IEEE. 1996.