CASE 2016

2016 Seventh Argentine Conference on Embedded Systems (CASE)

Regular papers.

www.sase.com.ar

University of Buenos Aires, Argentina

IEEE Catalog Number CFP1646V-PRT I ISBN 978-987-45523-9-6

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)

Regular Papers

August 10th-12th, 2016
School of Engineering – University of Buenos Aires
Buenos Aires, Argentina

2016 Seventh Argentine Conference on Embedded Systems CASE : regular papers / José Lipovetzky, Maximiliano Antonelli, Diego Brengi, Luciana De Micco, Mariano García Inza, Ariel Lutenberg ... [et al.]. - 1a ed ilustrada. - Ciudad Autónoma de Buenos Aires : ACSE - Asociación Civil para la investigación, Promoción y Desarrollo de Sistemas Eléctricos Embebidos, 2016.
50 p. ; 29 x 20 cm.

ISBN 978-987-45523-9-6

1. Ingeniería. 2. Actas de Congresos. I. Lipovetzky, José
CDD 629.8

ISBN Date: 21/07/2016

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)

Regular Papers

Printed Book ISBN 978-987-45523-9-6
E-Book ISBN 978-987-46297-0-8

Printed Book IEEE CATALOG CFP1646V-PRT
IEEE XPLORE COMPLIANT CFP1646V-ART

Editors:

Antonelli, Maximiliano. UNMDP
Brengi, Diego. INTI/UNLaM
De Micco, Luciana. UNMDP/CONICET
García Inza, Mariano. FIUBA
Lipovetzky, José. IB/CNEA/CONICET
Lutenberg, Ariel. FIUBA/CONICET/ACSE

Preface

The development of Embedded Systems is crucial for the development of industry, technology and science; and it is an area which has grown in the past years in Argentina and South America.

The SASE and CASE are intended to promote the development of embedded systems in the country and the region with the following activities:

- Presentation of scientific and technological works at the conference.
- Hands-on Workshops.
- Technical Lectures (Tutorials).
- Plenary sessions.
- Contest for students projects
- A program to assign electronic equipment to Universities.
- Travel and lodging grants for graduate and postgraduate students, professors and researchers of Argentina.

The objectives of the SASE are:

- To allow a more fluid exchange between academia and industry.
- To promote the exchange between researchers and students from different universities from Argentina and other countries.
- To allow the diffusion of scientific and technological development done in the region and worldwide.
- To encourage students from the country to get interested in the development of Embedded Systems.
- To coordinate and promote the actualization of curriculum contents related to Embedded Systems in undergraduate and graduate programs at the universities of Argentina.

This year, topic areas included at CASE are: Architecture of Microprocessors, ASICs, DSPs, FPGA and HDLs, Implementation of Embedded Systems, Embedded Systems Linux, Communications and Protocols, Embedded Software, Robotics, RTOS and Bioengineering. Scientific works were accepted in three categories, Regular Papers, Technical Forum, and Poster. This book contains the works accepted as Regular Papers. After an exhaustive peer review process only seven papers were accepted in this distinguished category and published here. Some of the other ones were moved to the categories of technological forum or poster.

We hope you enjoy the following papers, and the conference.

CASE 2016 Organizing Committee

Sponsors

Diamond Sponsors

- Asembli S.A.
- Cika Electrónica S.R.L.
- Electrocomponentes S.A.
- Intel
- Synopsys
- Vicda Argentina

Gold Sponsors

- CADIPEL
- Ernesto Mayer S.A.
- NXP
- Semak
- Sur Emprendimientos Tecnológicos
- Telit

Silver Sponsors

- Dai Ichi Circuitos
- L&R Ingeniería

Institutions which support SASE

Organizing Istitution

- ACSE (Asociación Civil para la Investigación, Promoción y Desarrollo de los Sistemas Electrónicos Embebidos)

Co-organizing Institutions

- CADIEEL (Cámara Argentina de Industrias Electrónicas, Electromecánicas y Luminotécnicas)
- RUSE (Red Universitaria de Sistemas Embebidos)

Sponsoring

- ADIMRA (Asociación de Industriales Metalúrgicos de la República Argentina)
- ANPCyT (Agencia Nacional de Promoción Científica y Tecnológica)
- CAPER (Cámara Argentina de Proveedores y Fabricantes de Equipos de Radiodifusión)
- CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas)
- ISOC (Internet Society)

Supporting

- AADECA (Asociación Argentina de Control Automático)
- CAI (Centro Argentino de Ingenieros)
- CAME (Confederación Argentina de la Mediana Empresa)
- CAMOCA (Cámara Argentina de Máquinas de Oficina, Comerciales y Afines)
- CASEL (Cámara Argentina de Seguridad Electrónica)
- CEIL (Cámara de Empresas Informáticas del Litoral)
- CESSI (Cámara de Empresas de Software y Servicios Informáticos)
- CIIECCA (Cámara de Industrias Informáticas, Electrónicas y de Comunicaciones del Centro de Argentina)
- CITEDEF (Instituto de Investigaciones Científicas y Técnicas para la Defensa)
- CNEA (Comisión Nacional de Energía Atómica)
- CONFEDI (Consejo Federal de Decanos de Ingeniería)
- Fundación Sadosky (Investigación y Desarrollo en TIC)
- IEEE Argentina (Institute of Electrical and Electronics Engineers)
- INTI (Instituto Nacional de Tecnología Industrial)
- MinCyT (Ministerio de Ciencia, Tecnología e Innovación Productiva)
- ORT (Instituto de Tecnología ORT)

Supporting Universities

- UBA: Universidad de Buenos Aires
- UNAJ: Universidad Nacional Arturo Jauretche
- UNC: Universidad Nacional de Córdoba
- UNCA: Universidad Nacional de Catamárca
- UNCOMA: Universidad Nacional del Comahue
- UNCUYO: Universidad Nacional de Cuyo
- UNER: Univerisdad Nacional de Entre Ríos
- UNICEN: Universidad Nacional del Centro de la Provincia de Buenos Aires
- UNLAM: Universidad Nacional de La Matanza
- UNLP: Universidad Nacional de La Plata
- UNLu: Universidad Nacional de Luján
- UNNE: Universidad Nacional del Nordeste
- UNNOBA: Universidad Nacional del Noroeste de la Provincia de Buenos Aires
- UNM: Universidad Nacional de Misiones
- UNMDP: Universidad Nacional de Mar del Plata
- UNPA: Universidad Nacional de la Patagonia Austral
- UNPSJB: Universidad Nacional de la Patagonia San Juan Bosco
- UNQ: Universidad Nacional de Quilmes
- UNR: Universidad Nacional de Rosario
- UNRC: Universidad Nacional de Rio Cuarto
- UNS: Universidad Nacional del Sur
- UNSA: Universidad Nacional de Salta
- UNSJ: Universidad Nacional de San Juan
- UNSL: Universidad Nacinal de San Luis
- UNSM: Universidad Nacional de San Martín
- UNT: Universidad Nacional de Tucumán
- UNTF: Universidad Nacional de Tres de Febrero
- UTN-FRA: Universidad Tecnológica Nacional – Facultad Regional Avellaneda
- UTN-FRBA: Universidad Tecnológica Nacional – Facultad Regional Buenos Aires
- UTN-FRBB: Universidad Tecnológica Nacional – Facultad Regional Bahía Blanca
- UTN-FRD: Universidad Tecnológica Nacional – Facultad Regional Delta
- UTN-FRH: Universidad Tecnológica Nacional – Facultad Regional Haedo
- UTN-FRLR: Universidad Tecnológica Nacional – Facultad Regional La Rioja
- UTN-FRM: Universidad Tecnológica Nacional – Facultad Regional Mendoza
- UTN-FRN: Universidad Tecnológica Nacional – Facultad Regional Neuquén
- UTN-FRP: Universidad Tecnológica Nacional – Facultad Regional Paraná
- UTN-FRRE: Universidad Tecnológica Nacional – Facultad Regional Resistencia

- UTN-FRRG: Universidad Tecnológica Nacional – Facultad Regional Rio Grande
- UTN-FRSF: Universidad Tecnológica Nacional – Facultad Regional San Francisco
- UTN-FRSN: Universidad Tecnológica Nacional – Facultad Regional San Nicolás
- UTN-FRVT: Universidad Tecnológica Nacional – Facultad Regional Venado Tuerto
- UTN-FRVM: Universidad Tecnológica Nacional – Facultad Regional Villa Mercedes
- UTN-FRT: Universidad Tecnológica Nacional – Facultad Regional Tucumán
- UADE: Universidad Argentina de la Empresa
- CAECE: Universidad CAECE
- FASTA: Fraternidad de Agrupaciones Santo Tomás de Aquino
- ITBA: Instituto Tecnológico de Buenos Aires
- IUA: Instituto Universitario Aeronáutico
- UBP: Universidad Blas Pascal
- UCC: Universidad Católica de Córdoba
- UCU: Universidad Católica de Uruguay
- UCSE: Universidad Católica de Santiago del Estero
- UM: Universidad de Mendoza
- UREP: Universidad de la República del Uruguay

Technical Program Committee 2016

SASE General Chair

- Dr. Ariel Lutenberg (FIUBA/CONICET/ACSE)

CASE Technical Program Chair:

- Msc. Diego Brengi (INTI/UNLaM, IEEE Senior Member)

CASE Publication Chairs:

- Dr. Luciana De Micco (UNMDP/CONICET, IEEE-CASS Member)
- Dr. José Lipovetzky (IB/CNEA/CONICET, IEEE-CASS Member)
- Dr. Mariano García Inza (FIUBA)
- Eng. Maximiliano Antonelli (UNMDP)

CASE Topic Chairs:

- Bioengineering: Eng. Juan Manuel Reta (UNER)
- DSPs: Dra. María Liz Crespo (ICTP)
- Embedded Linux: Luciano Diamand (UNR)
- Embedded Software: Dr. Ricardo Medel (Ascentio Technologies)
- FPGAs, HDLs and ASIC: Eng. Salvador Tropea (INTI/UTN-FRBA)
- Implementation of Embedded Systems: Msc. Cristian Sisterna (UNSJ)
  and Luciana De Micco (UNMDP/CONICET)
- Processor's architecture: Eng. Alejandro Furfaro (UTN-FRBA)
- Protocols and communications: Eng. Gustavo Mercado (UTN-FRM)
- Robotics: Claudio Verrastro (CNEA)
- RTOS: Dr. Ricardo Cayssials (UNS)
- Wireless communications: Dr. Leonardo Rey Vega (FIUBA)

Reviewers

Alcalde Bessia, Fabricio
Alessandrini, Gustavo
Antonelli, Maximiliano
Arias, Ricardo
Brengi, Diego
Briff, Pablo
Burgos, Enrique Sergio
Carbonetto, Sebastián
Carrá, Martín
Cayssials, Ricardo
Comas, Edgardo
Crespo, Maria Liz
De Micco, Luciana
Diamand, Luciano
Dondo, Julio
Escudero, Gustavo
Ferreyra, Pablo Alejandro
Ferro, Edgardo
Filomena, Eduardo
Fraire, Juan Andres
Furfaro, Alejandro
Garcia Inza, Mariano
Gavinowich, Gabriel
Golmar, Federico
Grimblatt, Victor
Hernandez Tabares, Lorenzo
Hidalgo, Roberto
Leiva, Lucas
Lipovetzky, Jose
Lutenberg, Ariel
Marchi, Edgardo
Martos, Pedro
Mata, Walter A.
Medel, Ricardo
Melo, Rodrigo Alejandro
Mercado, Gustavo
Morero, Damián
Oliva, Rafael
Orallo, Carlos Martin
Ovilla, Brisbane
Perez, Santiago
Perez Paina, Gonzalo F.
Petrashin, Pablo Antonio
Pucheta, Julian
Rey Vega, Leonardo
Reyes, Benjamin T.
Ridolfi, Pablo
Romeo, Marcelo
Sambuco Salomone, Lucas
Sisterna, Cristian
Sofo Haro, Miguel
Todorovich, Elías
Tropea, Salvador
Uriz, Alejandro J.
Verrastro, Claudio
Zacchigna, Federico G.
Zaradnik, Ignacio

Table of Contents

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

LabOSat: Low cost measurement platform designed for hazardous environments

M. Barella[1,3], G. A. Sanca[4], F. Gómez Marlasca[2], G. Rodríguez[1], D. Martelliti[1], L. Abanto[4], P. Levy[2,3], F. Golmar[1,3,4]

[1]Centro de Micro y Nano Tecnología del Bicentenario, INTI, San Martín, Bs. As., Argentina
[2]Centro Atómico Constituyentes, CNEA, San Martín, Bs. As., Argentina
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Bs. As., Argentina
[4]Escuela de Ciencia y Tecnología, UNSAM, San Martín, Bs. As., Argentina
mbarella@inti.gov.ar

***Abstract*—In this work the characteristics and performance of LabOSat platform to carry out experiments over electronic devices in aggressive environments are described. Configurable, portable, low weight and low cost are the main features. First measurements made in our laboratory are presented before testing in space.**

***Index Terms*—Measurement instrument, portable platform, space mission, non-linear devices, resistive switching.**

I. Introduction

LabOSat is a universal platform to perform experiments in harsh environments such as outer space, low pressure - low temperature conditions or high levels of radiation. LabOSat-01, the first version of the platform, is designed to harbour devices to be tested electrically under this kind of conditions. Inside LabOSat-01, there is a module, called MeMOSat, which forces with voltage or current customized sweeps ReRAM non-volatile memory devices and can be easily adapted to any two-terminal device. Moreover, this module, can run endurance tests to study how extreme conditions degrade the memory cells. Its predecessor, MeMOSat-1, is up and running inside "Tita", a BugSat manufactured by the argentinian company Satellogic [1] which is presently in a LEO (Low Earth Orbit) at approximately 500km.

Nowadays, two boards LabOSat-01 are being integrated on the new satellites that the company is developing and there is a third board executing the programmed experiments at our labs as control sample. Figure 1 shows the board with an inset of a ReRAM device.

In the following sections, a general description of system's hardware and software is given. Then, the capabilities of the board are explained followed by the description of the devices measured in this work. Finally, the characterization of current source and the implementation of LabOSat in the laboratory are presented. This work is the first step to validate LabOSat as an instrument platform for performing electronic measurements in hazardous environments.

II. System overview

A. Hardware description

The building blocks of LabOSat are depicted in Figure 2. The core of the board is a mixed signal microcontroller MSP430F1612 (Texas Instruments) which controls all the peripherals circuits (blocks) designated to specific tasks. This microcontroller is frequently used in Low Earth Orbit missions to drive CubeSats and was tested by NASA [2]. It uses 12-bit DACs outputs to precisely control the voltage or the current that will stimulate on-board devices. The 12-bit ADCs inputs are used to measure the response of the DUTs and other parameters that are periodically checked to guarantee the correct behaviour of the board. Finally, there are several addressing pins located at the GPIO ports that are dedicated to switch between blocks or modes of operation. The following are the most relevant blocks of LabOSat-01:

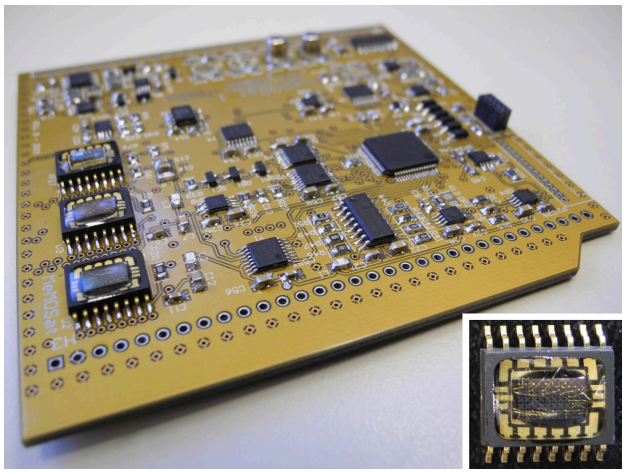


Fig. 1: LabOSat-01 board. Inset: encapsulated ReRAM device.

To measure temperature we use a LM74 thermometer (Texas Instruments) which communicates through SPI and is configured to measure temperatures over 0°C up to 125°C. Nevertheless, all commercial components of LabOSat are designed to operate in temperature ranges from -40°C to 85°C.

The dosimeters block, COTS pMOS transistors characterizated as radiation sensor, is designed to measure total ionizing radiation. This module is crucial for space applications or measurements inside a particle accelerator for example.

The MeMO block is dedicated to experiments over memory devices, particularly, devices based on Resistive Switching phenomena [3] which are expected to work well under big dose of radiation. Other devices could be embedded such as non-volatile memories like flash memories.

There is another experiment running inside LabOSat de-

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

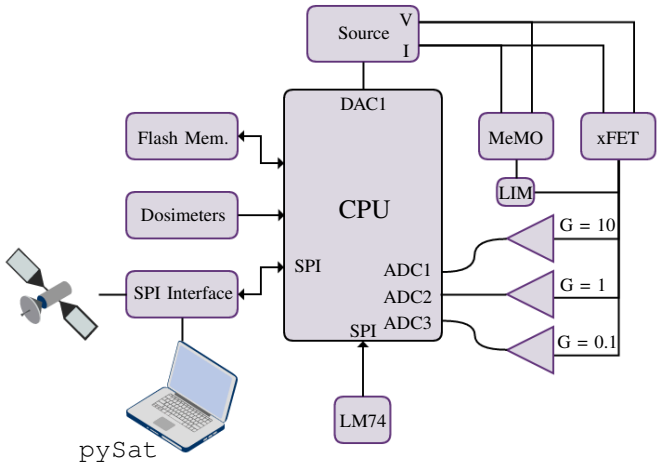


Fig. 2: Block diagram of LabOSat-01.

signed to study the performance of transistors. This block is named xFET and basically can test any transistor device, commercial or custom made.

The goal of the last two blocks is to electrically test devices degradation under hostile conditions. Stimuli to test devices can be delivered sourcing voltage or current and controlled with high precision using DAC outputs. When sourcing voltage, current limitation is possible. As said before, measurements are made using the ADC inputs and each of them is preceded by a buffer and an amplifier in order to gain or attenuate the signal giving wide range measurements and avoiding out-of-range ADC readings. The MeMO block allows operation over 28 devices and the xFET block over 6.

The current source circuit was inspired by the current loop used in the CIAA project [4] and was implemented with minor modifications to fulfill the requirements of ReRAM devices. This current block allows LabOSat to perform new experiments in MeMO module in contrast to its predecesor MeMOSat-1. When sourcing current, LabOSat can deliver up to 23mA with a maximum power of 230mW over DUT.

PCB is made of four layers where the outer ones are ground planes and the inner layers hold the tracks routed with Manhattan strategy to reduce crosstalk and parasitic capacities.

LabOSat also has an external port to perform MeMO or xFET remote experiments in a reduced-size board. The board weights 36g and it's dimensions are $100 \times 100 \times 15$mm which makes LabOSat a light-weight and portable platform.

B. Firmware description

When LabOSat-01 is turned on it starts in idle mode, waiting for instructions over SPI communication port (Figure 2). The firmware is written in C and is programmed to understand a short set of instructions. Most relevant are described in the following paragraph.

It's alive: First but not the most important instruction is `alive?`, which asks the board if it is listening to SPI port. This instruction must be executed until the board answers `alive` which could be translated as "I'm alive, synchronized and waiting for new instructions".

Execute Standard Test: The `stdTest` instruction or Standard Test routine comprises several steps where supply voltage of the board (V_{battery}), temperature (T), dosimetry (dosimeters' voltage) and response of ReRAM devices and transistors is measured. Figure 3 shows the pseudo-code of the Standard Test routine. This instruction can only be executed once a day.

```
stdTest request
if a day has passed then
  execute Standard Test
else
  do not execute Standard Test
end if
```

```
stdTest execution
read dosimetry, V_battery and T
for each ReRAM in bank 1 do
  measure IV or endurance test
end for
read V_battery and T
for each ReRAM in bank 2 do
  measure IV or endurance test
end for
for each xFET do
  measure modulation and output characteristic curve
end for
read V_battery and T
```

Fig. 3: Pseudo-code of Stadard Test request and execution.

How measurements are reported: The `sendReport` instruction requests the last report generated by LabOSat. It could be the report of a Standard Test or other instruction (not explained here). The length of the report is informed within the first bytes making the communication more robust by the implementation of a timeout.

C. Interface and Communication

All data is saved on microcontroller's flash memory after execution. To download, communication through SPI with `sendReport` command must be done. The length of the report depends on the kind of instruction sent and it is reported within the first bytes of data, inside the header of the report. The header also contains an identification number of the board, the report number, the CRC-16 of the report itself, a time stamp that indicates when the last instruction was executed and a status byte which indicates errors and logs. The calculation of report's CRC-16 is executed to check if memory or communication was corrupted by a bit flip as a result of ionizing radiation for example.

The decodification of the report is made with a Python script which facilitates the reading of the flash data and allows to see easily if the behaviour of each device differs from last execution.

The LabOSat-01 is programmed to produce reports of less than 3 kB because of satellite's telemetry bandwidth availability. On Earth, as we want to reproduce same operation

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

conditions (except the harsh environment) the report is downloaded with an emulated satellite made with an Arduino Nano using the built-in SPI to USB module and a script written in Python. The emulation program, i.e. communication and decodification, is called `pySat`.

III. Devices Under Test

A. Single-bit ReRAMs

The on-board DUTs are tri-layer structures composed of two metallic electrodes divided by an oxide thin film fabricated at our laboratories. The studies are focused on non stoichiometric titanium oxide (TiO_{2-x}) and manganite $La_{1/3}Ca_{2/3}MnO_3$. On this devices, the Resistive Switching works, taking advantage of oxygen vacancies inside the oxide cell. When an electric field is applied it produces the formation of mixed filamentary structures formed by these vacancies and metal atoms from the electrodes. This structure reduces the resistance of the dielectric oxide cell working as a parallel resistance of lower value. In some cases, depending on the applied field, the filamentary structure could short circuit the electrodes and leave the device in a very Low Resistance State (LRS).

The switching between different non-volatile resistance values or "states" is exhibited when this filamentary structure is disarmed (or even fused) and formed again. This is accomplished by applying voltage or current pulses of opposite polarity (bipolar Resistive Switching).

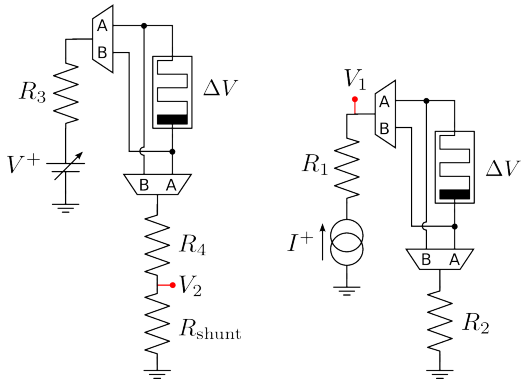
Measurement principle: The way we test degradation on devices is performing IV curves and endurance tests. The first allow us to study the dynamic response of the cell, precisely how SET (write) and RESET (erase) process evolve from first IV curve. The latter let us to understand how degradation modifies the device's performance as a memory unit working in a static way.

The devices can be controlled imposing a voltage or injecting current through it. In the first case, as can be seen from Figure 4a, the device is in series with a shunt resistance and the value of V_2 is measured to infer the voltage drop over the device. On the other hand, when current control is preferred, the device is subjected to the circuit showed in Figure 4b and V_1 gives information of the voltage drop over it. Either the case, the stimuli of the source will depend on the kind of experiment to be performed and the voltage or current value imposed by the source is set with the DAC output of the microcontroller. Each mode is configured with multiplexers and/or switches that are addressed with GPIO ports. As sources only apply positive voltage (current) two switches give the control of the polarity of the devices.

a) Voltage mode: the math to obtain the voltage drop and the current through the device when the voltage source is selected is shown in the following equations. The source sweep (V^+), parasitic resistances (R_3 and R_4) and shunt are known and V_2 is measured so Kirchhoff's law returns:

$$I_{\text{mem}} = \frac{V_2}{R_{\text{shunt}}} \tag{1}$$

$$V_{\text{mem}} = \Delta V = V^+ - V_2 - I_{\text{mem}}(R_3 + R_4) \tag{2}$$



(a) Voltage source. Measuring point = V_2.

(b) Current source. Measuring point = V_1.

Fig. 4: Simplified circuitry when device is being controlled by different sources. Labels "A" and "B" indicate the connections which select device's polarity.

b) Current mode: when current control is selected, the source sweep (I^+) and parasitic resistances (R_1 and R_2) are known and V_1 is measured instead. In this case, Kirchhoff's law returns:

$$I_{\text{mem}} = I^+ \tag{3}$$

$$V_{\text{mem}} = \Delta V = V_1 - I_{\text{mem}}(R_1 + R_2) \tag{4}$$

If polarity is inverted, either current or voltage mode, current flowing through memory can be considered negative taking into account the device Bottom and Top Electrodes. In consequence, voltage drop will be negative too.

Parasitic resistances arise from on-resistance of multiplexers and switches. Actually, LabOSat uses DG406 and ADG5413, respectively. Both of them have been characterized on Earth as a function of supply voltage at room temperature but, in order to obtain more accurate data, measured temperature and supply voltage can be used to extrapolate the tabulated data (datasheets available from Analog Devices and Maxim, respectevely). As a result, this will improve the estimation of voltage drops and currents.

As said in Section I, the devices can be subjected to two different kind of experiments: endurance test and IV test, static and dynamic response test, respectively.

The **Endurance Test** concerns application of successive pulses of opposite polarity in order to achieve a fixed number of commutations. For example, if the device is in High Resistance State (HRS) and one commutation is desired to occur, the test will run applying SET pulses until the transition to the LOW state takes place. In case of two commutations, after reaching the LRS, RESET pulses will be applied until high state is recovered.

The amplitude, duty cycle and period time can be configured and if the writing pulse (either SET or RESET) is done in voltage mode also the shunt resistance must be configured. Between each writing pulse a reading pulse is applied to measure if the state of the device has changed. Additionally, the algorithm behind the test requires two preset levels (states or resistance values) to define a window of operation. Is desired that each pulse commutate the device between levels

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

outside this window but, if is not the case, the number of tries, i.e. the number of pulses of SET or RESET pulses, can be augmented.

By the same token, the **IV Test** can be configured to stimulate the sample with a current sweep or a voltage sweep. The sweep is interpreted as a sequence of triangular stimuli, cast by the number of points (steps) and incremental positive and negative amplitude. As a result, the triangular sweep, starts from zero up to the maximum positive amplitude, then to maximum negative amplitude and back to zero. The maximum amplitude is defined by the *number of steps* multiplied by the *incremental amplitude*. Polarity of sweep can be set to start negative or positive. Henceforth, a starting positive voltage stimuli would be $0 \rightarrow V_{pos} \rightarrow 0 \rightarrow -V_{neg} \rightarrow 0$.

The sweep can be configured to read the resistance state between sweep steps. Also if voltage control is selected the value of the shunt resistance must be set. When the current sweep is selected the shunt parameter is ignored.

In addition, as this platform is designed to work under aggressive environments, storage of measurements is limited due to restriction in download bandwidth. Because of this, the *number of steps per record* is another parameter to set when IV test configuration comes into play. This parameter is related with the *number of steps* to be applied because it determines how many steps would be recorded.

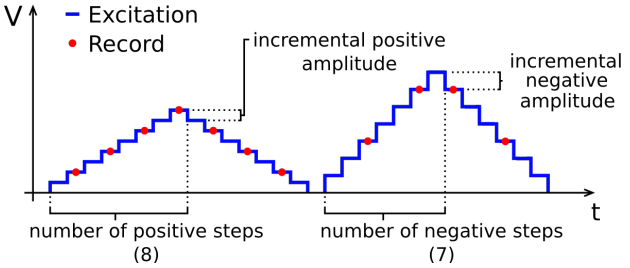


Fig. 5: Example of a voltage controlled IV test. Positive stimuli is conformed of 15 steps (8+7) and the negative of 13 steps (7+6). When second stimuli starts the device is inverted for the purpose of *seeing* the stress as negative. Recording points were chosen to illustrate the possibility of sampling (or not) the response at maximum stimulus.

To give an example on Figure 5 a voltage sweep of 15 positive steps (the configured 8 ascending steps plus 7 descending steps) and 13 negative steps (the configured 7 ascending steps plus 6 descending steps) with different incremental amplitude. When second sweep starts LabOSat swaps the polarity of the device in order to apply a negative stimuli. The number of *steps per record* was set to 2 in the positive excursion and 3 in the negative. As the number of positive steps is divisible by 2, LabOSat will register the measurement made when maximum positive pulse is applied. In contrast, as the number of negative steps is not divisible by 3 the response for the maximum negative value is not recorded. This example makes clear that forcing sweeps and reported data can be configured independently. This option gives LabOSat a great flexibility when managing big amount of data (long sweeps). Furthermore, when working with devices which exhibit hysteresis this possibility allows to report specific responses in order to save storage space still revealing the device behaviour.

Coupled with this configuration LabOSat software allows to select how many ADC readings are going to be done in each measurement. Possible values are one, two or four readings. After acquisition, the readings are averaged and reported as a single value.

B. Transistors

The fabrication of these multi-finger transistors use non-CMOS compatible materials and were fabricated within a cleanroom facility.

Measurement principle: Experiments on transistors are configurable to perform voltage sweeps on drain with fixed gate voltage and the opposite, sweeps on gate with fixed voltage over V_{xFET}, Figure 6. Source voltage, V_S, is set to 1V but can be simply configured via hardware to any desired value. Measurements are taken from V_D node and converted to drain current using the known value of R_D and the applied V_{xFET}:

$$I_D = \frac{V_{\text{xFET}} - V_D}{R_D} \tag{5}$$

This let the user plot two distinctive curves of the transistor: the output characteristic curve (I_D vs. V_{DS}) and the transfer curve (I_D vs. V_{GS}).

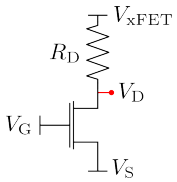


Fig. 6: Configurable experiment over custom transistor.

Encapsulation

As devices are custom-made within cleanroom facilities an encapsulation step is needed before integration to LabOSat. First, fabricated samples are diced to small dies to fit in a commercial SOIC-16 where they will be wire bonded. Second, an epoxy over the bonded devices is placed to protect them from physical damage. After the curing of epoxy the encapsulated devices are submitted to vacuum test up to 10^{-6}mbar and harmonic and random shaking test from 10Hz to 1kHz with 6.8**g** (RMS) maximum amplitude.

Between every fabrication step, from deposition over silicon wafer to epoxy protection on SOIC, an electric test is performed to see if the device is still working as expected. A picture of a device after the whole fabrication process is shown in Figure 1 (inset).

C. Dosimeters

LabOSat-01 has two commercial pMOS transistors that were characterized before integration to boards. These devices work as dosimeters, in the sense that incident radiaton dose can be infered from periodic measurements of the threshold voltage. As V_{th} is affected by the Total Ionizing Dose (TID), its shift can be used to estimate the dose absorbed by the board. This parameter is measured indirectly by biasing the source of the pMOS with a known current and voltage (I_S and V_S) while the drain is grounded. Then, reading the gate voltage allows to observe variations in V_{GS} which will be traduced to V_{th} variations [5], [6].

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

IV. Results

Linearity of the platform

One of the improvements of LabOSat-01 with respect to MeMOSat-1 is the possibility of forcing current through the ReRAM devices. Characterizing this block is critical to understand how measurements are made. The linearity of the current block was checked forcing current sweeps of half DAC range to a 1kΩ resistor while sampling ADC inputs and the DAC output with a Keithley SCS-4200. Also, LabOSat was asked to report the results in order to compare both acquisitions.

The response of the current source and acquisition stages was found to be linear in the range 0 - 2.5V. The data measured by LabOSat is in ageement with the curves acquired with the SCS-4200. The characterization of this block can be seen in Figure 7. The linearity of the system is guaranteed along the ADC input range. Additionally, this graph shows the capability of LabOSat when recording data. If an ADC input is out of range, the channel is saturated. Then, LabOSat discards the measurements taken by that channel and changes to the next not-saturated channel to continue to sample the signal without saturation avoiding losing changes of the device response.

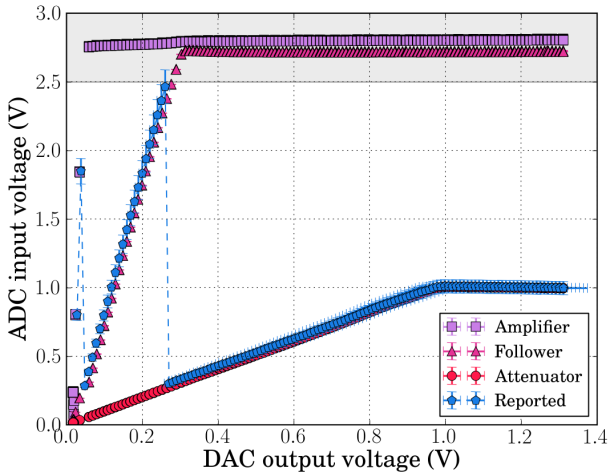


Fig. 7: System linearity when sourcing current. Measurement performed over a 1 kΩ resistance while current source is being controlled by DAC output. Multiple ADC inputs were acquired: attenuated, buffered (follower) and amplified. Shaded area indicates out of range ADC channels, thus not sensed.

This measurements allowed to find DAC offset voltage, DAC gain error and the ADC gain error for the microcontroller. These values were found to be lower than the maximum reported by the manufacturer and are being used for software correction in data processing.

Working with DUTs

As shown at the inset of Figure 8 the memory cell exhibits the typical behaviour of a Resistive Switching device. The non-smooth electrical hysteresis indicates that commutation is achieved [7], [8].

During the SET process, as the device is being controlled by current, the transition from HRS to LRS is power regulated meaning that $P = I^2R$ will be diminished if the remnant resistance gets lower.

The 12-bit DAC of the microcontroller and the ability of LabOSat-01 to precisely control current and voltage over the same device allow us to observe the dynamic behaviour of a typical commutation. Within set process 349 pulses are applied and 447 for reset process. The *number of steps per record* for set process is set to 5 and 7 for reset, this gives a record of 63 data points for set and 69 for reset. With this in mind the curve acquired with the platform has 132 points. This number was chosen to equitably distribute data storage along 18 devices, 13 ReRAM devices and 5 xFETs.

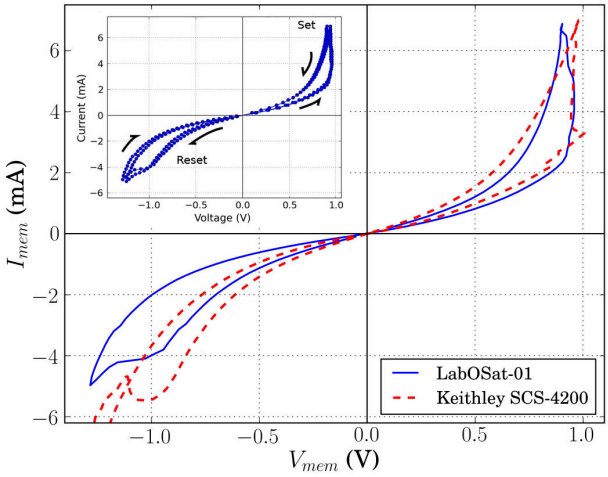


Fig. 8: Dynamical response obtained with different instruments over the same TiO_{2-x}-based ReRAM device. Inset: IV curves of the device after two consecutive Standard Tests.

Comparative Analysis

To illustrate the performance of LabOSat-01 as a measurement and characterization instrument, same I-V curves were performed on a Keithley SCS-4200 over the same device. These experiments were performed using KITE software with two routines to perform a current controlled SET sweep followed by a voltage controlled RESET sweep. Two SMUs were use to stress the sample. The programmed stimuli involved 142 steps for set and 302 for reset. The software allowed us to record the responses in all steps of the sweeps. Comparison can be seen in Figure 8.

Same behaviour is observed in dynamic response with either instrument. The shift between curves raises from fact that when devices are coupled into LabOSat-01 for testing, parameters of the IV test must be tuned to obtain the desired response as said in Section III-A. In contrast, measurements done with SMUs can be considered to be performed without any loads.

Experiments over transistors are shown in Figure 9 and 10. The plots concern several sweeps of V_{xFET} for different fixed gate voltages and sweeps on gate for different fixed V_{xFET} voltages. In the modulation curves, Figure 9, besides LabOSat forces the sweeps on V_{xFET} with same voltage values, V_{DS} are not repeated between curves. As described in Section III, V_{xFET} is imposed and V_{D} is directly measured so, actually, V_{DS} is deduced from measured data as V_{D} - V_{S}. This effect is consequence of the presence of R_{D} which allows LabOSat to calculate current through drain in a simple and effective

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

way. In more sophisticated instruments, as the Keithley SCS-4200 used in this work, there is no effect due to loads because the measurement principle is different from the one employed by the platform. However, this effect does not disturb the observation of the transistor behaviour.

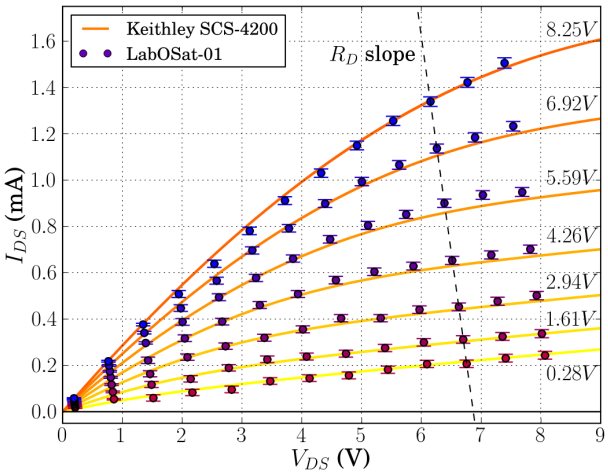


Fig. 9: xFET drain current vs. drain-to-source voltage measured with LabOSat-01 and Keithley SCS-4200 for several values of V_{GS} indicated next to solid curves. The line depict the effect of the load over the drain for same V_{xFET}.

Transfer curves of transistors, Figure 10, needs to be analized taking into account that R_D keeps $V_{xFET} \neq V_D$. When $V_{GS} \sim 0$, $V_{xFET} \sim V_D$ but increasing gate polarization leads to a bigger current flow through drain and so $V_{xFET} > V_D$ for $V_{GS} > 0$. This difference depends on the value of the load over the drain. At first sight, a small value seems to be the correct choice but the ADC's resolution sets a minimum value in order to clearly resolve drain currents. On the other hand, a large value would give a good dynamic range to work with drain currents but would produce high voltage drops. In this case, LabOSat, was designed to work with a medium load (500Ω) to obtain ratios $V_D/V_{xFET} > 0.85$ (15% variation from V_{xFET}). The shaded area in the plot indicates the points that differ in more than 10%, out of this area curves acquired with LabOSat and SCS-4200 overlap within the error bars.

V. Conclusions

First measurements with LabOSat-01 returned excellent results and exhibited to be in great agreement with the ones performed with more sophisticated and expensive equipment. They also showed that LabOSat is capable of forcing negative and positive currents (or voltage) on two-terminal devices. The ability to perform dynamic and static experiments and the flexibility it has when recording data make LabOSat a great instrument to study ReRAM devices.

In summary, LabOSat is a portable, configurable and low cost platform to perform voltage and current sweeps plus endurance test over electronic devices in harsh environment.

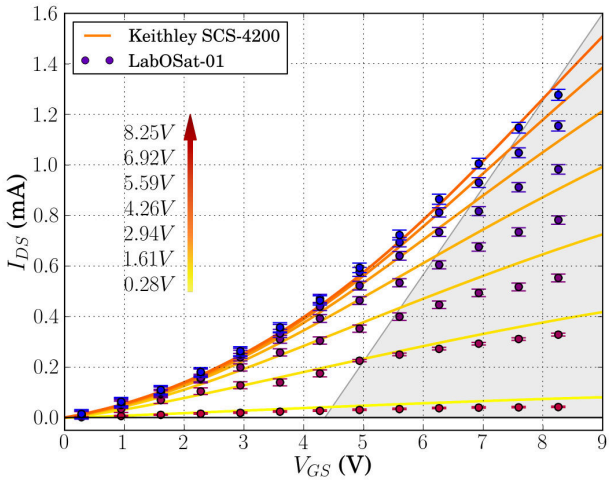


Fig. 10: xFET drain current vs. gate-to-source voltage measured with LabOSat-01 and Keithley SCS-4200 for several values of V_{DS} indicated next to the vertical color maped arrow. In case of LabOSat data, V_{DS} is not strictly fixed along same curve. Shaded area enclose the points were LabOSat sensed drain voltages 10% deviated from V_{xFET} due to load R_D.

Acknowledgment

The authors would like to thank to R. Ferreira, E. Paz, W. R. Acevedo and D. Rubí who participated in the fabrication of ReRAM devices; P. Stoliar, L. Hueso, E. Lopez and L. Patrone who were involved in fabrication of xFET devices; M. G. Inza and J. Lipovesky who facilitated the dosimeters. The authors acknowledge financial support from ANPCyT PICT 2013-0788 "MeMOSat" and UNSAM-ECyT FP-001.

References

[1] "Satellogic." [Online]. Available: http://www.satellogic.com/

[2] S. M. Guertin, M. Amrbar, and S. Vartanian, "Radiation test results for common cubesat microcontrollers and microprocessors," in *Radiation Effects Data Workshop (REDW), 2015 IEEE*, July 2015, pp. 1–9.

[3] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-Based Resistive Switching Memories Nanoionic Mechanisms, Prospects and Challenges," *Adv. Mater.*, vol. 21, no. 25-26, pp. 2632–2663, Jul. 2009.

[4] "Argentinian open industrial computer." [Online]. Available: http://www.proyecto-ciaa.com.ar/

[5] J. Lipovetzky, E. G. Redin, and A. Faigón, "Electrically erasable metalox-idesemiconductor dosimeters," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 1244–1250, Aug 2007.

[6] M. García-Inza, S. Carbonetto, J. Lipovetzky, M. J. Carra, L. S. Salomone, E. G. Redin, and A. Faigón, "Switched bias differential mosfet dosimeter," *IEEE Transactions on Nuclear Science*, vol. 61, no. 3, pp. 1407–1413, June 2014.

[7] N. Ghenzi, M. J. Sánchez, D. Rubí, M. J. Rozenberg, C. Urdaniz, M. Weissman, and P. Levy, "Tailoring conductive filaments by electroforming polarity in memristive based tio2 junctions," *Applied Physics Letters*, vol. 104, no. 18, 2014.

[8] D. Rubí, A. Kalstein, W. S. Román, N. Ghenzi, C. Quinteros, E. Mangano, P. Granell, F. Golmar, F. G. Marlasca, S. Suarez, G. Bernardi, C. Albornoz, A. G. Leyva, and P. Levy, "Manganite based memristors: Influence of the electroforming polarity on the electrical behavior and radiation hardness," *Thin Solid Films*, vol. 583, pp. 76 – 80, 2015.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Development of an UAV prototype for visual inspection of aerial electrical lines

Walter Benitez *, Yessica Bogado*, Ariel Guerrero† and Mario Arzamendia‡
* Research Center of Science, Technology and Advanced Innovation (CICTIA)
"Our Lady of the Assumption" Catholic University (UCA) - Hernandarias, Paraguay
Email: walter.benitez@ucap.edu.py, yessica.bogado@ucap.edu.py
†Innovation Center in Automation and Control (CIAC)
Technology Park of Itaipu (PTI) - Hernandarias, Paraguay
Email: ariel.guerrero@pti.org.py
‡ Laboratory of Distributed Systems
Faculty of Engineering, National University of Asuncion (FIUNA) - Asuncion, Paraguay
Email:marzamendia@ing.una.py

***Abstract*—The development of an attitude and altitude control system of an unmanned aerial vehicle for visual inspection of aerial electrical lines is presented. The system was implemented using quaternions algebra and the altimeter equation for the orientation estimation with the purpose of controlling the quadcopter. The proposed system was simulated in Matlab using a Newton-Euler model and compared with the performance on a real prototype operation, obtaining similar results in both cases.**

***Keywords*—aerial electrical lines, visual inspection, quadrotor, control theory, quaternions.**

I. Introduction

The electrical line inspections play a very important role in maintaining the quality of the energy delivered because of its relation with the predictive and preventive maintenance. It is clear that electrical companies should implement new methods and techniques that allow the routinely realization of such inspections avoiding as much as possible urgent maintenance tasks that require the interruption of the energy provision. The responsible institutions for the transmission and distribution of electrical energy use different inspection methods for controlling the state of the electrical lines and its accessories. The visual inspection consist of a meticulous task where physical damages are searched or possible difficulties that may reduce the efficiency of the electrical system. Diverse problems may be found when inspecting the lines, for example, at the insulating chains leakage lines may appear, insulating loss, breakage or cracking of the bells as well as loss of galvanized, rust, appearance of flakes or bubbles at the ironwork; even it is possible to see other inconveniences as damage by organic material, weather damage, the sliding of the shock absorber or invasion of the vegetation [1]. Visual inspection is an activity that accompanies the predictive maintenance with the objective of predicting failures and take preventive and/or corrective actions, keeping the conditions safe and preventing accidents [2]. There are four visual inspection techniques: foot patrol, climbing robots, manned aerial vehicles and unmanned aerial vehicles [3] [4]. The foot patrol is the most economical type and with the least risk because the lines are simply inspected from the ground with the support of binoculars. However the procedure is slow and the diagnosis is less precise. The manned aerial vehicles, specifically the helicopters, allow advantages as the speed and accessibility to irregular terrains, however the high costs and the danger exposures by the approximation of the operators with live lines do not allow their use commonly [3]. A more modern style is the use of climbing robots which offer a more precise inspection due to its proximity of the targets, but its complexity in the development for overcome obstacles, the effects of electromagnetic field over its sensors and electronics parts do not make them an economical solution. Currently the use of robotics platforms for observation is a growing interest field for many companies and institutions. A visual inspection using UAV (Unmanned Aerial Vehicle) are faster, more versatile and more economical [3] [4] [5]. These vehicles offer the capacity of stationary flight for a better image capture from different points around the electrical lines without putting the life of operators in risk. The UAVs or drones are vehicles that do not need a crew on them to control or conclude a specific task. It might be defined as robotic systems that land from a ground base and are controlled remotely by pilots, or autonomously following a mission previously programmed; or even both instances can be combined. The characteristics and costs vary according to the application, which it can be for environmental monitoring, search task, facilities supervision, surveillance, etc. The UAV may be catalogued according to different aspects, for example the flying capacity (autonomy), application or type of aircraft [6]. In the present work the UAV known as quadrotor is employed. The first step of this work is to study the mathematical modeling of the UAV which considers the aerodynamical effects of the vehicle. Currently, there are different variety of models [7] [8] [9] [10]. Defining an appropriate model plays an important role at the time of designing the control system and performing the simulation studies. The control assumes the availability of the variables of state, i.e., the estimation of the vehicle orientation. There

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

are different control algorithms due to the fact that the model of the quadrotor is multivariable and not lineal. Among the available bibliography of the topic, Schermuk [7] analyze the difference between the representations with Euler angles and quaternions, expose the advantages of the trajectory following around the main axis (eigenaxis) and finishes with a comparison between a PID (Proportional Integral Derivative) control and the LQR (Linear Quadratic Regulator). Bresciani [8] introduces with details a model based on the laws of Newton with the representation in Euler angles and its relationship with the DCM (Direction Cosine Matrix). On the other hand Fresk [9] proposed a control scheme based on quaternions, where an attitude model and a square non-lineal proportional control algorithm were implemented, both in the quaternions space without transformations and calculations with the Euler angles. In the present work the simulation, building and implementation of an UAV is done, applying a CAS (Control Augmented System) + SAS (Stability Augmented System) with quaternions algebra that later is verified by using it for a visual inspection of an aerial electrical line.

The rest of the paper is organized as follows, in Section II the definition of quaternions is presented and used to define a Newton-Euler model to describe the movement of the quadcopter. Section III presents the design of the control system (altituted and attitude) using the concepts of Section II. In Section IV, the quadcopter prototype built is described; Section V presents the results of simulating the control system in Matlab and Section VI the measurements of the performance of the control system during operation of the prototype. Finally, Section VI presents the conclusion and future work to be done.

II. Theoretical Framework

The theoretical framework is divided into two parts: the dynamic model that is related to the physical behavior of the aircraft and the representation of the attitude of it in space, which was made with Euler angles because they are much more intuitive for interpreting the equations in the simulation; and quaternions which is another way to represent the attitude of the aircraft that was used to implement control over it to avoid the problem of gimbal lock produced by the Euler angles [7].

A. Quaternions

The quaternions is a four dimensional hypercomplex number that can be used to represent an orientation of a body frame in 3D space. A quaternion Q have the following form $q_0 + q_1 i + q_2 j + q_3 k$, where q_0, q_1, q_2, q_3 are real numbers called components of the quaternion and i, j, k are imaginary units.

For rotation the quaternion has the following form [11]

$$Q = (\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}\hat{r}) \tag{1}$$

Where: $q_0 = \cos(\frac{\alpha}{2})$, $q_1 = \sin(\frac{\alpha}{2})\, r_x$, $q_2 = \sin(\frac{\alpha}{2})\, r_y$, $q_3 = \sin(\frac{\alpha}{2})\, r_z$

The angle α represents the magnitude to be rotated around the main axis r, defined as eigenaxis in Euler's rotation theorem. Just as the property of orthogonal matrices rotation [6], quaternion rotation obey the following [7]:

$$Q_C^A = Q_B^A . Q_C^B \tag{2}$$

$$(Q_B^A)^* = (Q_B^A)^{-1} = Q_A^B \tag{3}$$

Were $(Q_B^A)^*$ is the conjugated quaternion of Q_B^A

Also being $\hat{r} \in \mathbb{R}^{3\times1}$ the rotation axis of the quaternion Q and α the rotation angle, the following is allowed [7]:

$$\ln Q = \frac{\hat{r}\alpha}{2} \tag{4}$$

Considering the quaternion rotation it can be defined a rotation matrix with the components of the quaternion [12]:

$$\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_3q_0) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_3q_0) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_1q_0) \\ 2(q_1q_3-q_2q_0) & 2(q_0q_1-q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

By combining the property of quaternions with the sequence of rotation 3,2,1 established in Euler angles it can be stated that

$$Q_{Euler} = Q_\psi Q_\theta Q_\phi = \begin{bmatrix} \cos\frac{\psi}{2} \\ 0 \\ 0 \\ \sin\frac{\psi}{2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} \\ 0 \\ \sin\frac{\theta}{2} \\ 0 \end{bmatrix} \begin{bmatrix} \cos\frac{\phi}{2} \\ \sin\frac{\phi}{2} \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

Equation (6) presents the conversion of quaternion to Euler angles [12]:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{2(q_0q_1+q_3q_2)}{1-2(q_1^2+q_2^2)}\right) \\ \arcsin\left(2(q_0q_2+q_3q_1)\right) \\ \arctan\left(\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\right) \end{bmatrix} \tag{6}$$

B. Dynamic Model

According to Bresciani [8] there are four types of basic movements for a quadcopter: Sustentation (U_1), Roll (U_2), Pitch (U_3) and Yaw (U_4). These movements are represented in equations as the forces that drive themselves to implement these actions. Considering these four movements, the system dynamics and the direction cosine matrix we have a model called Newton - Euler and it is described by these equations:

$$\begin{aligned} \ddot{x} &= \frac{U_1}{m}(\sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\theta)) \\ \ddot{y} &= \frac{U_1}{m}(-\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\theta)) \\ \ddot{z} &= \frac{U_1}{m}\cos(\theta)\cos(\phi) - g \\ \dot{p} &= \frac{I_{YY}-I_{ZZ}}{I_{XX}}qr - \frac{I_{TA}}{I_{XX}}q\Omega_t + \frac{U_2}{I_{XX}} \\ \dot{q} &= \frac{I_{ZZ}-I_{XX}}{I_{YY}}pr - \frac{I_{TA}}{I_{YY}}p\Omega_t + \frac{U_3}{I_{XX}} \\ \dot{r} &= \frac{I_{XX}-I_{YY}}{I_{ZZ}}pq + \frac{U_4}{I_{ZZ}} \end{aligned} \tag{7}$$

As x, y, z represent the position in space, $\omega = [p, q, r]$ the angular velocity according to the vehicle body frame,

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

$I_{XX}, I_{YY}, I_{ZZ}, I_{TA}$ inertial moments of the vehicle and U_1, U_2, U_3, U_4 are the forces and moments described above. Then these forces are related to the rotation speed of the propellers by the following equations.

$$\begin{aligned} U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 &= bl(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_3 &= bl(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \\ U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \Omega_t &= -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{aligned} \tag{8}$$

Where $\Omega_n (n = 1, 2, 3, 4)$ represents the rotation of each propeller and its unit is rad/sec, Ω_t is the sum of all speed rotations, b is the thrust factor, d is the drag factor and l is the distance between the center of the motor to the center of quadcopter body.

III. Design of the Control System

Generally aircraft control systems can be classified into a SAS, CAS and an autopilot system. The stability augmentation system (SAS) aims to stabilize the vehicle controlling the dynamics (angular speed) while the control augmentation system (CAS) aims to maintain a desired orientation [7].

In order to make a simple control design and linearly independent on each degree of freedom the Newton - Euler model (7) is simplified as follows.

$$\begin{aligned} \ddot{x} &= \frac{U_1}{m}(\sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\theta)) \\ \ddot{y} &= \frac{U_1}{m}(-\cos(\psi)\sin(\phi) + \sin(\psi)\sin(\theta)\cos(\theta)) \\ \ddot{z} &= \frac{U_1}{m}\cos(\theta)\cos(\phi) - g \\ \dot{p} &= \frac{U_2}{I_{XX}} \\ \dot{q} &= \frac{U_3}{I_{XX}} \\ \dot{r} &= \frac{U_4}{I_{ZZ}} \end{aligned} \tag{9}$$

And to use the four basic movements as input signals an open loop control is applied with the inverse of (8).

$$\begin{aligned} \Omega_1^2 &= \frac{U_1}{4b} - \frac{U_2}{4bl} - \frac{U_3}{4bl} - \frac{U_4}{4d} \\ \Omega_2^2 &= \frac{U_1}{4b} - \frac{U_2}{4bl} + \frac{U_3}{4bl} + \frac{U_4}{4d} \\ \Omega_3^2 &= \frac{U_1}{4b} + \frac{U_2}{4bl} + \frac{U_3}{4bl} - \frac{U_4}{4d} \\ \Omega_4^2 &= \frac{U_1}{4b} + \frac{U_2}{4bl} - \frac{U_3}{4bl} + \frac{U_4}{4d} \end{aligned} \tag{10}$$

A. Attitude Control

For attitude control we proceeded to choose a CAS + SAS dual control (Fig. 1) because of its wide use in the field of UAVs [10]. For CAS a proportional approach was used and for SAS a PIDT1 (Proportional, Integral, Derivative with a low-pass filter) [13].

The error is obtained through the rotation needed to go from current quaternion (Q_a) to desired quaternion (Q_{des}), such

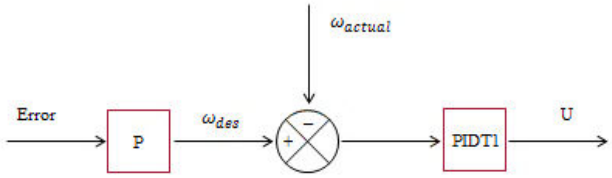


Fig. 1. Block diagram of CAS+SAS control

rotation obtained through (3) is represented by the quaternion error as follows:

$$Q_{error} = Q_a^*.Q_{des} \tag{11}$$

Where Q_a^* is the conjugated of Q_a.

Then from (4):

$$2\ln Q_{error} = 2\ln(Q_a^*.Q_{des}) = \hat{r}\alpha \tag{12}$$

As the components of $\hat{r}\alpha$ vector represents the error in each main axis [7] [14] [9].

$$E_x = r_x\alpha \tag{13}$$

$$E_y = r_y\alpha \tag{14}$$

From this the error for X and Y axis is obtained, however there is a problem with the Z axis because the DMP (Digital Motion Processor), which is in charge of fusing the data of the IMU (Inertial Motion Unit, MPU6050), does not fuse the data taken from the magnetometer. A solution to this is given by using ψ_{false}, calculated with the formula obtained in (6) with Q_a.

$$\psi_{false} = \arctan\left(\frac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)}\right) \tag{15}$$

The desired quaternion (Q_{des}) is obtained using (5) with ψ_{false} and desired rotation angles in the X and Y axis.

With this it only remains to consider the error of the Z axis and for that a complementary filter is applied on the orientation obtained through magnetometer (17) [15] and gyroscope (rdt) as showed on (16). After that the error is calculated with ψ_{des} (18).

$$\psi_{real} = \rho(\psi_{real} + rdt) + \psi_{mag}(1 - \rho) \tag{16}$$

$$\psi_{mag} = \arctan\left(\frac{m_x}{m_y}\right) \tag{17}$$

$$E_z = \psi_{des} - \psi_{real} \tag{18}$$

Where m_x and m_y are the values obtained with the magnetometer of the magnetic field of the earth according to the X and Y axis and have to be calculated correctly and rotated to the inertial frame with the current quaternion. And ρ>0 is a constant that indicates the contribution of the gyroscope to ψ_{real} and it has a maximum value of 1.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

B. Altitude Control

The height control is highly related to the "hovering", i.e. keeping the altitude in a certain height and holding the attitude without motion in a horizontal position relative to the ground [10]. As in attitude control, to perform the control action we proceeded to work with the same strategy of dual control, a CAS to control the position of the quadcopter according to a reference value and SAS to vary the quadcopter thrust according to a desired vertical speed [16] as seen in Fig. 2.

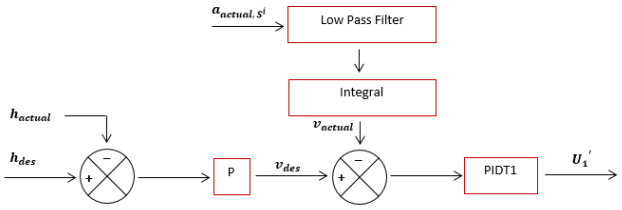


Fig. 2. Block diagram of altitude control

The estimation of values is necessary, the current height is obtained through barometer with the altimeter equation (21) [17], and then filtered with a low pass filter that is related to the speed v_{actual} ((22) and (6)). To obtain the speed first accelerometer data must be passed to the inertial frame, subtracting the values of gravity so as to be able to measure the acceleration without the influence of gravity as it shows in (20). Then these data is filtered with LPF (Low Pass Filter) of 50 Hz and proceeds to integration.

$$U_1 = mg + U_1' \tag{19}$$

$$a_{actual,S^i} = (a_{actual} - g_z)_{v3 \to S^i} \tag{20}$$

$$z = \frac{T_0}{\Gamma}\left[1 - \left(\frac{P}{P_0}\right)^{\frac{R_{spe}.\Gamma}{g}}\right] \tag{21}$$

$$KP_1 = p(1 - e^{C_1|v_{actual}|}) \tag{22}$$

$$\begin{aligned} h_{actual_n} &= (z - z_0)(KP_1 + 1 - p) \\ &+ (h_{actual_{n-1}})(p - KP_1) \end{aligned} \tag{23}$$

Where p is a value ranging from 0 to 1 that regulates the dependency of the current reading in relation to a previous one, C_1 is a negative value that regulate the dependency of the filter to the vertical speed of the aircraft and $v3 \to S^i$ represents the rotation from the body frame to inertial frame [6]. This filter is made to counteract the variation produced by the barometer when the aircraft hovers.

For the altimeter equation we have that R_{spec} is the specific constant of gas, P_0 is the pressure at sea level, P is the actual pressure, T_0 is the temperature at sea level and Γ is the slop for the uniform decrease of temperature relative to height. The reference height for quadcopter is z_0.

C. General Control Scheme

Once the sectors of the control system were developed they are connected to each other according to the Fig. 3. In the figure can be seen that is possible to choose a type of control either manually or automatically. If the manual control is selected, the user manipulates the force values and the desired tilt angles; on the other hand, the automatic control maintains the height or position by GPS and barometer. After this, it is passed to the SAS + CAS control and the four basic movements (U_1, U_2, U_3, U_4) are obtained and used in (10) to get the desired rotation. Getting with it the PWM needed to regulate the ESC to finally feedback all the control blocks through the sensors.

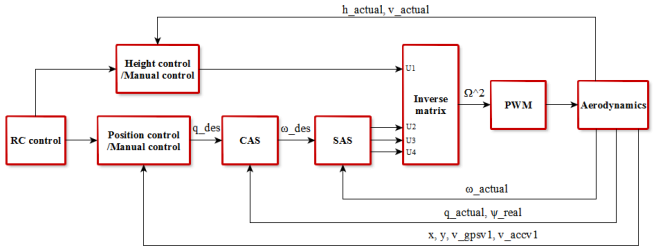


Fig. 3. General Control Scheme

IV. Hardware Description

The prototype is a quadcopter Fig. 4, the motors used are BLDC (Brushless Direct Current) type of 935 KV, which are three-phase motors that feed on a DC continuous source, the advantage of these engines is that they are lighter, more dynamic and more response efficient; the propellers are 10x4.5 inches. An ESC (Electonic Speed Control) 30 A was used for controlling the BLDC motors., this driver is also responsible for supplying power to the electronic components of UAVs. The main power source is a rechargeable LiPo (Lithium Polymer) 3S 5000 mAh 25C, which is a lightweight battery with a reduced volume. An eight channel radio control with PPM (Pulse Position Modulation) was used to operate the quadcopter, and covers distances up to 1.5 km and operates at a frequency of 2.4 GHz. The main element of the vehicle is the flight controller Crius AIOP (All in One Pro) v2.1, this electronic board has built-in various sensors like a gyroscope / accelerometer MPU6050 6-axis high precision altimeter MS5611-01BA01 and HMC5883L 3-axis magnetometer. The integrated microcontroller ATMEGA 2560 is an 8-bit, 16 MHz and communicates with external devices via the pins and serial ports. Two wireless modules (2.4 GHz) were used to transmit data from the flight controller to the ground station. In order to display in a computer the quadcopter objective a transmitter audio and video receptor of 500mW power operating at a frequency of 5.8 GHz was chosen and brings a CCD camera(Charge Coupled Device) with it. The power supply of the transmitter and receiver is a 1300 mAh LiPo battery for each. It also had incorporated a GPS module (Global Positioning System) to know the quadcopter's position. The total budget of this prototype was about 700 U$ FCA (Free Carrier).

V. Simulation and Implementation of Control

Simulations were performed using Matlab(Version R2013b) with the previously described model [8] [18], the adopted parameters were as follows:

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Fig. 4. Prototype

TABLE I
PARAMETERS OF THE PHYSICAL MODEL

Name	Symbol	Value
Quadcopter's Mass	M	$1.48\ kg$
Thrust coefficient	b	$12.76 \times 10^{-6}\ N.s^2/rad^2$
Drag coefficient	d	$1.07 \times 10^{-6}\ N.m.s^2/rad^2$
Quadcopter's radius	l	$0.23\ m$
Inertia relative to the X axis	I_{XX}	$9.86 \times 10^{-3}\ kg.m^2$
Inertia relative to the Y axis	I_{YY}	$9.86 \times 10^{-3}\ kg.m^2$
Inertia relative to the Z axis	I_{ZZ}	$16.64 \times 10^{-3}\ kg.m^2$
Total inertia of the actuators	I_{TA}	$74.12 \times 10^{-6}\ kg.m^2$

TABLE II
CONTROL PARAMETERS

CAS			
	Type	**Symbol**	**Value**
Attitude	Proportional	KP_ϕ	3
		KP_θ	3
		KP_ψ	2
Altitude	Proportional	KP_h	1
SAS			
	Type	**Symbol**	**Value**
Attitude	Proportional	KP_ϕ	0.75
		KP_θ	0.75
		KP_ψ	0.75
	Integral	KI_ϕ	0.35
		KI_θ	0.35
		KI_ψ	0.21
	Derivative	KD_ϕ	0.01
		KD_θ	0.01
		KD_ψ	0.01
Altitude	Proportional	KP_h	2.9
	Integral	KI_h	0.3
	Derivative	KD_h	0.0015

Two simulations were executed, one assuming that the sensors are ideal and another with disturbance. In attitude control, a reference value of 20° was choose and when sensors are ideal results show an almost nonexistent peak and a rise time of about 0.89 seconds as shows in Fig. 5. On the other hand when input a Gaussian noise of 5% of the value taken by the sensors the results shows a standard deviation of 0.43 degrees in the roll, 0.49 degrees on yaw and 0.54 degrees in the pitch, the values of overshoot and time rise are held as in Fig. 6.

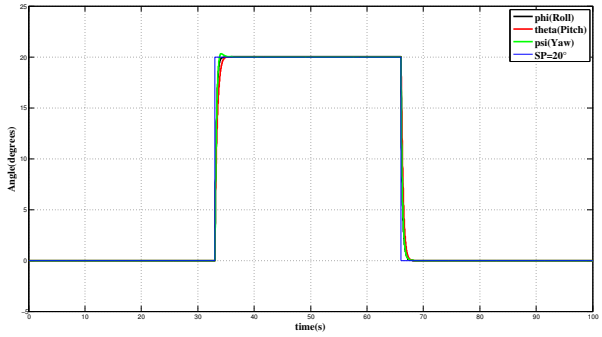


Fig. 5. Step response of the Yaw, Pitch and Roll angles without noise

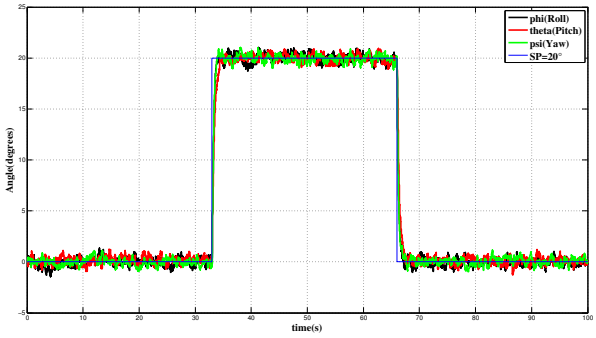


Fig. 6. Step response of the Yaw, Pitch and Roll angles with noise

At the height control reference value of 2 meters was used, for a simulation without disturbing the rise time is 2.5 seconds and with an overshoot less than 10% as it can be seen in Fig. 7. The simulation with Gaussian noise of 10% of the real value produced a standard deviation of 0.022 meters showing in Fig. 8.

VI. EXPERIMENTAL RESULTS

The Table III shows the contrast of standard deviations relative to a reference value between flight simulations performed previously and real stationary flights with the prototype.

The values of both, roll and pitch, are similar to the simulation results and the value of yaw differs mainly because of electromagnetic disturbance that motors produce in the magnetometer. As for height, pressure variations produced a standard deviation of 0.17 meters in hovering. It is necessary to note that rates estimated with the accelerometer increases the amount of error in the controller.

After contrast, the inspection of a medium voltage transformer in an overhead power line was carried out (Fig. 9). Quadcopter position was about 7 meters above ground and about 5 meters from the transformer and no undesirable effects

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

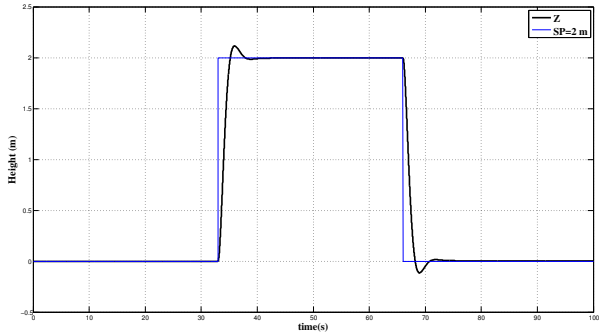


Fig. 7. Step response of altitude control without noise

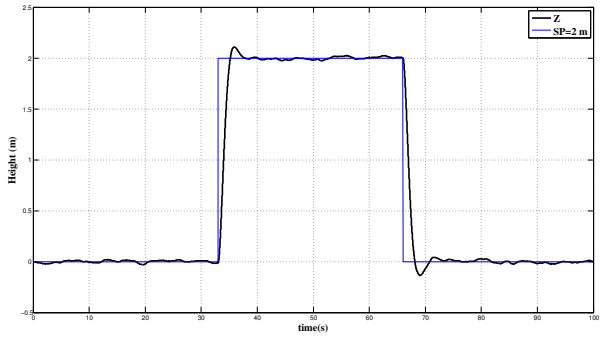


Fig. 8. Step response of altitude control with noise

Fig. 9. Visual inspection with the prototype seen from the ground

TABLE III
CONTRAST TABLE

	Simulated Test	Real Test
Roll (Degrees)	0.43	0.45
Pitch (Degrees)	0.54	0.41
Yaw (Degrees)	0.49	1.59
Height (meters)	0.02	0.17

were detected due to electromagnetic fields. The images were captured with the camera described before in section IV and transmitted from quadcopter to a computer on the ground.

VII. CONCLUSION

This paper presented a CAS + SAS (P + PIDT1) control system with the use of quaternion for UAV. The simulation of this system showed satisfactory results when comparing with the work of Bresciani, Kharsansky and Schermuk [8] [10] [7]. Finally, the prototype was used to perform the visual inspection of a medium voltage transformer in a power distribution line.

As future work, the following topics are suggested:

- Using extended Kalman filter with a more powerful microcontroller and to make a better estimate of the velocities and positions
- Implementing a tracking system for automated inspection
- Performing the contrast with other types of attitude and altitude control for UAVs
- When conducting inspections on transmission towers take into account the electromagnetic effect produced by high voltages and properly protect electronics of the UAV
- Studying and implementing the use of a thermal camera to identify hot spots

REFERENCES

[1] J. Boza, "Inspección integral de las líneas de transmisión," *Ingeniería Energética*, vol. 24, no. 3, pp. 3–7, 2003.
[2] DISPAC, "Manual de mantenimiento para redes eléctricas alta media y baja tensión," Quibd, 2015.
[3] J. Garcés, "Inspecciones aéreas de líneas de transmisión con alta tecnología," in *Jornadas Técnicas ISA*, 2012.
[4] J. León and M. C., "Estudio de soluciones existentes en el mercado para la inspección y manteniemiento de líneas eléctricas de alta tensión." Universidad Politécnica de Madrid, 2013.
[5] R. Gouws, "Prototype monitoring system for power line inspection by means of a pandaboard," *Journal of Energy and Power Engineering*, vol. 8, pp. 176–182, 2013.
[6] R. Austin, *Unmanned aircraft systems*. Chichester: John Wiley and Sons Ltd, 2010.
[7] D. Schermuk, "Diseño e implementación de un controlador para la orientación de un quadrotor," Facultad de Ingeniería de la Universidad de Buenos Aires, 2012.
[8] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," Master's thesis, Lund University, 2008.
[9] E. Fresk, "Full quaternion based attitude control for a quadrotor," in *European Control Conference (ECC)*, Zrich, 2013.
[10] A. Kharsansky, "Diseño e implementación de un sistema embebido de control de actitud para aeronaves no tripuladas," Facultad de Ingeniería de la Universidad de Buenos Aires, 2013.
[11] G. Torres del Castillo, "La representación de rotaciones mediante cuaterniones," *Miscelanea Matemtica*, pp. 43–50, 1999.
[12] M. Henderson, "Euler angles, quaternions, and transformation matrices," NASA, Houston, Texas, Tech. Rep., 1977.
[13] I. V. Airde. (2012) Attitude control structure, changing from pi to pid? diydrones.com. [Online]. Available: www.diydrones.com/forum/topics/attitude-control-structure-changing-from-pi-to-pid
[14] B. Michini, "Modeling and adaptive control of indoor unmanned aerial vehicles," Massachusetts Institute of Technology, 2009.
[15] Honeywell. (1995) Compass heading using magnetometers. [Online]. Available: www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/Magnetic__Literature_Application_notes-documents/AN203_Compass_Heading_Using_Magnetometers.pdf
[16] A. Copter. (2015) Altitude hold mode. [Online]. Available: copter.ardupilot.com/wiki/altholdmode
[17] D. of Mathematical Physics of the University College Dublin. (2005) The hydrostatic equation. [Online]. Available: maths.ucd.ie/met/msc/fezzik/Phys-Met/Ch03-Slides-2.pdf
[18] M. Ai-Omari, "Integrated simulation platform for indoor quadrotor applications," in *Mechatronics and its Applications (ISMA)*, Amman, 2013.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Enhancing deterministic in-vehicle networks with a Traffic Management Module

Christian Carrizo
Escuela de Informática y Telecomunicaciones
Facultad de Ingeniería
Universidad Diego Portales
Email: christian.carrizo@mail.udp.cl

Diego Dujovne
Escuela de Informática y Telecomunicaciones
Facultad de Ingeniería
Universidad Diego Portales
Email: diego.dujovne@mail.udp.cl

***Abstract*—Deterministic communication is currently a fundamental requirement for many real-time systems, such as drive-by-wire and autonomous driving solutions; however, in the last few years there has been a sustained increase in the number of other type of on-board connected devices such as actuators, rear vision cameras and multimedia services using different networks on the same vehicle, thus adding weight and complexity. The most efficient solution to this problem is to build a single in-vehicle network to support the delay and bandwidth constraints generated by each type of device. The main goal of this paper is to describe and analyse the performance of a deterministic network management module created to reduce jitter and packet loss, given variable traffic conditions. The most relevant parameters are obtained by simulation work, which permits to establish the actions and policies to execute the prototype modules for this kind of networks.**

I. Introduction

Current in-vehicle networks must comply with very strict constraints for applications such as breaking and steering; the most relevant of them being limited delay, extremely low packet loss and very narrow jitter margin. If any of these constraints is not met, the main consequences include economic losses, system failure and, in the worst case life loss.

Unlike standard general-purpose data networks such as Ethernet and 802.11 which are based on distributed multiple access schemes, in-vehicle networks constraints require a time-division multiple access mechanism to guarantee the required bandwidth and a predictable delay. As a consequence, designs are application-dependant as it is established by Gupta et al. [1]. However, recent research work has defined a number of strategies to implement distributed traffic control to provide a general network design and configuration framework for deterministic networks. The proposed architecture is composed by a high speed TDMA (Time Division Multiple Access) network where controllers, sensors and actuators are connected. The different devices are then connected through logical subnetworks.

Industrial and automotive type networks define at least two type of traffic classes: regular and critical traffic. The first one responds to predictable periodic packets, mostly generated by instrumentation, while the second belongs to the events such as breaking and steering, which is aperiodic and unpredictable. In terms of priority, the latter has the highest one, it cannot wait for an available slot to be transmitted, but the current slot must be used in its' place to guarantee timely delivery of critical traffic [2].

Although several models have been proposed for in-vehicle networks, and current standards to enable IPv6 on deterministic networks are under development, there is a lack of simulation use cases for in-vehicle networks based on real requirements to provide a reference con-

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

struction to evaluate communications protocol performance and limitations. Furthermore, the inception of a Traffic Management Module for in-vehicle networks reduces the delay and jitter values for critical traffic, compared against the reference case.

Our contribution in this paper is twofold. First, we develop a simulation use case for OMNET++ taking into consideration the most relevant network elements present in an in-vehicle network with their inherent characteristics, and second, we present the design, implementation and validation of a Management Module to improve the results, taking into account all the previously characterized requirements of the convergent traffic. To the best of our knowledge, this is first time a reference simulation environment for a mainstream simulator and a traffic management module are designed, implemented and validated for in-vehicle realtime networks.

The rest of the paper is distributed as follows: Section II Describes the State of the Art on the area; Section III describes the proposed management module; Section IV shows the simulation scenario and configuration; Section V describes the simulation results and finally Section VI concludes this work.

II. State of the Art

A. In-vehicle networks

In the beginning, the number of electrical and electronic devices available in vehicles such as cars, trucks and buses was small. The battery, the starting motor, the lights, a few fuses and switches were the main components. The continuous evolution of hardware and software has enabled auto makers to include new functionalities to improve security and driving experience, like information and entertainment services, traffic information and navigation systems, security and stability controls and advanced driver assistance systems (ADAS). Nowadays, most of the functionalities available in vehicles are controlled by the Electronic Control Units (ECU), which can exchange up to 6000 different signals, such as vehicle speed throughout in-vehicle networks [3]. Because of this, a number of specific bus technologies were developed with the aim to satisfy the different reliability needs, response time, flexibility and application bandwidth. Figure 1 shows an example of the different types of devices present in this type of networks, and the interconnection technologies.

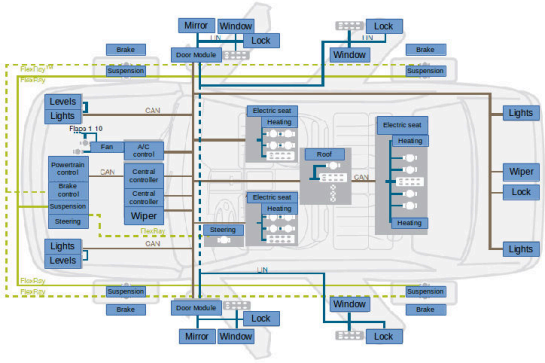


Fig. 1. Typical in-vehicle network. [4].

B. Current network technologies

- **Controller Area Network (CAN)** CAN [5] uses a bus to send and receive control messages in realtime at a bit rate of up to 1Mbps. The message length is variable between 1 and 8 bytes and uses its' own ID, which is unique within the bus. CAN supports full realtime communication. When a collision happens on the bus, the highest priority message takes control of the bus immediately.
- **Local Interconnect Network (LIN)** LIN [6] communications bus uses broadcast transmission in master-slave mode. It was developed by the LIN consortium as a less expensive network alternative to CAN for non-critical in-vehicle devices.
- **FlexRay** FlexRay [5] was developed to solve CAN's limited bandwidth. It enables data rates up to 10Mbps and it is compatible with event or time triggered communications. Multiple access is implemented using TDMA for realtime messages, while non-critical messages become time-triggered. This strategy improves bus usage efficiency while complying with realtime constraints.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

- **Media Oriented System Transport (MOST)** MOST is a communications standard for multimedia and entertainment services for the automotive industry. This technology was designed to provide efficient delivery of audio, video, data and control information between the on-board devices. It provides maximum bandwidth of 150Mbps using fiber or copper with a higher cost than CAN and FlexRay.

C. In-vehicle device domains

The different kinds of devices available in vehicles are classified into "domains" [7].

- **Powertrain** Is the group of components which provide energy to the vehicle. This includes the engine, axes, wheels, as components, plus several sensor devices to measure flow, pressure, speed, volume and stability. The controller must define the parameter sampling frequency, which in this case is very high, given the critical function they are in charge of.
- **Chassis** The powertrain plus the driving components such as breaks, steering system and suspension are attached to a structure called chassis. It has the same time restrictions as the powertrain: maximum controlled latency.
- **Bodywork and Comfort** This domain includes elements such as heating and air conditioning, window and seat control, lights and door locks. These sensors and controllers need a small amount of bandwidth and they can withstand high latency in the order of miliseconds.
- **Driver assistance** This domain includes those systems designed to help the driver in the driving task, plus complimentary systems to increase security of the passengers and pedestrians. Examples of these components are GPS, cruise control and automatic parking.

The interconnection of all the in-vehicle devices is becoming more complex and expensive because of the number of different networks implemented for different needs and following different standards. With the convergence to a single standard, all the communications between components can coexist in the same network, which would enable a scalable, reliable and unified network, thus reducing weight, cost, and, as a consequence, reducing fuel consumption.

TABLE I. Delay, bandwidth and reliability requirements for in-vehicle domains [3] [7]

Domain	End-to-end delay	Bandwidth	Reliability
Powertrain	$<10\ \mu s$	Low	High
Chassis	$<10\ \mu s$	Low	High
Bodywork and Comfort	$<10\ ms$	Low	Low
Driver assistance	$<250\ \mu s$ o $<1\ ms$ Depending on the system	20 - 100 Mbps per camera	Medium

D. Deterministic Ethernet

The current technology that best fits the former constraints is Ethernet, because of the associated low cost, high bandwidth and the wide range of protocols already available. Although specific technologies such as CAN and MOST originally covered their original requirements, their current bandwidth and delay limitations are bringing them to their design limits. However, standard Ethernet was not designed for realtime nor deterministic data transport, which are fundamental for in-vehicle communications; this required a specific standard to enhance the original Ethernet specification, to guarantee low latency and limited jitter, resulting in the IEEE 802.1 AVB and TTEthernet (Time-Triggered Ethernet) proposals. Due to space constraints, in this work, we concentrate on the TTEthernet specification.

TTEthernet implements traffic configuration and medium access policies for realtime communications over switched Ethernet networks. TTEthernet has a similar realtime medium access strategy as IEEE 802.1Qbv and PROFINET. These standards are based on the use of a preconfigured schedule with dedicated timeslots for realtime transmission, thus adding determinism and predictable delays. Even though FlexRay was designed for

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

realtime traffic, TTEthernet integrates realtime with best-effort traffic. A comprehensive comparison between FlexRay and TTEthernet can be found on the works of Steinbach et al. [8] and Zeng et al.[9]. Furthermore, TTEthernet can be seen as an extension of realtime protocols such as RT Ethernet, as proposed by MÃijller et al. [11].

III. Management Module

According to Finn et al. [13], the applications where critical traffic is critical, must include methods to control flow transmission of those packets in a hop-by-hop basis and allocate resources on each of the interconnection devices to have absolute guarantees for minimal and maximal latency. In this section we specify the parameters to design the prototype of proposed management module. We have surveyed the available literature in detail for in-vehicle network management modules, and we found that the closest approach was originated in 1994 with the CAN scheduler, a matter which was later studied and revised by Davis et al. [**?**].

The main disadvantage of a preconfigured scheduler, such as the one proposed on TTEthernet, is the inability to respond to packet reception variability. The main consequence is the artificial delay generated when a packet arrives to the transmitter and the corresponding slot has already passed. This means that the packet must wait until the next allocated slot is available. One possible solution is to establish longer timeslots to increase the chance of a packet of arriving to the corresponding slot, however, this technique would increase latency. In the quest for a new strategy our proposal provides resource allocation flexibility so as to add support for asynchronous traffic.

Our prototype is composed of two main elements. The first one is a modification of the TTEthernet switches in order to add new functionalities to the traffic traversing them and the corresponding slot allocation. The second is a general manager connected to all the switches which enables the distribution of new configuration or slot allocation messages.

Figure 2 shows the conceptual model of the system, which presents a simplified schema

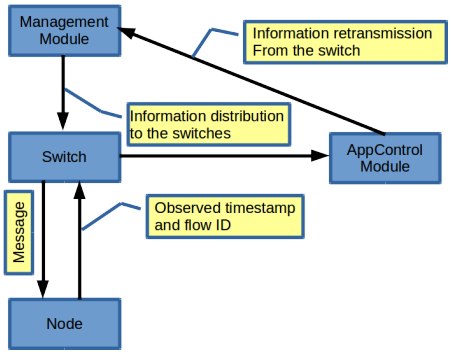


Fig. 2. System model: Management module interaction map

of the interaction between the two aforementioned elements and the rest of the network. The appControl module is part of the Switch module.

Our proposed mechanism increases the switch functionalities as follows: The original switch function only redirects packets to the correct port; the modified version enables the switch to dynamically adapt each flow by changing the receiver window size, duration and transmission window to minimize the packet delay due to queueing time to reduce end-to-end latency. In order to do this, the management module has access to the MAC layer parameters, while the PHY layer is fixed to 10 Gbps Ethernet. We fix the window duration in 12 clock ticks, equivalent to 1.2 μs. Using the same technique, once the type of flow is identified, slot selection is modified in order to comply with the requested maximum delay. The reserved slot duration is 1.2 μs. Finally, the "observed time" field is added to the message so as to enable the global manager to calculate the slot time on the other switches the message will traverse.

Finally, the global manager interconnects all the switches on the network to receive the packets sent by the appControl module, identify the source switch and then distribute the information among the other switches. Since the traffic flow is not symmetric on the network, the "observed time" must be compensated.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

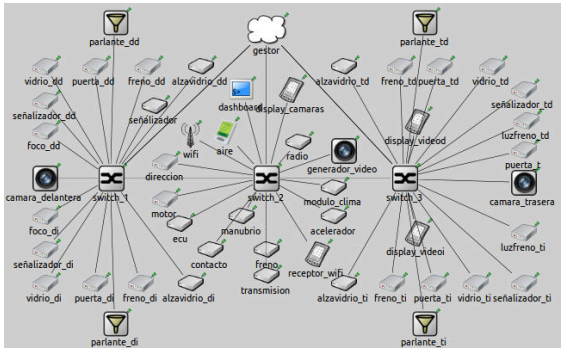


Fig. 3. Network topology.

IV. Simulation

Figure 3 shows the network topology we selected as the simulation scenario. We have included representative elements for each of the domains described on Section II-C. We have used OMNET++ Simulator with the CoRE4INET extension, which implements the TTEthernet specification.

Taking into account the proposed scenario, we have decided to include 3 types of messages to better represent this type of networks: time-triggered (TT), best effort (BE) and audio video bridging (AVB).

We have used TT traffic class to characterize all the critical and signal traffic inside the vehicle as realtime traffic. For on-board WiFi, the class type is BE, giving low priority to traffic such as navigation and instant messaging. Multimedia traffic uses B class and front and rear cameras use class A, both defined on the IEEE 802.1 AVB standard. We have used a 10 Gbps bandwidth (10GBASE-T) which is currently proposed to fulfill in-vehicle communication needs. The use of a 10 Gbps network is based on the fact that, given the same tests on 1 Gbps networks, none of the delay requirements could be fulfilled.

Our topology is composed by 3 switches and 17 nodes (12 TT, 4 AVB and 1 BE).

Table II shows the simulation configuration. Bakcground traffic is generated by multimedia and WiFi sources.

The corresponding domains follow:

- **Powertrain**:Engine and transmission.

TABLE II. Simulation Traffic characteristics

Type	Payload [Bytes]	Service rate [ms]	Priority
Control	0...750	0...6	TT
Driver assistance camera	350	2...3	AVB SR Class A
Media (radio)	200	2...3	AVB SR Class B
Media (video)	1428	2...3	AVB SR Class B
WiFi	750	0.5...0.6	Best-Effort

- **Chassis**:Brakes and steering.
- **Bodywork and comfort**: windows, air conditioning, lights, locks, radio, video, streaming.
- **Driver assistance**: Front and rear cameras.

A. Simulation characteristics

In our simulation, we have used 60 different seeds, which define a random value with uniform distribution to establish the node transmission time, in order to avoid any type of synchronization effect. The initial cycle length is defined on 6 ms and each node must transmit a packet during this cycle. The duration of all simulations is 15 s and we execute 30 runs per configuration. The simulation duration was calculated to obtain 2500 cycles during the specified time.

V. Results

Figure 4 shows the delay distribution when using a 6 ms cycle, which is not uniform. Since the node-generated traffic is asynchronous, there is a high probability that the packet arrives before or after with respect to the corresponding transmission slot, thus missing one cycle until the effective transmission. It can be observed that the Chassis and Bodywork and Comfort domains have a much higher latency than those specified on Table I. Table III synthesizes the behavior for all the domains, showing that they are out of specification using this configuration.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

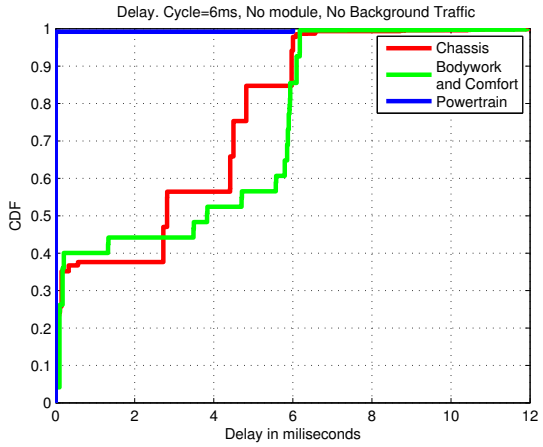


Fig. 4. Delay CDF for the first case.

TABLE III. Average Delay and Jitter for a 6 *ms* cycle.

Domain	Delay [*ms*]	*Jitter* [μs]
Powertrain	0.050958	0.0011227
Chassis	2.8	1.0516
Bodywork and comfort	3.2	1.388

A. Transmission cycle length

Taking into account the asynchronous events problem, The first parameter to examine in order to obtain a performance improvement in delay and jitter is the transmission cycle length. We propose to duplicate the cycle length to 12 *ms*.

Table IV shows the simulation results with a 12*ms* cycle length. Duplicating the size without doubling the slot length duplicates the number of available slots, however, the delay at the transmission queue increases.

B. Timeslot length

The second parameter to analyse is the slot length along the different devices on the network. TTEthernet implementation defines two types of windows: The first indicates the message arrival time, while the second fixes the outgoing time for each message. Complimentary to this configuration, there is the Permanence Point in Time which determines the maximum time a message can wait until it is

TABLE IV. Average delay and jitter for a 12 *ms* cycle

Domain	Delay [*ms*]	Jitter [μs]
Powertrain	0.086678	0.483218
Chassis	6.5	5.9369
Bodywork and Comfort	8.4	7.4451

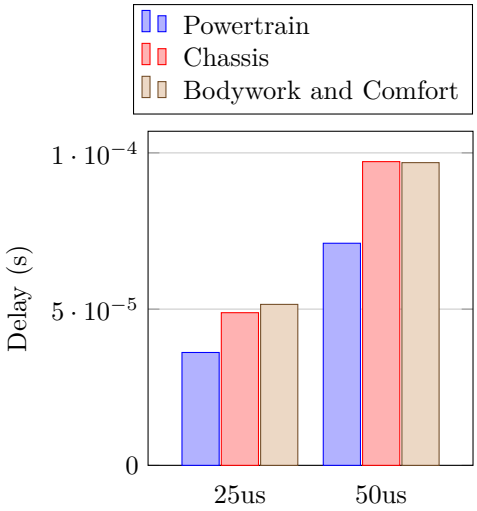


Fig. 5. Effect of changing the Slot length

transmitted. In order to analyse the impact of the slot length, the nodes on the networks are configured to transmit synchronously. We established two window sizes to test this setup:25μs and 50 μs. Figure 5 shows the results for each window size, indicating that the delay is sensitive to the slot length.

Figure 6 shows the CDF of the delay for Powertrain, Chassis, Bodywork and Comfort, and on Table V the results are quantified. When compared to Figure 4 and the corresponding Table III, the delay and jitter are reduced: For a 6ms cycle, on Powertrain from 51 *us* to 2.7 *us*, on Chassis from 2.800 *us* to 3.41 *us* and for Bodywork and Comfort from 3200 *us* to 24.75 *us*. In the case of jitter, under the same conditions, the module improvement on Chassis is reduced from 1051.6 *ns* to 2.24*ns*, on Bodywork and Comfort it is reduced from 1388 *ns* to 4.13 *ns* while keeping the same order of magnitude for the Powertrain value.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

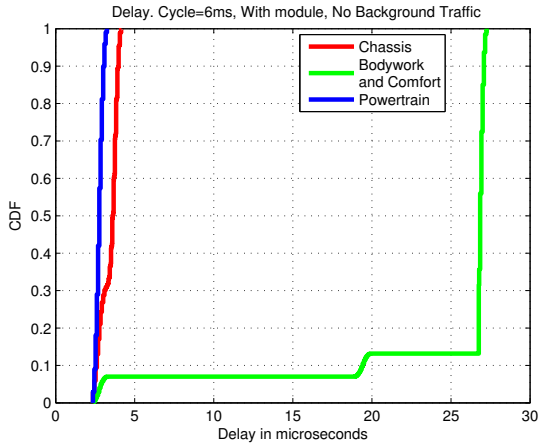


Fig. 6. CDF of the Delay with the implemented module, with no background traffic

TABLE V. Average of the Delay and Jitter for a 6ms cycle.

Domain	Delay $[\mu s]$	Jitter $[ns]$
Powertrain	2.7658	1.1981
Chassis	3.4152	2.2378
Bodywork and Comfort	24.755	4.1281

1) Cycle length: The effects of an increase in cycle length with the management module. Unlike the the cycle length increase when there is no management module present, in this case, doubling the number of slots does not generate a relevant increase on the proposed domains, thus showing the effectiveness of the proposed management module to support asynchronous traffic.

2) In presence of background traffic: In order to test the management module response under background traffic conditions, we use the configuration defined on Table II: Two nodes generate rear and forward camera video for driver assistance, two nodes represent audio and video traffic and one BE traffic generator is included to represent WiFi-type traffic. Figure 7 shows that there is a small variation in latency on the Bodywork and Comfort domain, while system performance keeps unchanged.

TABLE VI. Average of the Delay and jitter for a 12 ms cycle.

Domain	Delay $[\mu s]$	Jitter $[ns]$
Powertrain	2.7667	0.0015287
Chassis	3.8260	1.06987
Bodywork and Comfort	23.018	1.2784

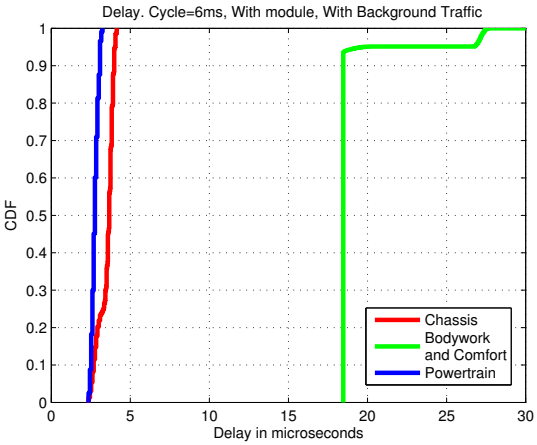


Fig. 7. CDF of the Delay with the implemented module and background traffic

VI. Conclusion and Future work

In this work, we have proposed and evaluated the design of a traffic management module for in-vehicle deterministic networks for different device domains: Powertrain, Chassis and Bodywork and Comfort, in order to comply with bandwidth, delay and jitter constraints. We have shown that our proposed management module reduces message transmission delay and jitter. Furthermore, we have also shown that the system keeps the maintains the same performance under AVB and BE background traffic conditions. We expect to continue working on this module by adding traffic load analysis for each link to enable load balancing, and adding redundancy to the network to increase robustness and reliability. Finally, we would like to analyse the effect on AVB traffic of the duplication of the cycle time to understand the performance limitations of the network.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Acknowledgment

The authors would like to thank to the PEACH STIC-AmSud 16STIC-08 / CONICYT (Chile) and "Fortalecimiento de la relaciones científicas inter-institucionales entre la Universidad Tecnológica Nacional Facultad Regional Mendoza (UTN FRM) y la Universidad Diego Portales (UDP) de Santiago Chile" from Ministerio de Educación (Argentina).

References

[1] R. A. Gupta and M.-Y. Chow, "Networked control system: overview and research trends," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 7, pp. 2527–2535, 2010.

[2] W. Shen, T. Zhang, and M. Gidlund, "Joint routing and mac for critical traffic in industrial wireless sensor and actuator networks," in *Industrial Electronics (ISIE), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 1–6.

[3] F. Soares, D. Campelo, Y. Ying, S. Ruepp, L. Dittmann, and L. Ellegard, "Reliability in automotive ethernet networks," in *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the*. IEEE, 2015, pp. 85–86.

[4] Freescale, "In-vehicle networking," 2006.

[5] Y. Lee and K. Park, "Meeting the real-time constraints with standard ethernet in an in-vehicle network," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1313–1318.

[6] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-vehicle networks: A review," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 534–545, 2015.

[7] Ixia, "Automotive ethernet: An overview," May 2014.

[8] T. Steinbach, F. Korf, and T. C. Schmidt, "Comparing time-triggered ethernet with flexray: An evaluation of competing approaches to real-time for in-vehicle networks," in *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*. IEEE, 2010, pp. 199–202.

[9] W. Zeng, M. Khalid, and S. Chowdhury, "A qualitative comparison of flexray and ethernet in vehicle networks," in *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*. IEEE, 2015, pp. 571–576.

[10] M. Abuteir and R. Obermaisser, "Simulation environment for time-triggered ethernet," in *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*. IEEE, 2013, pp. 642–648.

[11] K. Müller, T. Steinbach, F. Korf, and T. C. Schmidt, "A real-time ethernet prototype platform for automotive applications," in *2011 IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2011, pp. 221–225.

[12] T. Steinbach, H.-T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz, "Beware of the hidden! how cross-traffic affects quality assurances of competing real-time ethernet standards for in-car communication," in *2015 IEEE Conference on Local Computer Networks (LCN)*, 2015.

[13] N. Finn and P. Thubert, "Deterministic networking problem statement," 2014.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Frequency Domain Analysis of a RTOS in Control Applications

Francisco E. Páez*§, Ricardo Cayssials†, José M. Urriza*, Edgardo Ferro‡, and Javier D. Orozco‡§
*Universidad Nacional de la Patagonia San Juan Bosco, Sede Puerto Madryn
†Universidad Tecnológica Nacional - Facultad Regional Bahía Blanca
‡Universidad Nacional del Sur, Bahía Blanca
§CONICET
fpaez@unpata.edu.ar

***Abstract*—Control applications are implemented using real-time operating systems. Digital control theory is based on sampling intervals that have to be strictly met in order to get predictable behaviors. However, a real-time system may introduce execution jitters that may cause unpredictable effects on the control application. Stability, overshoot and settling time may be affected when an inadequate real-time system is used. Several papers have proposed different mechanisms to measure the jitter that a real time system produces. However, jitter can not be translated as a performance criterion in control theory. On the other hand, frequency domain techniques are widely applied in control theory for designing control strategies, as well as analyzing and measuring the performance of control mechanisms.**

In this paper, frequency domain analysis is used to measure the perturbations that a real-time operating system may produce on a control application. Harmonic distortion is defined as a criterion to evaluate the control performance of the the application. Experiments show that higher priority tasks are likely to be used for control tasks since the perturbation produced up to an utilization factor of 70% is adequate for most control applications.

***Index Terms*—Control applications, Real-Time Systems, Real-Time Operating Systems.**

I. Introduction

Several Real-Time Operating Systems (RTOS) are proposed for control applications. Digital control utilizes real-time systems to implement the control functions that the controller must achieve [1]. Real-time tasks perform each one of these control functions. These real-time tasks need to be executed periodically with exact timing constraints to guarantee that the system stays under control, meeting the design specifications. However, real-time tasks are executed by a processor according to the priority of each one of them that require to be executed. Lower priority tasks are relegated from be executed by higher priority tasks. Because the pattern of requesting tasks is not fixed in most of system, the time that a task is delayed from execution is not constant and consequently it produces a jitter in the task execution.

Several mechanisms have been proposed in real-time theory to measure the jitter that a real-time system produces on a control application. However, there is not a relationship between the jitter produced in the real-time domain and the perturbations caused by the jitter in the control domain: jitter is not a performance criterion for control theory.

Frequency domain analysis is widely utilized in control theory. The behavior of a control application may be described according to the response it produces to each input frequency. A typical control system is shown in Fig 1. The controller should be designed to accomplish the adequate attenuation and amplification of the different input frequencies to produce the desired response of the application [2]. Hence, from a frequency domain point of view, a controller is a filter designed to generate the correct close-loop response.

When temporal constraints are not met, the controller does not behave as a linear filter anymore and produces undesirable output frequencies. These frequencies are transmitted to the application through the actuators and may produce undesirable effects such as: vibration, heat, instability and noise.

A typical real-time system consists of several control functions in addition to other non-control functions (like user interaction, system maintenance, data communication, etc.). A scheduler mechanism is needed to share the processor among the different system tasks. Usually, the scheduler is implemented as a task of a RTOS. The scheduler implements a priority discipline that defines the next task that grants the processor for execution. Several priority disciplines have been proposed in real-time systems. EDF and FP are two of the most analyzed ones because of their real-time features. In [3], the real-time features of EDF and FP are compared but no analysis of these disciplines in control application is performed.

In [4], True-Time and Jitterbug are proposed as simulation tools for real-time systems in control applications. True-time proposes a stability region which bounds the jitter that a control application may support. This boundary is based on an energy analysis and consequently it produces a conservative conclusion about stability but none about performance nor perturbation. Jitterbug is a set of simulation models for Simulink/Matlab. It allows to simulate the behavior of a compound real-time/control application. However, as simulation is based on particular case studies, we cannot perform a general analysis of the features of the priority disciplines. Both tools are intended for simulation of case studies but no design criterion is proposed for control applications. In [5], [6] and [7], scheduling algorithms are proposed to stabilize control applications. However, the real-time scheduling algorithms do not provide criteria about the improvement on the control

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

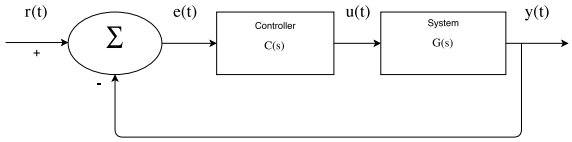


Fig. 1. Continuous-time Control System.

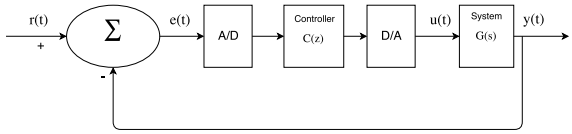


Fig. 2. Discrete-Time Control System.

performance. In [8], a bandwidth reservation mechanism for control applications is proposed, but no control performance improvement is proposed in control measurements.

In this paper, we applied frequency domain analysis to describe the perturbations that a RTOS produces in control applications. A set of random generated real-time systems were executed to measure the frequency perturbations produced for different utilization factors. We show, through these experiments, that the performance of a real-time system scheduled by a RTOS is adequate for utilization factors less than 70%.

The paper is organized as follows: Section II introduces the main concepts on state-space model and discrete-time control. Section III describes the typical task model applied to real-time theory. Section IV explains the main concepts of the frequency domain techniques utilized in classical control theory. Section V describes the mechanisms to implement control application in real-time systems. The real-time scheduling preemption is detailed in Section VI. Experiences are described in Section VII. Results are analyzed in Section VIII. Conclusions are drawn in Section IX.

II. Control Model

A controlled system consists essentially of a plant and a controller. The plant is the system to be controlled and the controller reads information from the plant and computes the actions required to achieve some control performance.

Modeling techniques are developed to express the behavior of both the plant and the controller. A model of a system is a simplified, abstracted construct used to predict the dynamics of the system. It is often possible to obtain an analytical system model using the laws that determine its behavior.

The state-space modeling is a classical control technique to model a system. For a linear, time-invariant, continuous-time system, a state-space description consists of a first-order differential equation vector, named state-space equation, for $x(t)$, named state variables,

$$x(t) = A \cdot x(t) + B \cdot u(t) \tag{1}$$

$$y(t) = c \cdot x(t) + d \cdot u(t) \tag{2}$$

where A is the system matrix, B is the input matrix, c is the output matrix, d is the feedthrough matrix, $u(t)$ is the input variable and $y(t)$ is the output variable.

The prototype control system encountered in classical control theory is shown in Fig.1. In this figure, the measured feedback signal is directly the system output $y(t)$. The input $r(t)$ is the reference input and $e(t)$ is the error signal.

The design aims to specify the transfer function of the controller, $C(s)$, to give to the closed-loop the desired control characteristics. These characteristics include stability and possibly some specification on the step response such as overshooting, settling time, and steady-state error.

A. Discrete State-Space Models

When a controller is implemented on a computer system, a discrete-time model should be utilized. Fig.2 shows a diagram of a discrete-time control of a continuous-time plant.

$C(z)$ is implemented as software and it is executed by a processor. The state-space model of a discrete-time system is expressed by difference equations:

$$x(k+1) = \Phi \cdot x(k) + \Gamma \cdot u(k) \tag{3}$$

$$y(k) = c \cdot x(k) + d \cdot u(k) \tag{4}$$

In [2], it is proved that, given a state-space description of a plant characterized by (A, B, c, d) in equations (1) and (2), and a sampling interval T, the equivalent discrete-time system is given by Φ, Γ, c, d, where

$$\Phi = e^{AT}, \Gamma = \int_0^T e^{AT} \cdot B \cdot \varrho\tau \tag{5}$$

The discrete-time model derived from this transformation is not an approximation but an exact description of the behavior of the plant at sampling instants.

The computational execution of the discrete model takes some time to complete. If the computational delay is very small compared with the dynamics of the system, it can be neglected. If the delay introduced by the computation is constant, then it may be considered in the transfer function of the controller.

On the other hand, if the computation produces a non-constant delay then the discrete-time model is not valid anymore, and consequently the same actions may produce different effects, leading to an undesirable behavior [2].

B. Specification of the Sampling Period

The selection of the sampling period of the system is important. If the sampling period is chosen too long, the continuous-time signal will not be able to be reconstructed. On the other hand, if it is chosen too small, the workload on the computer will increase and possibly the output update will not take place on time.

There exist several rules of thumb to determine the range to select the sampling frequency, defined as $ws = 2 \cdot \pi / T$ rad/sec. Most of these rules are based on relationships between

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

the sampling frequency and the closed-loop bandwidth of the system, denoted by w_B.

A higher sampling rate could turn the system very sensitive to the precision of the parameters and to round-off errors.

III. Real-Time System Scheduling

The discrete-time control model needs the periodical computation of the controller strategies. These strategies are performed by control tasks that are executed concurrently by a processor. A scheduler mechanism is needed to share the processor among the system tasks.

Real-time systems theory allows analyzing the temporal properties of a set of concurrent tasks. The general process model of a real-time system consists of a set Π of N periodic and non-periodic tasks [9]. Each task, τ_i is characterized by either its period in case of periodic tasks or minimum inter-arrival time for non-periodic ones, T_i, deadline, D_i, worst-case execution time, C_i, offset, O_i and priority, P_i.

$$\Pi = \{\tau_i = (T_i, D_i, C_i, O_i, P_i), 1 \leq i \leq N\} \quad (6)$$

Each time that a task requires the processor to be executed, it is said that the task is invoked. The ready task with the highest priority P_i is the next task selected to be executed. The notion of jitter is important for our discussion. Sampling jitter is the maximum difference between the exact sampling period of two consecutive invocations of the same task. Similarly, the output jitter is the maximum difference between the exact output instants of two consecutive invocations of the same task.

Exact, necessary and sufficient scheduling analysis conditions exist to guarantee that the temporal requirements will be satisfied in a real-time system. The most advanced technique computes the worst case response time of any task to guarantee that completes before its deadline.

The time that a task has to wait to be executed depends on the computation time required by higher priority tasks that are ready to be executed. Because the pattern of releases is not fixed, and execution times of a task may vary from invocation to invocation, this results in a variable interference and therefore a variable response time of the task. These time variations produce a jitter on both input sampling and output updating that may cause undesirable effects on the control system [10]–[12].

IV. Digital Control and Frequency Domain Analysis

The basic operation of a digital control system (Fig.2) is to read information from multiple sensors, calculate the output and send the results to actuators. $C(z)$ is implemented as software and it is executed by a processor. The input data to $C(z)$ is a sequence of numbers obtained from an A/D converter and the output is a sequence of numbers that is converted to a piecewise control signal by the D/A converter. Both A/D and D/A converters are sampled at regular intervals and transform the continuous-time transfer function $G(s)$ into a discrete-time model of the control application.

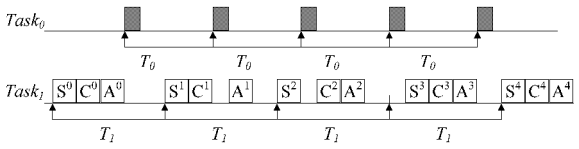


Fig. 3. Case A - control *Task*$_1$ is executed at a low priority level. The task suffers jitter due to higher priority tasks.

The controller should produce an output control signal adequate to control the application. When a classical control system is considered (linear and time-invariant), then the frequency of the signal at controller's output should be equal to the frequency of the signal at its input. This output control signal can be transformed to a summation of sine and cosine signals according to the Fourier transform. When jitter exist on the execution of the control task, then undesired frequencies appear on the output signal that are transmitted to the application through the actuators and may produce unwanted effects such as: vibration, heat, instability, noise. Consequently, the harmonic distortion may be used as an indicator of the perturbation that a control task produces when it is executed with jitter.

In this paper, we analyze the perturbation that concurrent execution of a RTOS and multiple real-time tasks produces over control tasks.

V. Scheduling Control Tasks

A control task can be divided into three subtasks with different real-time requirements:

- Sampling Subtask: this subtask has to be periodically executed to read the inputs of the system. Its execution has to be strictly periodic ($D << T$) to avoid non-linearities and time-variances that can lead to uncontrollable dynamics. The period of the sampling subtask is defined by the sampling period of the discrete-time control model and may vary in narrow range.
- Calculation Subtask: this subtask is executed after its corresponding sampling subtask has completed and it computes the control strategy. Its execution time depends on the complexity of the control strategy implemented.
- Actuation Subtask: this subtask may be executed either: (1) when the calculation subtask is completed or (2) after an fixed offset from the invocation of the sampling subtask. It should be executed as soon as it is invoked.

The control subtasks can be arranged in four main configurations:

A. Case A

The sampling subtask is executed periodically. The calculation subtask is executed when the sampling subtask is completed and then the actuation subtask. Fig.3 shows a real-time system that consists of two tasks: *Task*$_0$ and *Task*$_1$. While *Task*$_0$ is a non-control task, *Task*$_1$ is a control task constituted by its three corresponding subtasks: sampling subtask, S^i, computing subtask, C^i and actuation subtask, A^i, where i

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

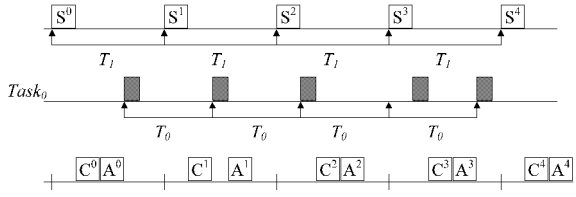


Fig. 4. Case B - the sampling subtask is assigned to a higher priority to avoid sampling jitter.

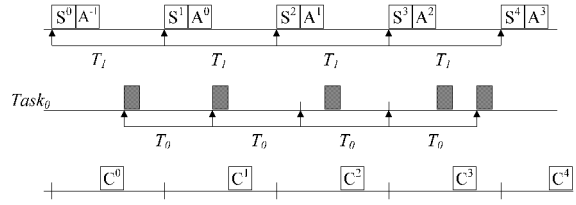


Fig. 5. Case C - Both sampling and actuation subtasks are assigned to a higher priority to avoid both sampling and actuation jitter. Actuation delay is introduced.

is the invocation index. In Fig.3, $Task_0$ is assigned with a higher priority than $Task_1$ to show the interference from higher priority tasks.

In this case, the arrival of a higher priority task may delay either the release of the task or the final actuation resulting in both input and output jitter. In Fig.3, sampling jitter on the $Task_1$ takes place because of, for instance, the interval between S^2 and S^3 is greater than T_1 while interval between S^3 and S^4 is less than T_1. Actuation jitters can be noted because of the interval between A^0 and A^1 is greater than T_1 whilst the interval between A^3 and A^4 is less than T_1.

B. Case B

The above problem of input jitter can be alleviated if the task is split into two computational entities (threads), the sampling subtask and the rest and where each part is executed at a different priority. By running all sampling subtasks at a higher priority, the input jitter is reduced dramatically. Fig.4 shows the execution of the three subtasks of $Task_1$ in two different priority levels.

C. Case C

The previous configuration solves the problem of input jitter, but not output jitter. A simple extension to overcome the problem is to run the actuation subtask at a higher priority together with the sampling task.

In this case, the sampling subtask is executed periodically. The actuation subtask is executed just after the sampling subtask is completed, however it outputs the result of the calculation of the previous invocation. All computation subtasks run as before at lower priorities. Because this is a fixed delay of precisely T units, its impact can be modeled precisely in the discrete-time control model. In this situation we have no output jitter but only a longer delay in the actuation. Fig.5 shows the execution of the three subtasks of the $Task_1$.

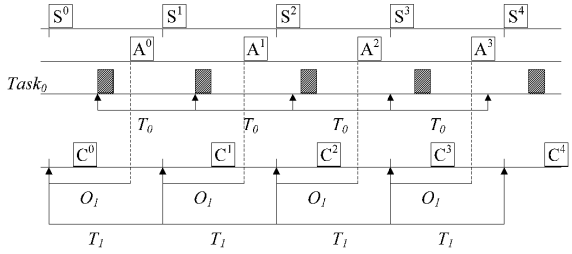


Fig. 6. Case D - The actuation subtask is invoked at a determined offset. Sampling and actuation jitter are eliminated.

D. Case D

The previous case is a particular case of the most general problem of having the subtasks at different priorities. In this case, the sampling subtask is executed periodically. The actuation subtask is executed with a fixed offset from sampling period. Response time analysis techniques can be used to compute the smallest offset that ensures that the computation subtask will always have completed before the actuation subtask is released. In this case, there is no output jitter and a smaller delay than in the previous case.

Fig.6 shows the execution of the three subtasks of $Task_1$ in two different priority levels. When the priority of the actuation subtask is just below the sampling subtask and $O_1 = T_1$, this case reduces to case C.

Complexity and overhead are the disadvantages of this approach. This configuration requires almost as much resources of the systems as if they were three independent tasks. Besides, an optimal priority assignment is not trivial.

Real-time tasks that implement control functions can be scheduled according to any of the cases described above. However, when two or more control functions have to be performed simultaneously, the real-time scheduling mechanism will produce perturbations on the control application independently of the case selected.

VI. Scheduler Preemption

Hard real-time theory considers that no deadline can be missed. Real-time schedulers may implement non-preemptive as well as preemptive algorithms to make it suitable to the application. Non-preemptive algorithms are designed so that once a task switches to the execution state, it is not removed from the processor until it has completed. In non-preemptive algorithms, the context switching is called only when the task completes or blocks. On the other hand, preemptive algorithms are driven by the notion of prioritized computation [10], [13]. The task with the highest priority should always be the one currently assigned to the processor. If a task is being executed and a new task with a higher priority is released, the task on the processor should be removed and returned to the ready state until it is once again the highest-priority ready task in the system.

Non-preemptive algorithms introduce a limited scheduling overhead because of the reduced number of context switches.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

On the other side, executing a non-preemptive scheduling algorithm might lead to limited schedulability performances due to the large blocking imposed on tasks with smaller deadlines [9]. Preemptive schedulers improve the response time of the highest priority tasks.

Weakly and firm real-time systems are proposed to design systems that may miss deadlines in a controlled manner [14]. However, no analysis are performed for this kind of real-time systems for control applications.

In this paper, we evaluate the control performance of a real-time system implemented over a RTOS with a preemptive algorithm with a Fixed Priority policy.

VII. Experimental Set-Up

In this section we perform a performance evaluation experiment to analyze the frequency perturbation produced by a RTOS used in control applications. We choose FreeRTOS version 8.1.2 as the RTOS for the experiment.

We generated a set of real-time systems with the following features:

- We considered real-time systems with 5, 7 and 10 tasks.
- We considered utilization factors of 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90 and 95 percent.
- Task execution times were randomly generated with a uniform distribution in 3 different ranges: (1) between $25\mu s$ and $1000\mu s$, (2) between $25\mu s$ and $10000\mu s$ and (3) between $25\mu s$ and $100000\mu s$.

A thousand real-time systems were generated for each combination of number of tasks, utilization factor and task execution times. Each real-time system was executed long enough to have at least 300 invocations of each task. All the real-time systems were executed on a MBED LPC1768 board with an ARM®Cortex™-M3 MCU running at 96MHz.

Each task of the system was programmed to perform the following activities: (1) read a value from a table, (2) wait some time in order to produce the required execution time and (3) write the data into a memory variable. The reading is associated with the sampling action of the control task while the writing is associated with the output updating action.

The table stores 32 values of a sine wave. Each time that the task is invoked, it reads a value and increments a pointer to the next value of the table for the next invocation. Therefore, as writing is performed periodically, it produces an output of a sine wave. Each time that a writing event takes places, it is logged in a file for further frequency domain analysis.

Frequency domain analysis was applied to the data obtained to show the perturbations that each one of the priority disciplines produces on the control tasks. Ideally, if a task τ_i is executed with no jitter, then the spectrum analysis shows a very sharp component at frequency $f_s = 1/(T_i \cdot$ lenght of the table). When the task is executed with jitter, different harmonic frequencies appear in the spectrum analysis (Fig.7).

The perturbation produced by the RTOS and the higher priority real-time tasks is then calculated subtracting from the

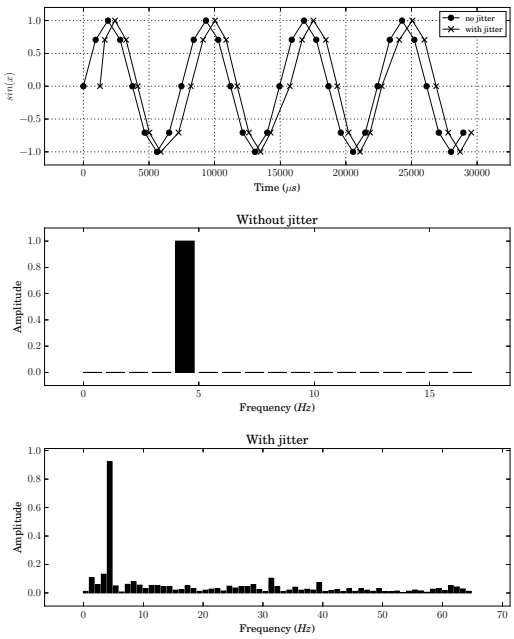


Fig. 7. Example of the *frequency spectrum* of an ideal (no jitter) real-time execution, and the frequency spectrum of a high harmonic distorted signal (with jitter). The $sin(x)$ values come from a precomputed table.

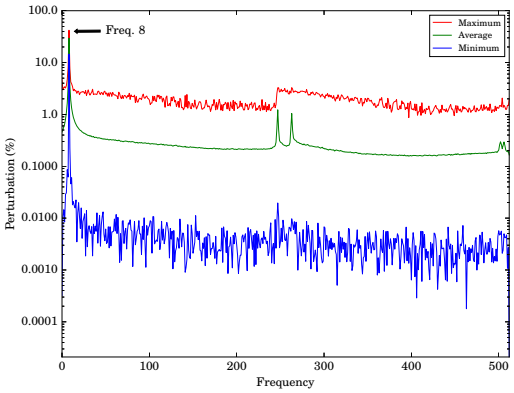


Fig. 8. Maximum, Average and Minimum perturbation for frequency components for lowest priority task of a system of 10 tasks and utilization factor equal to 90%.

frequency spectrum obtained for each task, the ideal frequency spectrum that should be produced in a jitter-free execution.

Fig.8 shows the minimum, average and maximum perturbation for each frequency component. The figure is an example of the lowest priority task of a real-time system with 10 tasks and utilization factor of 90%. The component number 8 is the component corresponding to the frequency of the sine wave produced by the execution of the real-time task (fundamental component). It can be noted that the perturbation is maximum for this component in which the minimum, average and maximum values are very close. The percentage of the perturbation for the fundamental component reaches up to 30% and up to 2% for the rest of the components.

Fig.9 shows the average of the maximum percentage of

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

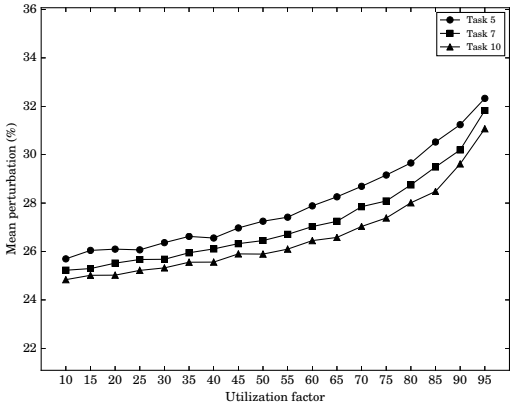


Fig. 9. Percentage of average of the maximum perturbation for the lowest priority task for real-times system with 5, 7 and 10 tasks.

component perturbations for the lowest priority task for system with 5, 7 and 10 real-time tasks. The average of the maximum perturbation is around of 24%-26% for the lowest utilization factors, increasing for higher utilization factors. It also can be noted that the perturbation produced depends on the utilization factor and the behavior is similar for different number of tasks.

VIII. Discussion

Real-time system jitter produces perturbations on control applications. Frequency analysis shows that perturbation may reach 32% when multiple real-time tasks are executing concurrently, and a RTOS is utilized.

It can be noted that the perturbation is distributed among all the frequency components of the spectrum with a maximum perturbation in the fundamental component. Fig.8 shows a pattern that persists for all the task of the system, for systems with different number of real-time tasks.

The magnitude of the perturbation should be taken into account when real-time systems are design for control applications. Control actions should transfer energy to the application in order to obtain a controlled behavior. Control theory is utilized to define the adequate actions for each application. However, perturbations produced by real-time system may: (1) produce an unpredictable behavior of the system and (2) produce a waste of action control energy. Low power consumption mechanisms in real-time systems should be compared to the wasting of energy that perturbations produce in control actions.

IX. Conclusion and Future Work

Real-time systems are used to implement controllers in control applications but there is not a straightforward relationship between real-time systems and control applications. Real-time theory deals with periodic tasks, missed deadlines and scheduling disciplines. Classical control theory applies frequency domain techniques to design and evaluate the performance of the controllers. These controllers are designed to produce a linear, time-invariant response. An inadequate implementation of the controller may cause undesirable and unpredictable consequences on the control application. However, real-time scheduling analysis cannot be utilized to evaluate the control performance of the controller implemented as a real-time task. In this paper, the real-time performance of a RTOS-based system for control applications is tested. The experiments measured the harmonic distortion that the real-time tasks produce. This approach could be helpful in the evaluation of a real-time system with control parameters, and give insights for an adequate implementation of a control application.

Acknowledgment

The authors would like to thank the anonymous reviewers for their comments and recommendations, and the ARM University Program, which provided the mbed microcontrollers used in the experiments.

References

[1] D. Goswami, R. Schneider, A. Masrur, M. Lukasiewycz, S. Chakraborty, H. Voit, and A. Annaswamy, "Challenges in automotive cyber-physical systems design," in *2012 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS XII, Samos, Greece, July 16-19, 2012*, pp. 346–354, IEEE, 2012.

[2] R. Vaccaro, *Digital Control: A State-space Approach*. McGraw-Hill series in electrical and computer engineering: Control theory, McGraw-Hill, 1995.

[3] G. C. Buttazzo, "Rate monotonic vs. EDF: judgment day," in *Embedded Software, Third International Conference, EMSOFT 2003, Philadelphia, PA, USA, October 13-15, 2003, Proceedings* (R. Alur and I. Lee, eds.), vol. 2855 of *Lecture Notes in Computer Science*, pp. 67–83, Springer, 2003.

[4] A. Cervin, *Integrated Control and Real-Time Scheduling*. PhD thesis, Lund University, 2003.

[5] A. Aminifar, P. Eles, and Z. Peng, "Jfair: a scheduling algorithm to stabilize control applications," in *21st IEEE Real-Time and Embedded Technology and Applications Symposium, Seattle, WA, USA, April 13-16, 2015*, pp. 63–72, IEEE, 2015.

[6] R. Majumdar, I. Saha, and M. Zamani, "Performance-aware scheduler synthesis for control systems," in *Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011, part of the Seventh Embedded Systems Week, ESWeek 2011, Taipei, Taiwan, October 9-14, 2011* (S. Chakraborty, A. Jerraya, S. K. Baruah, and S. Fischmeister, eds.), pp. 299–308, ACM, 2011.

[7] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Stability-aware analysis and design of embedded control systems," in *Proceedings of the International Conference on Embedded Software, EMSOFT 2013, Montreal, QC, Canada, September 29 - Oct. 4, 2013*, pp. 23:1–23:10, IEEE, 2013.

[8] A. Aminifar, E. Bini, P. Eles, and Z. Peng, "Designing bandwidth-efficient stabilizing control servers," in *Proceedings of the IEEE 34th Real-Time Systems Symposium, RTSS 2013, Vancouver, BC, Canada, December 3-6, 2013*, pp. 298–307, IEEE Computer Society, 2013.

[9] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[10] G. C. Buttazzo, M. Bertogna, and G. Yao, "Limited preemptive scheduling for real-time systems. A survey," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 3–15, 2013.

[11] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Trans. Computers*, vol. 53, no. 11, pp. 1462–1473, 2004.

[12] K. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design, Third Edition*. Dover Books on Electrical Engineering, Dover Publications, 2011.

[13] Y. Wu and M. Bertogna, "Improving task responsiveness with limited preemptions," in *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2009, September 22-25, 2008, Palma de Mallorca, Spain*, pp. 1–8, IEEE, 2009.

[14] G. Bernat, A. Burns, and A. Llamosí, "Weakly hard real-time systems," *IEEE Trans. Computers*, vol. 50, no. 4, pp. 308–321, 2001.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

ISVV applied over space embedded software

Paola Pezoimburu, Federico López, Florencia Rao
Unidad de Sistemas Embebidos
SUR Emprendimientos Tecnológicos
Ciudad Autónoma de Buenos Aires, Argentina
paolapezoimburu@suremptec.com.ar, federicolopez@suremptec.com.ar, florenciarao@suremptec.com.ar

Abstract— In the development of a satellite, there are key subsystems that can be classified as critical to the success of the mission, since a failure in any of them could threat the fulfillment of the objectives of the mission. These subsystems require the application of rigorous verification and validation methodologies in order to ensure a high degree of dependability. This article describes the independent software verification and validation process conducted on the SAR antenna release and deployment system, as part of the development of SAOCOM mission.

Keywords— ISVV; Embedded Systems; Software; Dependability; Verification; Validation; Space Systems

I. INTRODUCTION

In the frame of the National Space Plan, CONAE is developing the SAOCOM satellites constellation, which comprises two L-band Synthetic Aperture Radar (SAR) satellites. SAOCOM satellites constitute a system of Earth observation, dedicated to the use of remote sensing data for optimization of socio-economic and scientific studies, that will integrate the SIASGE (Italian – Argentine Satellite System for Emergency Management), jointly with the satellites of the COSMO-SkyMed Italian constellation.

The SAOCOM SAR antenna dimension in deployed configuration is intended to be 10 x 3.5m. In order to fulfil the launcher envelope the antenna is divided in 7 panels, a central one and two wings folded at both sides of the spacecraft. An embedded system based on 8051-architecture microcontroller is dedicated to control the release and deployment of the panels. Considering that a failure in this control unit could result in a wrong deployment sequence, or even in a missing of the deployment, this subsystem can be classified without doubts as a mission-critical system.

Moreover, the Independent Software Verification and Validation (ISVV) is a process carried out by an organization independent of the software supplier that is intended to improve quality and reduce costs of a software product. ISVV is fundamentally targeted at critical systems, and because of that the European ECSS Software Engineering standard [1] recommends it as part of space software project processes.

In this context, in the role of customer of software development, CONAE decided to perform an ISVV process over the software embedded in SAOCOM SAR release and deployment control unit.

Following sections describe the verification and validation techniques and methodologies selected by SUR Emprendimientos Tecnológicos to perform the ISVV process over the SAOCOM SAR release and deployment system. Although specific technical details about the system under analysis, as well as the ISVV specific results, are not included due to non-disclosure agreements affecting this information, the aim of the article is to present the case as an important precedent of adopting this practice in Argentina space industry.

II. ISVV OVERVIEW

The key technical objective of ISVV is to find faults and to increase the confidence in the software, which should therefore reduce the development risks. This objective can be reached by performing additional and complementary verification and validation of a software product by an organization that is independent from the software supplier.

The "independence" notion of the ISVV is valuable and should permit to have a "fresh viewpoint" on the product and on the applied process. The ISVV supplier focuses on finding possible weaknesses and faults, using different methods and tools from those of the development organization, with a “destructive” attitude. Depending on the maturity of the software under analysis, it is also possible and desirable to infer in the software design. Otherwise, the main objective of ISVV execution is to acquire more information about the developed software, and find as many weak points as possible, allowing to establish a plan for dealing with them. It is important to remark that the independent V&V process is performed parallel to the software development process, and delayed with respect to it. One of the main goals is to feedback the development process with the ISVV findings.

European Space Agency (ESA) provides a guide to support ISVV processes applied to ESA projects [2]. This guide defines the ISVV process with management, verification and validation activities, and provides advice on ISVV roles, responsibilities, planning, and communication as well as methods to use for the various verification and validation tasks. The ESA guide was taken into account as a framework for proposing and carrying out the activities described in current article, that were performed in following stages:

- ISVV level definition.
- Verification of software technical specifications (TS).

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

- Verification of software design.
- Verification of software source code.
- Validation of the software product.

For all stages the same methodology was adopted: at first planning the activities and selecting the techniques and tools to be used, then executing the activities on the corresponding software items, and finally collecting results, findings and evidences, and reporting them to customer.

III. ISVV Level Definition

The objective of the ISVV level definition is to limit the scope and guide subsequent verification and validation activities as well as the methods proposed to be used to perform them. In this stage, each software component or unit is classified with a different ISVV level: 'No ISVV activities required', 'Basic ISVV is required' or 'Full ISVV is required', based on its criticality category.

Two complementary methods were executed for analyzing software criticality: Software Failure Modes, Effects and Criticality Analysis (SFMECA) [3][4], and Hardware-Software Interaction Analysis (HSIA) [4].

Due to the development was in an advanced stage at the time of ISVV starting, software architectural and detailed designs were available and then they were taken as the main input for SFMECA, in order to determine which would be the software components and functionalities to be analysed. In addition, hardware failures analysis performed by system supplier were used for starting the HSIA analysis.

A. SFMECA technique

Several techniques exist that could be applied, such as the mentioned SFMECA, the Software Fault Tree Analysis (SFTA), and the Software Common Cause Analysis (SCCA).

The SFMECA is an extension of the hardware FMECA, applied to software. The main purpose of the SFMECA is to identify potential software-caused failures, through a systematic and documented analysis of the credible ways in which a software component can fail, the causes for each failure mode, the effects of each failure, and the criticality of the analysed software based on the severity of the consequences of the potential failure modes.

In the context of ISVV process the SFMECA presented an important advantage over the other methods. Since the software functionalities could be traced to the code units, the possible failures identified by this method could be used as inputs to be reviewed in the stage of source code verification. Also, this method allowed to reveal potential system failures caused by software. Then SFMECA was the technique chosen for criticality analysis.

1) General procedure

Each software functionality was individually analysed for finding possible ways in which it could fail. Each failure mode had an associated kind of effect, at local and at system level, and according to the effect, a severity category was assigned. If existed any compensating provision to the failure mode, it was considered to possibly reduce the criticality category. Finally, criticality category was assigned for each detected failure mode, depending on the severity of effects and the existing compensating provisions. Steps described before were fully registered in a SFMECA table, where a failure mode was presented per each row, and different columns were completed with the analysis results.

It is important to remark that there were detected dependencies between some functionalities. For instance, sometimes in order to make certain functionality work, another one needs to work properly too. In these cases, it was analysed the lowest level functionality first, and the results were propagated to the upper dependency level functionalities.

2) Failure modes and causes

Failure modes were postulated considering the following four possible kinds of events:

- Functionalities not performed.
- Functionalities performed wrongly (e.g. wrong/null data provided, wrong effect).
- Functionalities performed too late.
- Functionalities performed too early.

3) Effect types and severity classification

In traditional hardware FMECA, the failure effects resulting from each failure mode shall be determined at the level of the item under investigation (local effect) and at the level of the product under analysis (end effect). For software analysis, the following considerations were taken into account:

- If the effects of the failure mode did not cross the boundaries of the software being analysed, they were considered as local effects. e.g.: a function fails.
- If the effects of the failure mode propagated through the interfaces of the software, and it could be established that the failure mode under analysis could be a cause of a system failure, then they were considered as end effects. e.g.: a command fails.

4) Criticality categories

In hardware analysis, the criticality value for a specific failure mode is assigned combining the severity and the probability of the failure mode. As in software there is no an assignment of a probability value, the criticality of a software failure mode is assigned based on the final effects at system level, considering any prevention/compensation mechanism.

Finally, in SFMECA analysis, a criticality category was assigned individually for each software functionality. Then the criticality of a module was defined as the highest of the criticality categories of the functionalities included in the module. Software criticality categories defined in ECSS software product assurance standard [5] are showed in Table I.

TABLE I. SOFTWARE CRITICALITY CATEGORIES [5]

Category	Definition
A	Catastrophic consequences

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

B	Critical consequences
C	Major consequences
D	Minor or negligible consequences

B. HSIA method

The HSIA is conducted on the hardware/software interface, primarily to identify the software response to hardware failure. The HSIA identifies the event chains and relationships between events. The objective of a HSIA is to systematically examine the interfaces between hardware circuits and software systems to ensure that hardware failure modes are being taken into account in the software requirements. Conversely it may also cover the analysis of the potential stress induced to hardware components by the software and especially in case of anomalous behaviour.

The HSIA is a method recommended by ECSS Software Product Assurance Handbook as a support to the software dependability analyses and assessment techniques. The HSIA is a way to link analyses made on the hardware, to the software. In practice and especially in on-board space systems, the emphasis is put on software in charge of the management of hardware failure cases.

Consequently, and ideally, the HSIA should be performed concurrently with the FMEA/FMECA for all hardware products involving software. If the analysis is performed sufficiently early in the programme, it can influence the hardware design and the software requirements. Particular attention should be paid to each failure mode of hardware:

- involved in compensatory provisions (redundancy, protection), especially to identify potential side effects, common modes and dependencies (e.g., through software) that may prevent the fault handling mechanisms from working properly;
- controlled by software, to identify and assess the corresponding software requirements.

The HSIA is used to verify that the software specifications cover the hardware failures according to the applicable requirements, namely Fault Detection, Identification and Recovery (FDIR) requirements describing especially the management by the software of the considered hardware failure cases. The following information is typically considered for each hardware failure mode:

- Symptoms triggering the software action (parameters accounting for the failure mode);
- Action of the software (failure isolation and recovery);
- Independency between the failure mode and the ability of the software to react as expected;
- Effect of the software action on the product functionality (through induced possible software/hardware cascading effects);
- Identification of complementary means (from the ground operators for instance).

1) HSIA execution

In practice, the HSIA is supported by the elaboration of a list of questions to apply to each identified failure mode of the considered hardware components. The method is highly dependent on the selected list of questions, which should be structured as rigorously and systematically as possible, to provide an efficient and reliable support to the analysis. The analysis concludes with a set of recommendations that could be recommended changes, or complementary analyses.

Then the formal procedure used to elaborate the HSIA is summarized in the following steps:

1. Looking up the FMECA made by the supplier for hardware failures that are incumbent on software.
2. Classify the cases of hardware failure, resulting in a set of risky situations to be analysed
3. Making a list of questions to apply to each identified failure mode of the considered hardware components.
4. Apply the questionnaire to the cases mentioned in step 2.
5. Elaborate an individual register for each hardware failure analysed, that includes the questionnaire, the analysis of answers and the conclusions.

IV. Verification of Technical Specifications

Next step in ISVV process is the verification of software technical specifications, including interface requirements.

Reviewing the software development process [6], it starts with the requirement baseline (RB) definition, which is made by the customer. This first set of requirements should define the software need in the system (the WHAT), explaining the software behaviour with respect to interfaces with other systems, hardware that should control, operations that should support, etc. Then, software supplier takes the RB definition as main input to establish the software technical specifications (TS), which should answer to the needs through a functional implementation taking into account software and hardware constraints in the system (the HOW). In summary, TS should represent a specific software solution for needs described by RB.

The independent verification of software TS attempts to verify that the RB representation in terms of function, capability, performance, safety, dependability, qualification, human factors, data definitions, documentation, installation and acceptance, and operation and maintenance is complete, correct, consistent, accurate, readable, and testable.

As main parameter to perform this verification, ECSS technical requirements specification standard [7] was taken into account. Documents prepared by software supplier ‘Software Requirement Specification’ and ‘Interface Control Document’ were verified against ECSS standard and results allowed to improve software technical specification. Findings of this activity allowed to update and complete supplier documentation, and in some cases the findings would be analysed again in the independent verification of source code.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

V. Software Design Verification

After the TS Analysis, and having performed the criticality analysis at the software unit level, the following step in the ISVV process is the Design Analysis. The Design Analysis consists on the evaluation of the design of the software products, focusing in aspects such as dependability, error handling mechanisms, initialisation / termination of software components, interfaces, processes synchronisation and budget analysis, among others.

Because of the maturity of the software under analysis at the time of starting of ISVV process, there was no space to perform the design analysis, as there was no opportunity to infer in its design, neither architectural nor detailed.

However, it should be remarked that design documentation was used to construct a traceability matrix from software TS to source code units, that later constituted one of the main inputs to independent source code verification.

VI. Independent Source Code Verification

The software source code verification consists on the evaluation of the source code of software product focusing on:

- reliability, availability and safety, ensuring that the sufficient and effective fault detection and isolation and recovery mechanisms are included,
- error handling mechanisms,
- initialisation / termination of software components,
- interfaces between software components and between software and hardware components,
- threads / processes synchronisation and resource sharing, and
- budget analysis, including schedulability analysis detection of any programming error causing the software to behave in a way that violates any of the applicable specifications.

For the verification of the source code, three different techniques were used: inspection, static analysis, and simulation testing. The techniques were used in a complementary manner, enhancing the verification process. For example, the findings aroused in the code inspection in some cases served as triggers to design test scenarios to verifying them by simulation.

For each finding the associated risks and consequences were analysed. The materials generated for verification (e.g., code executed on simulation), the description of the aroused findings and its evidence, and the risks analysis performed for each one, were delivered to customer at the end of this ISVV stage. Then, the customer and the supplier could analyse these materials and assess the necessity for doing modifications on the source code to resolve the detected vulnerabilities.

1) Source code inspection

Source code inspection is a systematic examination performed by trained individuals who look for potential defects using a well-defined process.

Particularly in ISVV process, the aim of the source code inspection along with the other methods for verification is:

- to ensure that the code is correct, and in conformity to software requirements and coding standards;
- to ensure that the code implements proper events sequences, consistent interfaces, correct data and control flow, completeness, appropriate allocation of timing and sizing budgets, and error handling;
- to ensure the code implements safety, security, and other critical requirements correctly.

During the code inspection, in particular, in order to be avoided, the following situations were looked for:

- Counters with a type different from unsigned.
- Implicit comparisons, e.g: `if(Function())` or `if(!Function())`

Also, for organising the inspection, the following rules were followed:

- Faced with functions with the same criticality, give priority to those containing conditional/nested loops
- Faced with functions with the same criticality and conditional / nested loops, give priority to those with higher level conditional / nested loops, including combinations of both
- Faced with functions with the same criticality and conditional / nested loops of the same level, including combinations of both, give priority to those with the greatest amount of exit options.

2) Source code static analysis

The goal of static analysis is to check the code properties without executing the code. Static analysis is carried out by using a specialized software tool, and it has a different perspective than a compiler. Rather than finding a way that the program can be interpreted, it looks for ways in which the program might be misinterpreted.

It is not necessary to design algorithms for testing. Static analysis is responsible for conducting a comprehensive analysis focused on syntax, logic and run-time errors like: out of bounds checking, check for uninitialized variables and unused functions, portability checking, order of evaluation, constants checking, static initialization, size of scalars, weak definitions, standard codification checking (e.g. : MISRA).

Results of static analysis are reported as a list of messages and references to the source code lines that generates the message. Each message is classified as 'error', 'warning' or 'info', due to its severity. 'Errors' and 'warnings' were checked message by message searching for the reason that led to the alert, while 'info' were globally analysed and classified.

3) Simulation testing

The last technique utilized in the source code verification was simulation testing. The aim of simulation testing is to test features that will be impossible to test in software validation.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

One of its greatest advantages is the possibility to visualize internal registers from the microcontroller; this also gives the possibility to test time restrictions that will be impossible to test through validation. Another valuable feature of simulation testing is the possibility of generating worst case scenarios as well as introducing any kind of errors to the software.

The adopted approach for implementation of test cases was analysing the software behaviour against actions triggered by commands or sequences of commands, or actions that normally take place in software initialization, but without performing any modification on software under analysis.

The input data to the "Function Under Test" is applied using the payload of the command (when a command with payload is employed) or modifying the microcontroller memory. The control points where input data is applied or output data is verified are defined for each test case.

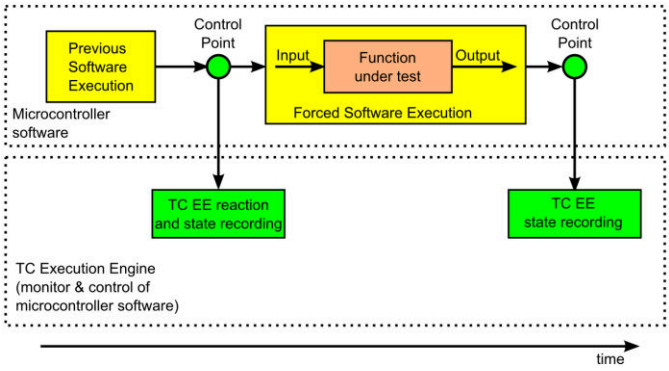


Figure 1 - Ideal model of test case implementation

Figure 1 exemplifies an ideal model of test case (TC) implementation. Some differences arise in real TC. In some cases, many control points are needed to perform a complex input or a sequence of commands. In other cases, the function under test is part of the software initialization. Therefore, the software was not forced to one execution path, as the normal software flow is the one to be tested. The control points were then used to record the state of some variables, without a reaction. A special case is the testing of Power On Self Test (POST) functions, where failure injection was used to modify the software execution path.

TC were implemented by code scripts with following flow:

- Set software data preconditions.
- Perform a software reset.
- Wait to the end of software initialization. This means that the TC Execution Engine waits until the software reaches the main loop.
- Record some state.
- Send a command.
- Wait for some time or some event.
- Record the final state.
- Compare to the expected state and computes result.

VII. Validation of the Software Product

The last step in the ISVV process is the independent software validation. The aim of this activity is testing the final embedded software, already installed on its target hardware platform, to demonstrate that the implementation meets the TS in a consistent, complete, efficient and robust way.

Independent validation looks for acquiring more information about the software performance under both nominal and adverse conditions.The execution of this activity is expected to reveal the existing software vulnerabilities, when they exist, and enables to prepare according software operation procedures to manage them, when it is possible. In case of detecting inevitable failures occurrence, validation process also allows to collect information about the failure and to recommend proper software modifications.

1) Prerequisites

To ensure efficiency of the independent validation activity, it is important that the software under test is in a mature and healthy state. To accomplish it, the prerequisites are:

- The independent verification process shall be previously performed in order to support the identification of the test cases.
- The software under validation shall be the same analysed in independent verification process, in order to obtain consistency in the results of both processes.
- The software under test shall be already validated by the software supplier. The ISVV activities are not expected to redo or replace the software supplier's validation activities.
- A suitable software validation facility (SVF) or operational test platform is available.

SVF provided by software supplier was integrated by:

- Engineering Model of subsystem under analysis, including hardware and software (the corresponding to source code independently verified).
- Electrical Ground Support Equipment (EGSE) designed to support integration and test activities during development process. This equipment provides the necessary means for communicating the system under analysis with Ground Segment (sending telecommands and receiving telemetry data). Also, EGSE simulates external interfaces between the system and other subsystems, emulating their behaviour under specific scenarios of interest.

2) Identification of test areas and test cases

The purpose of this task is to identify areas to be subject to independent validation. The identification of TC is an iterative process, where new TC might be identified during the establishment and execution of previously identified TC.

As a start point, test cases already designed by software supplier were analysed in order to understand its scope and to complement them with new derived subtests. This first analysis allowed to find certain vulnerabilities in software

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

development documentation, as well as information to be completed in some test cases definition.

Since validation task is performed after the independent source code verification, test areas and test cases emerged in TS Verification and Source Code Verification in conjunction with the test areas and cases arising in ISVV Level Definition, were taken into account for test areas and test cases designing.

Additional tests were developed to cover the product behaviour at the boundaries or even beyond the limit:

- Stress tests to verify the combinatory of the boundaries.
- Robustness tests to verify beyond the constraints and the error propagation.
- Performance tests to verify the software behaviour in operational cases; this concerns mainly the CPU load, the long duration.

In the tests that software is subject to conditions beyond the limit, knowing that tests will fail, the aim is to get to know how it failed, and if the failure is observable, identifiable and recoverable. By this approach the system could be characterized under different scenarios, and operational action could be designed to handle eventual contingencies.

The output of this task is a description of each identified test case, including the steps to be followed, expected results and pass/fail criteria.

3) Implementation of test cases into test procedures

Test procedures are the implementation of test cases, i.e., test cases expressed in the test language as provided by the SVF. The key of this activity is to achieve knowledge about the SVF or operational test platform.

4) Execution of tests procedures

Once the test procedures are written, the execution of validation tests can be done. Additional information about how to perform the tests shall be collected and later included in the independent validation report.

The main outputs of this task were the results collected in the Log Books, which included following information:

- Test area: Test cases are classified in test areas.
- Test ID: Unique ID used to identify each test case.
- Test rationale: Explains the objective of the test.
- Test overview: Description of the aim of the test.
- Input data: Necessary information to complete the test. It may include telecommands and payload data.
- Output data: Information necessary to decide if the test passes/fails. Data observed or received from subsystem under analysis.
- Starting conditions: System conditions in order to start the test. It may include the status of external interfaces or subsystems emulated by EGSE.
- Steps: Brief description of the followed steps.
- Pass/fail: Test status according pass/fail criteria previously stablished.

Additional observations and test evidences were collected, and later included in independent validation report.

5) Investigation of failed tests

Failed tests shall be analysed in order to ensure that the detected failures are due to problems in the software under test, and not to inconveniences in the software validation facility or errors in the test procedures.

In this stage, validation execution results were also reported to customer, and it was expected that both software customer and supplier review the validation procedures and results, and give additional information, especially in cases in which tests failed. This feedback was required to properly close this stage.

6) Report of independent validation results and findings

After reviewing the validation failed tests, complete set of test results was delivered to customer. A report was prepared describing how tests were executed, including observations and errors found. All Log Books containing test results, as well as the evidences collected, were attached to the report.

Findings revealed during independent validation activity were reported and analysed separately. Risks and consequences associated to each vulnerability were identified and detailed described in the finding reports.

VIII. CONCLUSIONS

Activities carried out along ISVV process allowed to detect the main software vulnerabilities and their associated risks. This valuable information could be analysed and assessed by both software customer and supplier, and in some cases resulted in software modifications as well as documentation updates.

REFERENCES

[1] ECSS Secretariat, “ECSS-E-ST-40C Space engineering - Software” ESA Requirements and Standards Division, March 2009.

[2] ESA, “ESA Guide for Independent Software Verification and Validation”, December 2008.

[3] ECSS Secretariat, “ECSS-Q-HB-80-03A Space product assurance – Software dependability and safety” ESA Requirements and Standards Division, January 2012.

[4] ECSS Secretariat, “ECSS-Q-ST-30-02C Space product assurance – Failure modes, effects (and criticality) analysis (FMEA/FMECA)” ESA Requirements and Standards Division, March 2009.

[5] ECSS Secretariat, “ECSS-Q-ST-80C Space product assurance – Software product assurance” ESA Requirements and Standards Division, March 2009.

[6] ECSS Secretariat, “ECSS-E-HB-40A Space engineering - Software engineering handbook” ESA Requirements and Standards Division, December 2013.

[7] ECSS Secretariat, “ECSS-E-ST-10-06C Space engineering - Technical requirements specification” ESA Requirements and Standards Division, March 2009.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Breaking the Barriers to Advanced Power Management in Systems on a Chip

Elías Todorovich, Ramiro Carlucho, Guillermo Paoletti
Fac. de Ciencias Exactas, Univ. Nac. del Centro de la Pcia. de Bs. As., Argentina
etodorov@exa.unicen.edu.ar

Ray Brinks, Silvano Rossi
Fac. de Ingeniería, Univ. Nac. del Centro de la Provincia de Buenos Aires
Argentina
srossi@fio.unicen.edu.ar

***Abstract*— Complexity in System on a Chip (SoC) design is rapidly escalating with the advent of more advanced nodes and the ability to integrate 100's of cores on a single die. This highlights the need to understand the power consumption of the device in the early architectural phases of SoC design. The Power Analysis tooling, centered on Network on a chip, and developed as part of this project allows for quick iteration of power management topologies and tradeoffs in power management policies. The approach followed allows for successive iteration and composability of the solution as the design progresses and additional details become available. Flexibility is present in the system to assign both degree of uncertainty and replace the imprecise information with new information as the design process is completed.**

Keywords*— Network on a chip; System on a chip; Power estimation; Programmable logic

I. INTRODUCTION

Processor, memory, and peripherals are no longer the only key considerations of SoC design, the on-board network is the communication backbone that frequently determines the success or failure of the chip being designed. Moores' law and architectural innovations enabled a rapid development of processors that achieved unprecedented performance increases (35,000X since 1978 [1]). New, larger memories were also developed to satisfy the computing requirements of complex applications running on these powerful processors. Since 2002 the high power consumption of air-cooled single-processors chips lead to a new approach to get more performance: multiple processors per chip. Furthermore, complex computing systems on a chip (SoC) continue to grow. The industry is witnessing an explosion in the number of processor cores and other programmable hardware cores in a SoC, such as DMA-capable initiators [2]. These systems require efficient interconnection far more sophisticated than earlier buses with new challenges in system design and validation. In line with the reasoning above, modern network on a chip (NoC) technologies have technological resources available for functionalities unthinkable a decade ago including the ability to efficiently and rapidly detect power saving opportunities. This paper highlights (A) a new approach towards a power-aware SoC design, and (B) an EDA tool for power estimation in a NoC-based system. A case study on a modern SoC-FPGA is presented that validates the opportunities for power savings when using a state of the art NoC.

In a NoC-based SoC, modules such as processor cores, memories and other intellectual property (IP) blocks exchange packetized data using the NoC as a subsystem for data transport. Network interfaces, routers and point-to-point links define a NoC. NoCs have the following main features: (1) energy efficiency and reliability; (2) scalability of bandwidth when compared to traditional bus architectures; (3) reusability; and (4) distributed routing decisions [2] [3].

It is possible to include power awareness in the NoC development roadmap based on the following ideas: 1) Hardware response times can save power over software response times, 2) Early detection of power saving situations represent incremental power saving opportunities, 3) Fast reaction to wake up events/requests minimizes the performance impact. An accurate power estimation environment can provide a better approximation of the potential power savings that could be obtained in the system; it also allows quick SoC topology iterations to understand the correct grouping and power partitioning to be created in the SoC.

This paper describes the development of an EDA tool that coupled with a commercially available NoC, such as Sonics Giga Network (SGN), allows for the creation of a power efficient system. Together with a correct power management domain partitioning various power management policies can be explored to determine the best tradeoffs for the design.

II. PREVIOUS WORK

Various aspects on NoC, including hardware, communication structures, application mapping issues and power consumption have been tackled and explored in the last decade [4-6], many of them focused on NoC-centered low power design [7-10]. Several relevant papers closely related to power-aware NoC have been published in the last years. However most of the previous work is focused on low power techniques applied to specific NoC components or particular target architectures. Unlike those papers, this work proposes a widespread and high-level NoC-centered power estimation tool

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

that enables designers to study power consumption figures at different stages including very early design exploration. For example, a high level power estimation methodology is introduced in [11] whereby is possible to provide cycle accurate power profile for enabling power exploration at system level for a NoC router. In [12] adaptive virtual channel is proposed as an efficient novel technique to reduce power dissipation of NoC switch. An approach focused on input buffers of packet-switched NoC, proposing an innovative dual-crossbar design that combines advantages of buffered and bufferless networks is presented in [13], achieving satisfactory results in power saving over buffered networks with virtual channels. In [14] a preference-based multi-objective evolutionary decision support system is presented with the aim to aid NoC designers assigning and mapping a prescribed set of IPs into an application task graph and into a NoC physical structure. A power management technique based on distributed frequency scaling (DFS) scheme for NoC-based Multiprocessor System-on-Chip (MPSoCs) was implemented in [15]. An analytical modeling and simulation with optimal power and minimal IC area for a high performance NoC synthesis was presented in [16]. That paper introduces a topology generation technique for performance and power efficient MPSoCs using a Tabu search based optimization method. In [17] the authors present a solution for power-aware NoC through routing and topology reconfiguration, called Panthre, showing relevant results in reduction of network total power with a minimal degradation in performance.

III. Advanced Power Management in a SoC

Advanced power management is enabled by new capabilities present in commercially available Networks-on-Chip (NoCs) which allow the creation of many power and clock domain partitions. This in turn enables chip designers to make finer grained power islands and leverage periods of inactivity in the chip sub-systems.

A. SGN NoC

This work is based on advanced commercially available NoCs from Sonics Inc., particularly the SGN technology. SGN is a configurable, on-chip network enabling the design of advanced SoC communications networks using a high-speed scalable fabric topology structure [18]. Target applications include smart mobile devices, networking, battery operated, wearable, and home consumer devices (See Fig. 1).

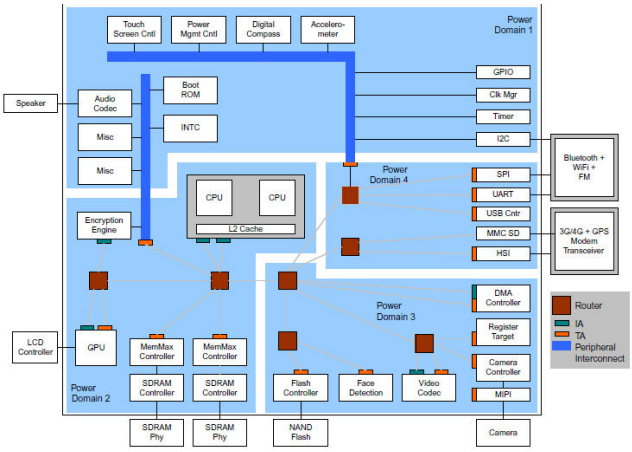


Fig. 1. SGN interconnect example.

SGN provides a high performance network for the transportation of packetized data, utilizing routers as the fundamental switching elements. Routers allow for pipelining and buffering, enabling designers to tailor these elements to their exact requirements for area and frequency.

SGN virtual channels [19], plus non-blocking flow control, and quality-of-service allow advanced concurrency. Besides that, serialized, virtual channel based, packetized fabric reduces wiring area and wiring congestion. It is possible to manage multiple power, voltage, and clocking domains globally and within local subsystems. Power management allows for very fast wake-up and shut-down. SGN has socket support for ARM Advanced eXtensible Interface (AXI3/AXI4/AXI ACE, Limited AXI Coherency Extensions) and Open Core Protocol (OCP) 3.0 protocols that yields no loss of performance or area for protocol conversion [20][21]. AHB and APB are supported through optimized bridges.

SGN IP cores, including verification IP cores, are configured and generated using Sonics EDA tools. An SGN interconnect is built using a socket-based design methodology. It supports industry-standard interfaces to the IP cores that make up an SoC. Cores connect to agents within an SGN interconnect that decouple the functionality of each IP core from the interconnect communications required among the cores, automatically adjusting for mismatches in data width, clock frequency, and protocols. This approach allows core designers to tailor the communications for each core, while still permitting SoC integrators to balance the characteristics of inter-core communication paths with respect to metrics such as latency, physical span, clock frequency usage, power domain usage, area, and power consumption. SGN NoCs can be implemented on FPGA as shown in [22].

This advanced communication infrastructure allows an arbitrary number of power and clock domain crossings in the fabric. The automation present in the system can leverage the ease of use and need of the designer to partition the chip into power and clock domain islands.

B. Power Estimation and Analysis

The SoC can be viewed as a set of composable elements that merge onto a single die multiple IP blocks that carry out independent tasks using common infrastructure resources such as DRAM. In this same fashion the power utilized by each of the units can be viewed as a set of composable elements that can be analyzed independently. The Power Analysis tool breaks down the functionality into its sub-components and allows for each of these to be modeled independently. The modeling can be carried out at various abstraction levels allowing for careful analysis of critical system components and a more abstract representation of the less important sub-units.

Three tasks are needed to complete the system design of the power management of the chip:

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

a) System level power and clock topology design. The entire SoC is partitioned into power management slices that can in turn be hierarchical containers for additional partitioning. These blocks define a set of signals that can be used to identify the level of activity and the ability to transition the slice into various stages of power conservation. Examples could include reducing frequency, removing clocks, reducing voltage levels and finally removing all power from the slice. The key information required includes power consumption of each IP core in each of its selected power management modes.

b) The second task is to group the activity signals into logical equations that determine when each power finite state machine can advance to the next step. The power management events can be summarized as a set of equations with inputs from each of the on-chip power or clock management domains. This coupled with the topology chosen and the power management policy to be applied determines the sequencing of power in the system. Many tradeoffs are present at this time that can be explored using the tooling described in this document.

c) The third task specifies the activity level in the system to model the power (or energy) consumption in the chip. Activity in each slice can be modeled using the supplied activity models or the event detection signals can be stimulated directly to create behavior patterns. It is also important to specify state transitions in each Slice (See Fig. 2).The activity level of each IP in the system is composable by power/clock domain, therefore the activity level only needs to be specified at the unit level. If no activity level is known a simpler method using signals that model activity can be used in lieu of the actual core activity.

IV. Power Analysis Tool

The Power Analysis tool flow supports two starting points and integrates the three tasks defined above as shown in the Fig. 3.

A. *Power Intent (IEEE 1801-2013)*

If the design under evaluation has a Power Intent file compliant with the IEEE 1801-2013 [23] standard, much of the required information can be imported directly. The tool includes a parser that captures the structure of the power domains and declared signals. These signals are then used to create a set of equations that represent power management events that are then mapped to a power management state. In the absence of a Power Intent file the system can be created manually.

Power Intent functions with useful information for this tool are: Create_power_domain, Add_power_state, Create_logic_net, Create_logic_port, Connect_logic_net.

Different predefined signal activity models can be selected and loaded for each signal in the design. The generic models are specified in XML format. These files contain a set of constructs, parameters and values that drive the events for the selected signal. An event is considered a value change of a signal, so these models can represent single signals, part of a core, or a complete core. The tool reads these files and generates the structure to map the activity pattern to a specific signal in the design. Some configurability is present in the models that allow the user to approximate the behavior of the cores in the real system.

B. *Built in Models*

Not all system design methodologies use Power Intent files so an alternative method to quickly incorporate design blocks to the system is to use the pre-defined models that are part of the tool. The available models are: Ethernet, Generic Model, HDMI progressive, Serial ATA, USB, Video Custom, Video Processing. Each of these has various degrees of configurability. For example, the HDMI progressive model includes a description of the active video area, consistent with the resolution of the screen. In this way, a 1920x1080p screen has 1920 horizontal pixels and 1080 active lines. Apart from the active picture region there are blanking intervals: horizontal blanking, that correspond with an end-of-line, and vertical blanking that is the equivalent to an end-of-frame. All the pixels of a single line are transmitted together until the end-of-line is triggered. After all of the lines are transmitted, the end-of-frame is triggered. In progressive video all the lines of each frame are written together. The refresh rate gives information on how many frames are refreshed on the screen in one second. Two of the available parameters to the user are `Width_1` and `Height_1`, the screen width and height, respectively.

Besides these predefined parameters in the models, the user can define new ones with specific signal activity patterns.

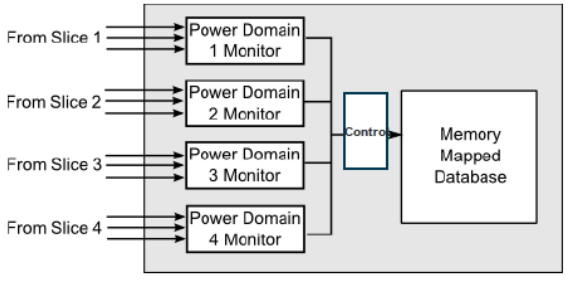


Fig. 2. Power Domain Connections

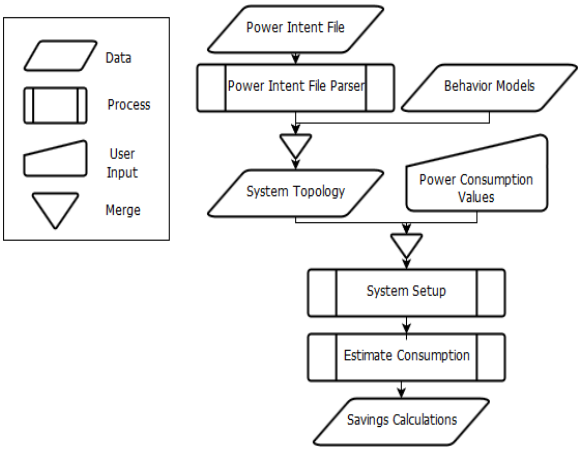


Fig. 3. Power Analysis Tool Flowchart

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Power domain states and transitions are modeled by a finite state machine that is specified in a tabular format for ease of use by the designer (see Table 1). This abstracts the actual logic in the system from the RTL implementation. The Power Management IP block is implemented according to this specification to create the power finite state machines.

Once the system is fully specified, the user sets a simulation time and runs the simulation to generate the power analysis information for the system as designed.

The power management analysis tool provides the following information: 1) Log of the state changes made by the tool under the analyzed scenarios, 2) Time in each state of the simulated scenarios, and the corresponding partial and total power consumptions, 3) Comparative analysis of the results based upon scenario contrasting, and 4) Impact of different signals and subdomains.

V. CASE STUDY

The accuracy of the Power Analysis tool was contrasted to an actual application running on an FPGA implementation of a Video Processing system. The results predicted by the tool were then compared to the actual results measured on the FPGA board.

The selected case study is based on the Zynq Base Targeted Reference Design (TRD) developed by Xilinx to demonstrate the impact of High Level Synthesis (HLS) technologies on performance and developing time on the ZC702 Evaluation Kit [24] [25]. It implements a video processing pipeline implemented in Programmable Logic (PL). The AP SoC allows the user to implement a video processing algorithm that performs edge detection on an image (Sobel filter) either as a software program running on the Zynq-7000 AP SoC Processing System (PS) or as a hardware accelerator inside the PL. The user can switch between a software and a hardware implementation and evaluate the cost and benefit of each implementation. This TRD was selected because it has a complex enough video processing system that uses AXI interconnections and is compatible with the new Xilinx Vivado development tool [26]. Moreover, it can be considered a challenging application for power management due to the hard timing constraints present in full-HD image processing. In Vivado the interconnection network is managed as other IP-cores in such a way that it can be replaced in a relatively easy fashion by the SGN NoC. Details on that NoC replacement are reported in [22].

The TRD is organized in 6 power domains in a compositional hierarchy as shown in Fig. 4.

TABLE 1. HP0 STATES AND TRANSITION DEFINITIONS

State	*Video Capture*	*Video Display*
ON (0)	On	x
ON (1)	!On	On
Standby (0)	Standby	!On
Standby (1)	!On	Standby
Off	Off	Off

Power Domain Monitors were developed to sample specific signal from the NoC of each power domain. States such as On, Idle and Off are defined for these Power Domains as shown in Fig. 4. With this information, the power manager can determine the next state for each domain. For development purposes, the Power Manager is implemented so that it stores in the FPGA memory all the specified state transitions together with a timestamp. For example, in the Video Capture domain the monitor captures end-of-line and end-of-frame events.

The information stored in the FPGA is accessed via CoreSight [27], using an ARM DSTREAM module. For further details, see [22].

The retrieved raw data needs to be imported for formatting, processing, visualization, and analysis using an additional application. This program reads the state log and shows it in a user friendly interface. In this way, the behavior of each Power Domain and the whole system can be tracked and analyzed. In this case 65,000 events were captured in each run.

A unique challenge arises using an FPGA to create the reference design and modeling the power management effects. Since the FPGA fabric is essentially a single power domain there is little opportunity to measure the power of each domain in each state directly. Therefore the project uses the timestamped activity and behavior to determine percentage of time in each state. This captured timestamp behavior is then contrasted to the predicted behavior from the Power Analysis tool. For ease of communication of results each sub-unit is assigned power consumption values coming from Vivado estimations. Since both the modeled and FPGA designs use the same power factors this does not contribute an error in the work performed.

A. *Experimental Results*

Table 2 shows detailed results for each power domain from both the runs on FPGA, and the simulations on the developed tool. In order to increase the confidence in the results, the

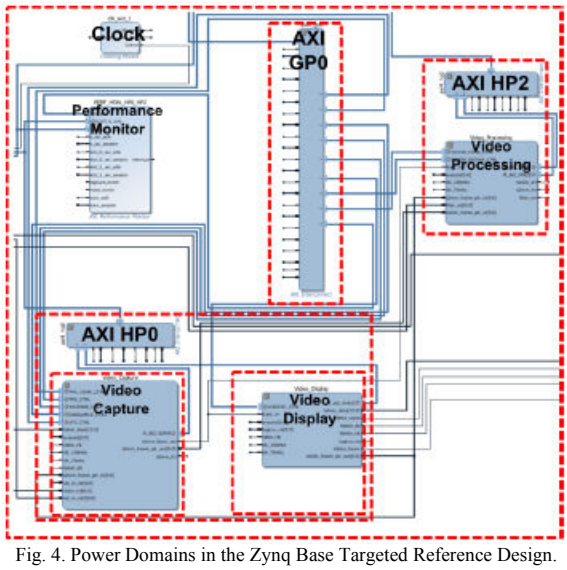


Fig. 4. Power Domains in the Zynq Base Targeted Reference Design.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

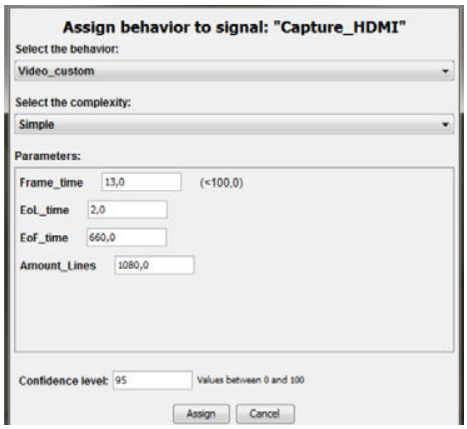


Fig. 5. Activity pattern definition in Video Capture

experiment was run multiple times. In Table 2, the averaged values are shown in column Time. To represent error values and its influence to the user, uncertainty is introduced. It considers the errors in behavioral signal patterns and how they affect the final estimated result. When choosing a behavioral model the user can define a confidence level, where 100% indicates that the model is exact. If a lower confidence level is entered, the behavioral model incorporates the uncertainty in the timing of the transitions between power states.

In this case study, confidence is obtained experimentally. The behavior of the components is known in 90-95% of the time, but there is a 5-10% when the behavior is random and hence the confidence level arises.

In order to evaluate the developed tool, the steps explained in Section IV were completed to compare experimental and simulated results.

The HP0 Slice states depend on its contained Slices. As shown in Table 1, the HP0 Slice turns off when both sub-Slices are off, turns on when at least one of the sub-Slices turns on, and is in the standby state when one of the sub-Slices is in standby and none of them is on.

Video Capture configuration is based on an HDMI activity pattern and the corresponding parameterizable model is used. Thus, the Capture_HDMI signal is set as shown in Fig. 5.

Video Capture behavior is well understood with idle time after each end-of-line and each end-of-frame. The power management states were mapped as on while reading data, standby after the end-of-line and off after the end-of-frame.

The Video Display Slice uses another HDMI signal, the Display_Buffer, to model its activity. It is similar to the Capture_HDMI module.

To configure the Video Processing Slice we know that it needs to run for 13,980 clock cycles and it is idle for 2,680 cycles. A signal representing this behavior is set. GP0 is configured so that 2.5% of the time is on.

Table 2 shows that the simulation and the experimental signal activities are almost identical in this case. In Table 3 estimated power consumptions and potential power savings are reported in a real scenario, i.e., considering a mix of different states in each power domain. The power savings are obtained by comparing the power consumption of the power domain in the ON state during the entire execution against the power consumption with the power domains changing its states. It is clearly observed that, although a very fast response times are required, the effort in low power design techniques should be on the video-related modules. These challenging response times could be obtained by state-of-the-art power-aware NoCs.

TABLE 2. POWER DOMAIN RESULTS: ON FPGA AND ESTIMATED

		Power [mW]	Time [%]		Uncertainty [%]
			FPGA	*Simulated*	*Simulated*
GP0	On	23	2.6	2.5	47.5
	Off	2	97.4	97.5	14.0
AXI HP0	On	56	95.5	97.3	6.7
	Standby	32	3.6	2.7	23.6
	Off	17	0.9	0.0	0.0
Video Processing	On	60	84.6	84.4	4.5
	Off	10	15.4	15.6	23.1
Video Capture	On	72	83.8	83.3	4.3
	Standby	17	12.3	12.8	13.9
	Off	1	3.9	3.9	40.7
Video Display	On	37	84.1	83.3	1.1
	Standby	8	11.8	12.8	2.3
	Off	1	4.1	3.9	18.0

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

TABLE 3. SYSTEM-LEVEL POWER RESULTS: ON FPGA AND ESTIMATED

	Consumption [mW]		Power Saving [%]		Uncertainty [%]
	FPGA	*Simulated*	*FPGA*	*Simulated*	*Simulated*
GP0	2.5	2.5	89.0	89.0	14.9
AXI HP0	54.8	55.4	2.2	1.2	7.1
Video Processing	52.3	52.3	12.8	13.0	7.4
Video Capture	62.5	62.2	13.2	13.6	6.9
Video Display	32.1	31.9	13.1	13.8	1.9
Global	204.3	204.2	17.6	17.7	7.1

VI. CONCLUSIONS

The complexity of SoC design requires understanding the power consumption of the device during the architectural phase of the project. At this time a high level understanding requires very fast iterations to explore multiple topology options. The Power Analysis tooling allows for successive approximations to the power understanding and the composability nature of the solution allows for replacement of sub-units as the design is better understood.

A complete analysis over a medium-complexity case study was performed to demonstrate the proposed concept. The developed tool can be integrated in an EDA tool chain and it was shown capable of instantiating a system in a simple way including support for the IEEE 1801-2013 standard. The Power Analysis tool simulates the system under evaluation and computes the time in each state of the emulated scenarios and its corresponding power consumption. As the tool enables rapidly modeling, it is relatively easy to compare the results based upon different scenarios. Thus, the impact of different SoC-based power management architectures can be evaluated to trade-off power consumption against system complexity.

ACKNOWLEDGMENT

This work was supported in part by Sonics Inc., through an internship program with the UNCPBA (RR 14148/14).

REFERENCES

[1] D.A. Patterson and J.L. Hennessy, Computer Organization and Design, Fifth Edition: The Hw/Sw Interface, Morgan Kaufmann, 2013.

[2] Benini, L; De Micheli, G. “Networks on chip: a new SoC paradigm”, IEEE Computer, 35(1), pp. 70-78. Jan. 2002.

[3] Guerrier, P.; Greiner, A. “A generic architecture for on-chip packet-switched interconnections”, in DATE, pp.250-256. Mar. 2000.

[4] Teocharides, T., “Networks on Chip (Noc): interconnects of next generation Systems on Chip,” Advances in Computers, Elsevier, vol. 63, pp. 35-89, 2005.

[5] Agarwal, A.; Iskander, C.; Shankar, R., “Survey of Network on Chip (NoC) architectures & contributions,” Journal of Engineering, Computing and Architecture, vol. 3, no. 1, pp. 1-15, 2009.

[6] Fernández-Alonso, E.; Castells-Rufas, D.; Joven, J.; Carrabina, J., “Survey of NoC and programming models proposals for MPSoC,” Int. Journal of Computer Science Issues, Vol. 9, no. 2, pp. 22-32, 2012.

[7] Lee, K.; Lee, S.; Yoo, H., “Low-power Network-on-Chip for high performance SoC design,” IEEE Trans. on VLSI Systems, vol. 14, no. 2, pp. 148-160, 2006.

[8] Guang, L.; Liljeberg, P.; Nigussie, E.; Tenhunen, H., “A review of dynamic power management methods in NoC under emerging design considerations,” in Proc. of NORCHIP 2009, pp. 1-6, 2009.

[9] Berman, A., “Power Reduction Techniques for Networks-on-Chip,” Research Thesis, Israel Institute of Technology, p. 84, 2010.

[10] Bezerra, G. B. P., “Energy consumption in Networks on Chip: efficiency and scaling,” Dissertation, University of New Mexico, p. 145, 2012.

[11] Lee, S., E.; Bagherzadeh, N., “A high level power model for Network-on-Chip (NoC) router,” Computer and Electrical Engineering, Elsevier, vol. 35, no 6, pp. 837-845, 2009

[12] Ezz-Eldin, R.; El-Moursy. M.; Refaat, A. M., “Low Power NoC Switch using Novel Adaptive Virtual Channels,” International Journal of Computer Science Issues, vol. 8, no 5, pp. 303-308, 2011.

[13] Zhang, Y.; Morris Jr.; R., Kodi, A. K., "Design of a performance enhanced and power reduced dual-crossbar Network-on-Chip (NoC) architecture," Microprocessors and Microsystems, Elsevier, vol.35, no.2, pp.110-118, 2011.

[14] Nedjah, N.; Carvalho da Silva, M. V.; Macedo Mourelle, L., "Preference-based multi-objective evolutionary algorithms for power-aware application mapping on NoC platforms," Expert Systems with Applications, Elsevier, vol.39, no.3, pp.2771-2782, 2012.

[15] Raupp da Rosa, T.; Larréa, V.; Calazans, N.; Gehm Moraes, F., “Power consumption reduction in MPSoCs through DFS,” in 25th Symp. on Int. Circuits and Systems Design (SBCCI), Brasilia-Brazil, pp. 1-6, 2012.

[16] Tino, A. and Khan, G.N., "High performance NoC synthesis using analytical modeling and simulation with optimal power and minimal IC area," Journal of Systems Arch., vol.59, no.10, pp.1348-1363, 2013.

[17] Parikh, R.; Das, R.; Bertacco, V., “Power-aware NoCs through routing and topology reconfiguration,” in Proc. of ACM/EDAC/IEEE Design Automation Conference (DAC) - San Francisco, USA, pp. 1-6, 2014.

[18] Sonics, Inc., “SonicsGN Reference, On-Chip Network System OP”, March 2014.

[19] Dally, W.J., "Virtual-channel flow control," Parallel and Distributed Systems, IEEE Transactions on , vol.3, no.2, pp.194,205, Mar 1992.

[20] ARM Limited, “AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite, ACE and ACE-Lite”, 2011.

[21] OCP International Partnership, Open Core Protocol 3.0 Specification [Online]. Available: http://www.ocpip.org.

[22] Todorovich, E.; Leonetti, M.; Brinks, R., "An advanced NoC with debug services on FPGA," in IEEE IX Southern Conference on Programmable Logic (SPL), pp.1-6, 5-7 Nov. 2014.

[23] IEEE/IEC International Standard - “Design and Verification of Low-Power Integrated Circuits," in IEC 61523-4 Edition 1.0 2015-03 (IEEE Std 1801-2013) , pp.1-351, March 24 2015.

[24] Xilinx Inc., “ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide”, Sep. 2015.

[25] Xilinx Inc., “Zynq-7000 All Programmable SoC Technical Reference Manual”, February 2015.

[26] Xilinx Inc., “Vivado Design Suite User Guide, Designing IP Subsystems Using IP Integrator”, Nov. 2014.

[27] ARM, “ARM CoreSight Architecture Specification v2.0”, 2013.

2016 Seventh Argentine Symposium and Conference on Embedded Systems (CASE)
ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Modeling Embedded Applications: An Orderly Simplification of Finite State Automata Description

Eduardo Daniel Cohen, Esteban Volentini and Pablo Gruer
Laboratorio de Microprocesadores, FACET
Universidad Nacional de Tucumán
S. M. de Tucumán, Argentina
dcohen@herrera.unt.edu.ar, evolentini@equiser.com.ar, jpgruer@gmail.com

***Abstract*— This work presents an orderly path for the description and simplification of finite state automata (FSA) representation of embedded systems. This approach takes into account some typical features of this kind of systems, i.e. in a given state of a FSA, usually only a few among a large set of system inputs are taken into account. In addition to this, the system presents identical reaction to many inputs in several states. These and other considerations allow the proposal of a modeling strategy that belongs to the "divide and conquer" paradigm.**

Keywords—embedded systems; finite state automata; modeling methods.

I. INTRODUCTION

In this work, we present some methodological proposals concerning the adoption of an orderly description of finite state machines (FSM) as a formal modeling language for embedded applications. FSM are well suited for modeling and designing embedded applications and are valuable in both hardware and software implementations [1] [2] [3]. Nevertheless, as the complexity of applications grows, classical FSM approach suffers from a lack of hierarchical structure, which produces a combinatorial explosion of the number of states and transitions.

The better a system is described, the more likely it is that a good implementation will emerge [4] [5]. A good FSM description should express with precision the required level of detail to understand the intended system behavior. It is common practice to begin with a global description of the system behavior and then to add details during a refinement process consisting in the addition of states and transitions. Hence, the initially ordered automaton becomes more and more complex, as well as less and less readable and maintainable. Furthermore, this procedure is error-prone, as all the consequences of adding new inputs and/or states are frequently not fully taken into account. Therefore a methodical approach to overcome this limitation becomes mandatory during the modeling and design stages.

An approach to overcome the above-described drawback consists in the application of Statecharts [6], which requires specific professional working environments or toolboxes such as https://www.itemis.com/en/yakindu/statechart-tools/. On the other hand, as proposed in this work, it is possible apply practical rules such as representing many states and transitions as a single activity, combined with a mechanism to filter and/or aggregate input signals.

The diversity of modeled systems makes it difficult to define methods based on a systematic application of a reasonable number of rules. Rather than a rigid prescription of steps and rules, in this work we propose to adopt a flexible approach based on the identification of some features that could characterize many systems to be modeled: (1) often only a few among a set of numerous inputs are taken into account at a given system state and (2) multiple different combinations of inputs usually trigger the same reaction. We also consider the distinction between *bounded stay* states (bounded states for short), in which the automaton remains only for a limited lapse of time, usually controlled by a timer, and *unbounded stay* states (unbounded states for short), where the system stays most of the time waiting for inputs to occur. Usually these unbounded states implement the main functional operations of the system and they can be preempted, generally by user-triggered activities such as reconfiguration.

This work proposes a modeling approach that belongs to the divide and conquer paradigm and is based in the following set of assumptions:

- A given system may assume a high number of states. Many of them are bounded states and therefore the system spends a very low proportion of its execution time on them. On the other hand, unbounded states usually represent the system's behavior in normal conditions that include most of the user's functional specifications, e.g. an alarm system that is in a disarmed state. It is important to incorporate description tools adapted to each one of the situations mentioned above.
- The number of system inputs may be high but most of the states are sensitive only to a reduced subset of input signals. As an example, consider an FSM that controls an elevator: if a state represents a floor, next state depends only in neighboring floor sensors, and only these inputs are relevant for transitions to a new state.
- Usually, there exist many different inputs to which the system produces the same output from many of its internal states. As an example let us consider a system where the input of any critical sensor has to trigger an alarm, no matter in which state the system is.

It would be desirable to be able to take advantage of the characteristics of the assumptions described above.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

To illustrate how these features could be exploited to build a simpler FSA model of reactive systems, the case study of a home alarm system will be briefly described.

II. CASE STUDY: A HOME ALARM SYSTEM

A. General Description

In a high level of abstraction, the behavior of a home alarm system may be described by a FSM with only three states: *INACTIVE, DISARMED and ARMED.*

B. Refining the Model

The *ARMED* state can be refined by adding new states that describe different armed behaviors.

- *ARMED PRESENT*: allows the presence of people inside the house, some sensors are inactive but not all.
- *ARMED ABSENT*: all sensors are active.
- *ARMED PRESENT FORCED*: similar to "armed present" but faulty sensors are deactivated.
- *ARMED ABSENT FORCED*: similar to "armed absent" but faulty sensors are deactivated.
- *ARMED TEMPORARY*: intermediate state that allows the transition between PRESENT and ABSENT states. Some sensors are activated after a predefined time interval so that the user may leave the house.

Likewise, the *DISARMED* state needs to be refined by adding a new state *DISARMED TEMPORARY*: some sensors are temporarily disarmed to allow entering the premises.

It should be considered that to perform a transition from an *ARMED* state to a *DISARMED* state, a password consisting of five numerical keys followed by a special "Enter" key has to be typed. Thus, a refined state diagram would have to add six intermediate states. It becomes clear that successive refinement operations would lead to a rapid growth of the number of states and transitions, resulting in an unreadable state diagram.

In many cases, similar to the one described above, the system behavior has to include particular sequences of intermediate states. Each of these sequences will be called an "activity". This kind of activity may induce purely sequential thinking, which is error-prone. A common error of sequential thinking results from not taking into account other possible parallel events that could disrupt the main sequence (e.g. entering a password). In this case the first digit of the password would lock the system in waiting only for the following numerical keys. Taking into account all possible pre-empting events results in even a more complex state diagram. The following sections introduce the definitions and tools necessary to simplify the FSA that represents a given system.

III. SIMPLIFYING TOOLS

A. Initial Definitions

1) Reaction:

Let a system be in state S_i, a reaction to a system combination of inputs may consist on at least one of the following actions: (1) there is a transition from S_i to S_j and (2) there are new outputs from the system.

2) Stimuli:

A stimulus is a combination of input signal values that causes a system reaction at a given state of the system, i.e. there is at least one system state S_i such that if S_i is active and the stimulus takes place, there will be a reaction. It is important to note that, depending on the state assumed by the system, the same combination of inputs might not get any reaction. Likewise, the same stimulus may cause different reactions in different states of the system.

3) Activities:

Let the system be in an initial state S_i, an activity is defined as the complete sequence of stimuli, reactions and bounded states that allows the system to transition to a new state S_j.

We note that usually S_i and S_j are unbounded states, although this is not a necessary condition. On the other hand, states that belong to an activity have to be bounded states.

As an example, consider the activity that takes place in the system described in the case-study to transition from an *ARMED* state to a *DISARMED* one: the user has to input a five number password and an "enter" key in order to disarm the system. The user has a limited amount of time to perform this operation; otherwise a time-out will abort it.

According to the number of stimuli, it could be useful to classify activities into:

- *Atomic activities* triggered by a single stimulus. Other stimuli cannot interfere in any way. As an example, let us consider the case study again. Suppose there is one key "ARM": pressing it generates an atomic activity that brings the system to the ARMED state. These activities are fast when compared to the time elapsed between any two stimuli.
- *Non-atomic activities* composed by a succession of stimuli and their corresponding reactions. This kind of activities could be *exclusive* or not. *Exclusive* activities should be considered as atomic activities, since no interference is possible. On the other hand, *non-exclusive* activities could be pre-empted by one or more activities before they get to the final state of the sequence. As an example, consider the case study previously described: if a user types a five-digit password to disarm the system and a sensor is activated in parallel, the alarm should go off and the input activity aborted.

We point out a functional difference between atomic activities and exclusive non-atomic activities: an activity composed by only one stimulus is atomic "per se" whereas a non-atomic activity has to be defined as exclusive or not by the designer, in response to system requirements and specifications.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

B. Sequences of bounded states viewed as activities

Consider the case of a system transition between two states, S_0 and S_n, which goes through n-1 bounded states, as illustrated by Fig 1.

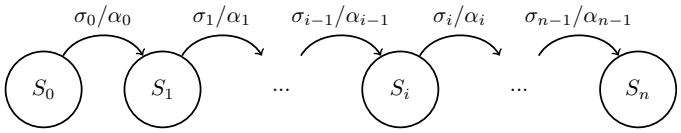


Fig. 1. A sequence of bounded states.

The sequence is triggered by stimulus σ_0 and the system transitions through states S_1; S_2; …; S_{n-1} up to state S_n producing a sequence of actions α_0; α_1; …; α_{n-1} in response to particular stimuli σ_o; …; σ_{n-1} . In the simplest case, when this sequence is atomic, i.e. it cannot be interrupted by any other stimulus, it can be defined as an activity and the state diagram can be simplified as in Fig. 2, by using the statechart style of description [4]. Strictly speaking the activity includes transitions σ_o/α_0, $\sigma_{n-1/}\alpha_{n-1}$ and super state A.

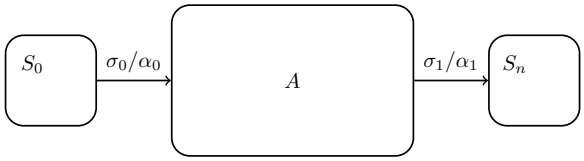


Fig. 2. An atomic activity represented in a statechart.

In many cases the activity is required to be non-atomic. The sequence from state S_i to state S_k may be preempted by stimulus σ_p. This can be modeled by super state S_p (a statechart OR state). The statechart language offers the possibility to restore the state that was active at occurrence of the preempting stimulus, by using a history connector, as illustrated by Fig. 3.

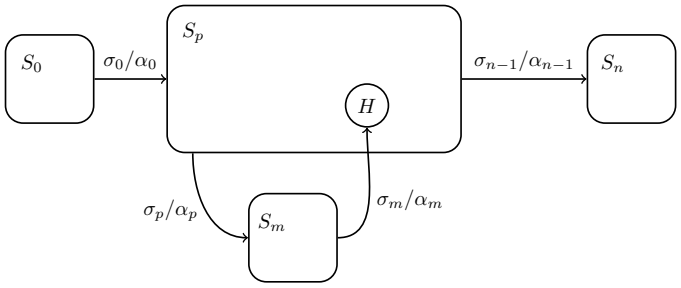


Fig. 3. Preemting stimulus in the statechart diagram.

A simpler approach is shown in Fig. 4: a thick arrow, called activity "A", replaces the sequence of Fig. 1. Stimulus σ_p may be considered as a starting stimulus of a different activity; the problems of interaction among activities are left to a subsequent and separate step of the divide and conquer technique. Thus, it will not interfere with the simplicity of the high level description being introduced.

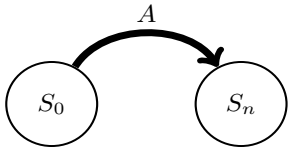


Fig. 4. Activity A hides the complexity of interaction among activities

A tool to analyze interactions among activities consists of an interaction table that will be presented in a follow-up article. Statecharts may be also used in this new step to further refine the description of activities interaction.

C. Differential perception of stimuli

Activities are reactions of the system to one or more stimuli and result in output signals (external reaction) and/or changes of state (internal reaction). Different states may allow different system reactions to the same stimulus.

1) Stimuli perception: when the system transitions from state S_i to state S_j, the set of awaited stimuli changes. Some stimuli in S_i do not affect S_j , hence they are not stimuli of S_j. It is said that in S_j the system has no "perception" of some stimuli that affect S_i.

2) External reaction to stimuli: the system outputs due to the same stimulus may be different depending on the system state.

3) Internal reaction to stimuli: from a state, S_i, the system may transition to different states depending on the next stimulus occurrence.

D. Unifying states

Let a subset of states $\mathcal{S}$ of a system be such that: (1) identical perceived stimulus generates the same external reaction on every state of $\mathcal{S}$, and (2) any transition caused by identical stimulus to different states of $\mathcal{S}$ may bring the system to any state belonging to $\mathcal{S}$ or to a unique state Q out of $\mathcal{S}$. Then $\mathcal{S}$ can be unified in a single state U, yielding a simplified state set. Note that states in $\mathcal{S}$ may not react to the same stimulus but if they do, the external reaction must be equal for each one of them. Different states may only transition to the same state out of $\mathcal{S}$ under identical stimulus.

In order to show that the previous statement is true, it suffices to take into account that transitions among states may be produced only in the following cases:

1) Self-loops on U that change the perception of stimuli, are equivalent to transitions between states belonging to $\mathcal{S}$.

2) Outgoing transitions from states in $\mathcal{S}$ to states not in $\mathcal{S}$, caused by the same stimulus, lead to a unique state Q with the same external reaction by definition. Therefore they can be replaced by a single transition from U to Q.

3) Transitions, caused by the same stimulous, entering to any state of $\mathcal{S}$ from a state W not in $\mathcal{S}$, can be replaced by a unique transition from W to U, since subsequent reactions will be identical following the two previous points.

Therefore, stimuli that cause the same reaction from any state in $\mathcal{S}$ will give the same reaction from state U and the behavior of the system is not modified.

We remark that our approach to simplifying FSMs differ from those related to non-completely specified automata models [7]: modifying system perception ensures that any input

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

stimuli not specified for state S_i will never be present when this state is active.

Although minimization algorithms [8] are based on the formal definition of state equivalence relations, the simplification showed above is not: states in $\mathcal{S}$ may be unified in U even if they do not react to the same input sequence. Furthermore, one state S_i in $\mathcal{S}$ may react to a given stimulus whereas other State S_j in $\mathcal{S}$ may not "perceive" it.

It follows that the simpler equivalent state diagram contains not only states and activities, but also different perceptions of stimuli.

E. *Modifying System Perception*

To implement the modification of the system perception as a consequence of transitions in the state diagram, a simple stimuli-masking operation is carried out. Each activity leading to a new state of the system must perform this operation when necessary.

Consider, as an example, the case of up to 64 binary stimuli, ordered as a 8x8 matrix Σ. Each position *(i, j)* represents a stimulus. $\Sigma_{i,j} = 1$ means that stimulus *(i, j)* is present, whereas $\Sigma_{i,j} = 0$ means it is not. Let the matrix PM^k be the perception mask of activity k such that $PM^k_{i,j} = 0$ if stimulus (i ,j) is masked out for activity k and $PM^k_{i,j} = 1$ if if it is not. The perceived stimuli matrix for activity k, PS^k, is given by:

$$PS^k = \Sigma \wedge PM^k \quad (1)$$

The input/output subsystem updates matrices Σ and PS^k [9]. The last executed activity updates matrix PM. Perception matrices may be constant and fixed beforehand for each activity or they may vary, depending on the particular system that is being modeled.

IV. EXAMPLE: CASE STUDY REVISITED

To illustrate the case of a state diagram simplified by means of activities, consider the alarm system introduced in section 2. Fig. 5 shows a high level state diagram that we will describe now.

Initially, all sensors are disabled and the system may react only to input keys, as the only allowed activity consists in decoding a special password. The present state is INACTIVE.

Once the special password is entered, the perception of permanent sensors (fire and anti-tampering) is enabled. A stimulus coming from any perceived sensor would trigger the alarm. The system stays in state DISARMED.

The arm-present activity adds the perception of external sensors, allowing for the movement of people inside the house. Perceived stimuli would trigger the alarm. The system gets to ARMED PRESENT state.

The reader may easily check that the ARMED PRESENT FORCED state is similar to the ARMED PRESENT state, except that the transition activity does not enable the perception of sensors, which, due to malfunctioning or other reasons, remain active. A similar case occurs with other ways to arm the system, i.e the ARMED ABSENT state.

Every different way of arming the system would require a new different armed state (i.e. ARMED PRESENT, ARMED ABSENT, …). The fact that each arming activity modifies the system perception allows unifying them in a unique state ARMED, as shown in Fig. 5. Since not all of the systems arming activities are represented in Fig 5 for simplicity's sake, every activity and the way it affects perceptions of the system is presented in Table I.

The system admits two different passwords: one to transition between INACTIVE and DISARMED states, and another one for switching between DISARMED and ARMED states. To define which of the many arming activities will take place, the user has to type a function key before the password.

In order to disarm the system, the user needs to get into the house through a path of delayed-sensors and enter a password. The first activation of a delayed-sensor will enable a time-out action (TIMER2) that consists of masking out the whole set of these sensors for a short time. If TIMER2 expires before typing the right password, the system stays in the ARMED state and the alarm is triggered.

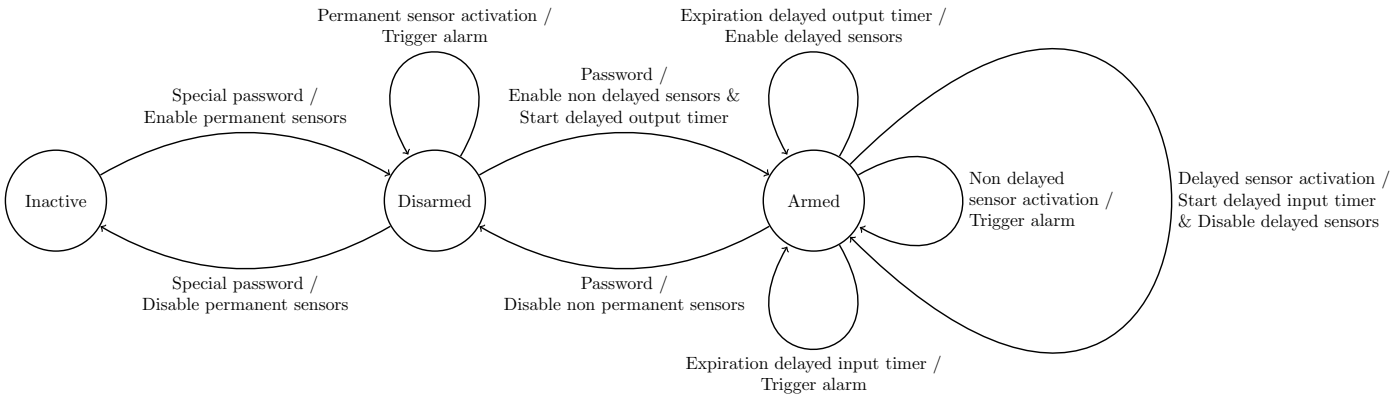


Fig. 5. High-level state diagram of the alarm system.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

TABLE I.

Activity	Perception Changes		
	Initial State	*Final State*	*Effect on Perception*
Special password	1	2	Enable permanent sensors
	2	1	Disable all sensors
Password	2	3	Enable non delayed sensors
	3	2	Disable non permanent sensors
Expiration delayed output timer	3	3	Enable delayed sensors
Delayed Sensor Activation	3	3	Disable delayed sensors

[a.] Effect of Activities into the system perception

A future version of the system may require arming a particular zone Z of the house whereas the rest remain unarmed. It is straightforward to define a new perception matrix that would allow only permanent sensors and Z sensors to be active. A new activity ARM ZONE would add this new perception matrix when bringing the system to the ARMED state. This example shows that the simplifying tools provide an important contribution to the maintainability of the system.

V. CONCLUSION

Activities are the main tool to build a simpler state representation of FSMs mainly due to the following reasons: (1) they change the perception of a system, thus allowing many equivalent states to be unified, (2) they contain bounded states, providing a way to further reduce the number of states, and (3) they provide an upper level of abstraction. Activities interactions do not need to be considered. This can be left for a following step. Statecharts and many tools from real time systems [10] may be used at this stage.

The example revisited in section IV shows that the tools introduced in this article contribute to better system maintainability. We believe that there are many systems for which these tools could bring the same advantage.

Complexity and memory requirements of this simple representation are transferred to activities and perception masks. This new approach may be considered as a way to apply the divide and conquer strategy, yielding a simpler higher-level hierarchical representation.

The activities approach allows the definition of software threads for their implementation in a programmed logic environment, such as a microcontroller, for an embedded system.

Bounded states, belonging to activities, usually differ in functionality from unbounded ones. The former are often related to exceptional cases, usually not present in the early phases of user specification. They may represent unusual and complex conditions, which could affect important features of the system such as security, reliability and disponibility. Since activities may be carried out as software threads, their design can be carried out with well-known programming languages (C, for instance).

REFERENCES

[1] F. Balarin et al., Hardware-Software Co-Design of Embedded Systems: The Polis Approach, Kluwer Academic Publishers, ISBN 0-7923-9936-6, 1997.

[2] Valvano J, Real-Time Interfacing to Arm® Cortex(TM)-M Microcontrollers, 5th Edition, Createspace, SC, USA, 2015.

[3] White E., Making Embedded Systems, 2nd Edition, CA, USA, 2012.

[4] Maier M. & Rechtin E., The Art of System Architecting, 2nd edition, CRC Press, Inc. Boca Raton, FL, USA, 2000.

[5] Keating, M., The Simple Art of SoC Design, Springer Science+Business Media, New York, USA, 2011.

[6] Harel D., "Statecharts: a visual formalism for complex systems", Journal Science of Computer Programming, Vol. 8, Issue 3, Elsevier, Philadelphia, USA, 1987, pp. 231-274.

[7] Higuchi H. and Matsunaga Y., "A fast state reduction algorithm for incompletely specified fihnite state machines", Proc. Design Automation Conference, ACM Press, New York, USA 1996. pp. 463-466.

[8] Hopcroft, J. E., "An n log n algorithm for minimizing the states in a finite automaton" in Z. Kohavi, editor, *The Theory of Machines and Computations*, pp. 189–196. Academic Press, New York, USA, 1971.

[9] Cohen, E. D., Volentini, E. & Giori, M., "Múltiples Entradas y Salidas en Sistemas Embebidos", Memoria – Investigaciones en Ingeniería, Nº 13, Montevideo, Uruguay, 2015, pp. 49-62.

[10] Laplante, P. A. & Ovaska, S. J., Real-Time Systems Design and Analysis, 4th edition, IEEE Press, John Willey & Sons, New Jersey, USA,2012.

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

ISBN: 978-987-45523-9-6 E-Book: 978-987-46297-0-8 Printed IEEE CATALOG: CFP1646V-PRT IEEE XPLORE: CFP1646V-ART

Author Index

www.sase.com.ar

August 10th-12th, 2016
University of Buenos Aires, Argentina

ISBN 978-987-45523-9-6