XVII IMEKO World Congress
Metrology in the 3rd Millennium
June 22–27, 2003, Dubrovnik, Croatia

A SELF-TESTING APPROACH TO TESTING OF MIXED ANALOG-DIGITAL MICROSYSTEMS BASED ON MICROCONTROLLERS

Zbigniew Czaja

Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics,
Department of Electronic Measurements

Abstract –A new approach based on the 4D method [1,2] is proposed to self-testing of analog networks (circuits) of mixed analog-digital microsystems controlled by microcontrollers. It is characterised by simplicity and facility of the implementation of diagnosis algorithms in simple and popular microcontrollers. In the paper the creation of a fault dictionary and self-testing procedures of analog networks (single soft fault detection and localisation) for these microsystems are described.

Keywords: self-testing, diagnosis methods, microsystems, microcontrollers.

1. INTRODUCTION

At present more and more electronic devices and microsystems consist of a digital part, used for control and processing data, and an analog part mostly used for adjustment of input signals from sensors and output signals to actuators (mixed analog-digital microsystems). Hence, new self-testing methods of these microsystems, especially for analog parts, are needed.

In many cases, a microprocessor or a microcontroller controls the work of these microsystems. Therefore, together with a main program code, we can contain in it self-testing procedures. The procedures should be realised not only in signal microprocessors, but also in simple, cheap and popular microcontrollers generally applied in practice.

Hence, a new implementation of the diagnostic 4D method [1,2], which satisfies above requirements for self-testing of the analog part of the mixed analog-digital microsystem is presented. It is illustrated on the example of a microsystem based on the microcontroller PIC16F877 [3]. This is an embedded microcontroller with RISC CPU (35 single word instructions). It contains up to 8K x 14 words of FLASH Program Memory, up to 368 x 8 bytes of Data Memory (RAM), up to 256 x 8 bytes of EEPROM Data Memory and following peripheries: 8-bit Timer0, 16-bit Timer1 and Timer2, two Capture, Compare, PWM modules, 10-bit multi-channel Analog-to-Digital converter, Synchronous Serial Port with SPI and I^{2}C, USART module and Parallel Slave Port.

2. THE 4D METHOD

The fault diagnosis via the 4D method consists of two stages. In the first pre-testing stage, a family of identification curves is generated from the following transformation [2]:

$$V_i(p_i) = \mathrm{Re}(F_i^1(p_i))\mathbf{i} + \mathrm{Im}(F_i^1(p_i))\mathbf{j} + \mathrm{Re}(F_i^2(p_i))\mathbf{k} + \mathrm{Im}(F_i^2(p_i))\mathbf{l}\,, \tag{1}$$

where: **i**, **j**, **k**, **l** – are versors, Re(·), Im(·) – real and imaginary parts of the circuit functions F^1 and F^2.

Transformation (1) maps changes of analog circuit component parameters p_i (i=1,..,N, where: N – the number of elements of the analog circuit) into the i-th identification curve in four-dimensional space (4D). Because it is difficult to present a 4D space in a figure, its two component planes are represented by exemplary families of identification curves for the analog networks (Fig. 1).

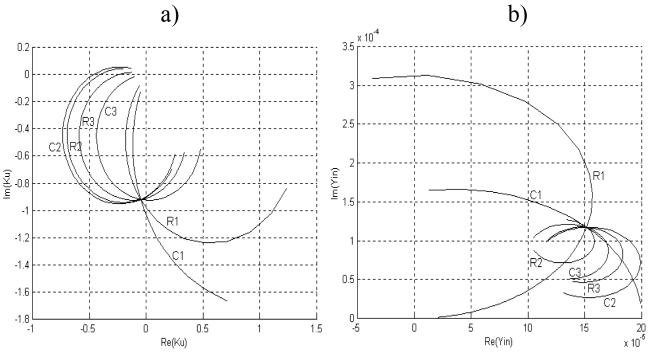

Fig. 1. The family of identification curves of an analog network: a) for the voltage transfer function *Ku* ,b) for the input admittance *Yin*

In the second stage, the real and imaginary part of two circuit functions is measured. The result, as a measurement point, is placed in this space. The affiliation of the measurement point to the proper curve localises a single fault. Additionally, if curves are scaled in suitable circuit parameter values, localisation of the point on the curve enables fault identification.

In practice analog network (circuit) elements have tolerances. This causes fuzziness of the curves and they take the form of identification stripes, as shown in Fig. 2.

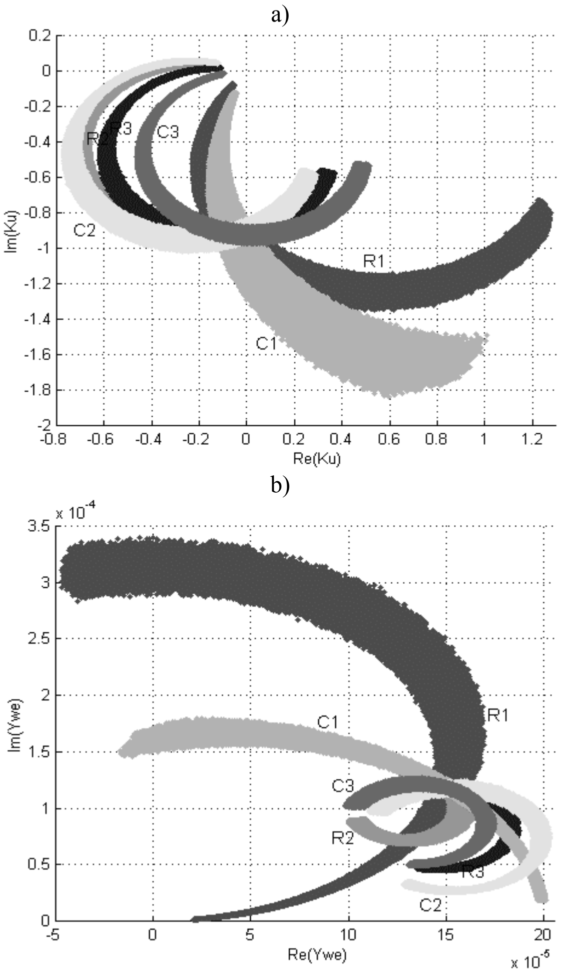

Fig. 2. The family of identification stripes of the analog network:
a) for the voltage transfer function *Ku* , for the input admittance *Yin*

These maps of identification stripes will be used to build the fault dictionary of the analog network of the mixed analog-digital microsystem.

3. THE DIAGNOSIS PROCEDURE BASED ON THE 4D METHOD

The full diagnosis procedure consists of two stages, too. In the first stage the fault dictionary is created on a personal computer using Matlab. Next it is loaded to and saved by serial EEPROM's of the microsystem.

The second stage is self-testing of the analog networks. The microcontroller, based on the measurement results and on the fault dictionary, performs the fault detection. Simultaneously, when the fault will appear, the microcontroller localises it and informs about it the microsystem. Otherwise the microcontroller goes to the main program.

3.1. Methodology of the creation of the fault dictionary

An 8-bit accuracy of measurements of the real and imaginary parts of two circuit functions is assumed. These parts are normalised to a range of values from 0 to 255.

The investigated example of the analog network is a 3rd-order low-pass filter consisting of 6 elements. The first circuit function is the voltage transfer function *Ku*, the second one is the input admittance *Yin*. They are measured at the frequency of f=735Hz. 5% tolerance for capacitors and 1% tolerance for resistors are assumed.

In the first step of fault dictionary creation for the analog circuit the map of identification stripes is generated using the transformation (1) and Monte Carlo method (Fig. 2). Next, this map for each circuit functions is divided into 256 x 256 rectangular fields (pixels). Thus each pixel has x^j co-ordinate (Re$F^j(\cdot)$) and y^j co-ordinate (Im$F^j(\cdot)$), where j=1,2 is the index of circuit function used in transformation (1). Byte representation of each pixel, and the same of the fault dictionary for the separate function is assumed (separate for Fig. 2a and Fig. 2b). Hence, a single bit of each byte in this dictionary can be assigned to the fault (element). And so, the bit number 0 means no fault if it is “1”, the bit number 1 is assigned to a fault of the resistor R1, etc (It is assumed that the number of circuit elements is less then 8, what is often satisfied in practice for these circuits). The meaning of all bits is shown in Fig. 3. When the pixel is contained in the given identification strip, “1” is written to a position in the byte represented the faulty element for which this strip was created.

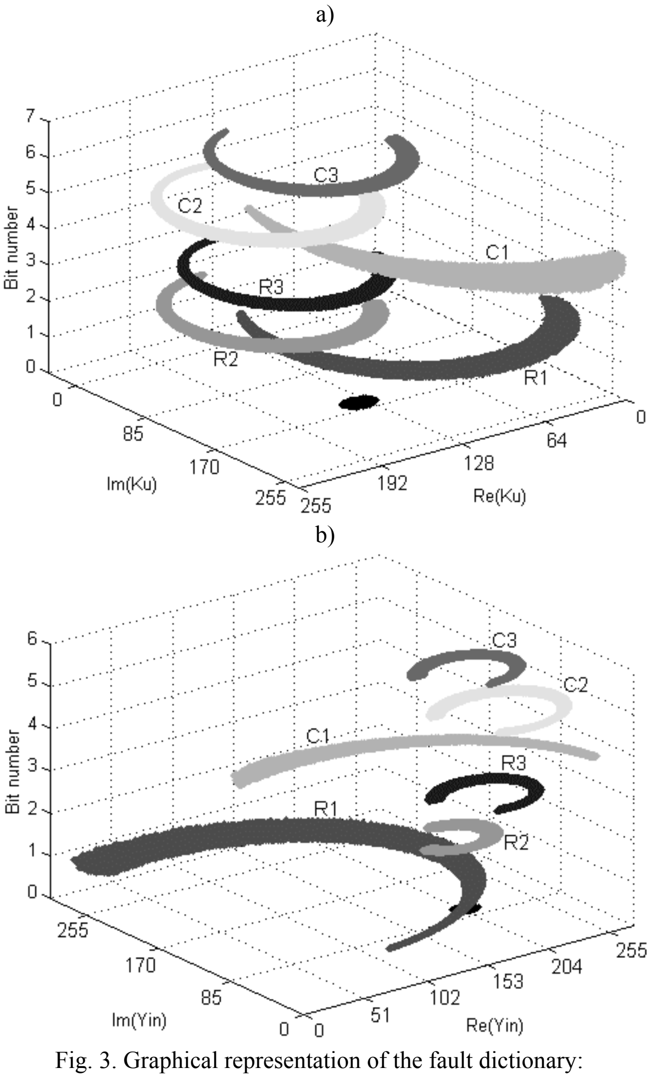

Fig. 3. Graphical representation of the fault dictionary:
a) for the voltage transfer function *Ku* , b) for the input admittance *Yin*

In this way, for each circuit function we have a variable which is a vector consisting of 65536 bytes. The variables are used to generate two files.

It was assumed that a low byte of address is the real co-ordinate *x*, and a high byte of the address is the imaginary co-ordinate *y* for each byte in the files. So byte address is described by the following formula: $x + y*256$, where $x = 0, 1, .., 255$ and $y = 0, 1, .., 255$.

For each circuit function the file with its fault dictionary is loaded to an appropriate serial EEPROM with SPI interface via RS-232 interface using functions implemented in the microcontroller (Fig. 4). The fault dictionary of the voltage transfer function *Ku* is there in chip U1, and the fault dictionary of the input admittance *Yin* is there in chip U2. A 64kB serial EEPROM with SPI was chosen (type AT25HP512 of Atmel [4]).

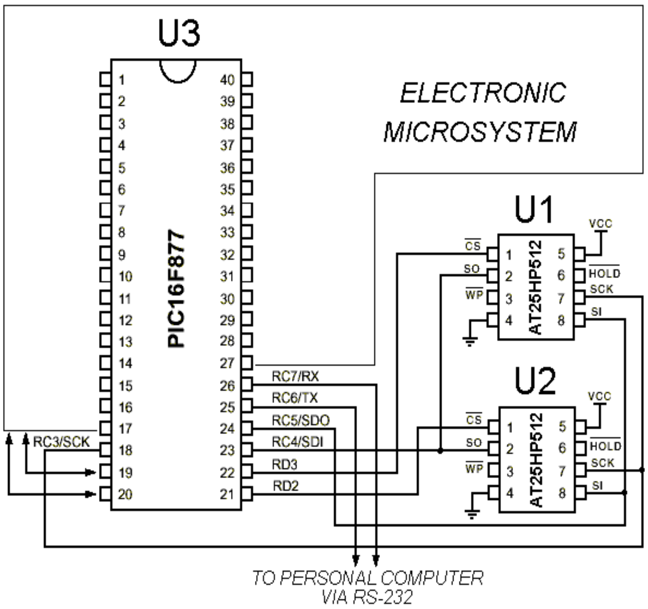

Fig. 4. Example of the mixed analog-digital microsystem based on the microcontroller PIC16F877 and with two SPI serial EEPROMs the AT25HP512

3.2. *Self-testing of analog networks*

At the start of self-testing the real and imaginary parts of two circuit functions are measured. The methodology of these measurements will be presented in separate papers. Here, it is assumed that these parts are normalised and represented by single bytes, where ReF1, ImF1, ReF2, ImF2 are results of measurements of real and imaginary parts of function F^1 and F^2 (the transfer function K_u and the input admittance Y_{in} at f=735Hz for the investigated example).

The fault detection and localisation procedure is the following: the microcontroller via SPI interface reads a byte from the first EEPROM at address (ImF1, ReF1) and a byte from the second EEPROM at address (ImF2, ReF2). Next, it operates a logical AND on these bytes. This operation improves the localisation resolution (the property of the 4D method [1,2]). If bit number 0 is set (“1”), the analog network is fault-free, otherwise the network is faulty and reading of remaining bytes enables fault localisation.

Details of the self-testing algorithm together with its assembler code are presented below.

At the beginning of the assembler code indispensable definitions are included. They deal with simplicity of code, addresses of registers, definition of instruction set for the AT25HP512 and definition of assembler constants

```
; main definitions
#define  bank0   bcf status, rp0
#define  bank1   bsf status, rp0
#define  cs1     portd, 3
#define  cs2     portd, 2

; addresses of working registers
ImF1    equ  h’20’
ReF1    equ  h’21’
ImF2    equ  h’22’
ReF2    equ  h’23’
result   equ  h’24’

; instruction set for the AT25HP512
wren   equ  b’00000110’    ; set write enable latch
wrdi   equ  b’00000100’    ; reset write enable latch
rdsr   equ  b’00000101’    ; read status register
wrsr   equ  b’00000001’    ; write status register
read    equ  b’00000011’   ; read data from memory
write  equ  b’00000010’    ; write data to memory

; assembler constants
fsr_value       equ  h’20’
conf_sspstat   equ  b’01000000’
conf_sspcon    equ  b’00100000’
```

The next part of the assembler code deals with initialisation of the SPI interface. To accomplish communication, SDO and SCK pins are configured as outputs, and SDI pin as input. The module MSSP works as SPI interface in master mode with “0” clock polarity.

```
; SPI interface initialisation
bank1
bcf portc, 3            ; SCK as output
bsf portc, 4            ; SDI as input
bcf portc, 5            ; SDO as output
bcf cs1                 ; CS1 as output
bcf cs2                 ; CS2 as output
movlw conf_sspstat
movwf sspstat           ; configure SSPSTAT register

bank0
movlw conf_sspcon
movwf sspcon            ; configure SSPCON register
bsf cs1
bsf cs2                 ; deselect U1 and U2 device
```

The last by one part of the assembler code contains function declarations regarded to service of EEPROM memory. send_data function, which co-shares the code with send_instruction function (it reduces the size of code), in first step moves data to W register from address in memory pointed by FSR register, next it increments a content of FSR register. In the next step it realises the same as the send_instruction function, so it sends via the SPI interface a byte from W register and simultaneously the read byte writes to this register. read_eeprom function reads the byte from address (*ImF*, *ReF*) in EEPROM memory. It uses the send_instruction function to write the code of the READ instruction to the memory. Next it uses two send_data functions to move two bytes of the byte address. At the end it uses the send_instruction function which generates impulses needed to read byte from the EEPROM memory. The result is saved in result register by operate logical AND (property of the 4D method [1,2]).

```
; functions service SPI interface and AT25HP512
; function send_data
send_data
    movf indf, w            ; data in W register
    incf fsr                ; FSR points next register

; function send_instruction
send_instruction
    movwf sspbuf            ; start transmission
    btfss sspstat, bf
    goto $-1                ; wait for end
    movf sspbuf, w          ; read byte in W register
    return

;function read_eeprom
read_eeprom
    movlw read              ; in W register READ code
    call send_instruction   ; send READ code
    call send_data          ; send high byte of address
    call send_data          ; send low byte of address
    call send_instruction   ; read byte from EEPROM
    andwf result, 1         ; read byte in result register
    return
```

The last part of the assembler code contains the full procedure of fault detection. At the beginning it moves to FSR register an appropriate value, so that this register points ImF1 register and it moves h'FF' value to the result register. In next step it reads the byte from first EEPROM memory (low level on CS1 pin selects device U1) and the byte from second EEPROM memory (U2 device). At the end it tests the bite number zero in the result register. If it is set „1" the microcontroller goes to initialisation and execution of the main program. If this bit is reset, the microcontroller goes to "alert" state and informs the microsystem about the appearance of a fault in its analog network.

```
; main fault detection procedure
fault_detection
    movlw fsr_value
    movwf fsr               ; FSR register points to
                            ; ImF1 register
    clrf result
    comf result             ; h'FF' in result register

    bcf cs1
    call read_eeprom        ; read byte from U1 device
    bsf cs1

    bcf cs2
    call read_eeprom        ; read byte from U2 device
    bsf cs2

    btfsc result, 0         ; testing procedure
    goto initialisation     ; if OK go to main program
    goto alert              ; else go to "alert" procedure
```

The last two parts of the assembler code are written in a way, which enables minimisation of the size of the result code. It is important, because this code should take a place as small as possible in the program memory of the microcontroller. So the all the remaining place can be used by the main program.

4. CONCLUSIONS

The paper deals with a new method of creation of the fault dictionary for self-testing of analog networks in mixed analog-digital devices and microsystems, a new self-testing procedure based on this dictionary and on the 4D method.

An advantage of the proposed self-testing procedure is the possibility of its implementation in a simple microcontroller existing in the SPI interface. It is necessary to underline the fact that in spite of its simplicity, it enables fault detection and localisation in analog networks with component tolerances taken into consideration.

The additional advantage of this approach is adding only two inexpensive serial EEPROM's to the microsystem and dedicate to them only two pins of the microcontroller (to a SPI interface we can connect other devices in parallel).

REFERENCES

[1] Z. Czaja, "The fault location algorithm based on two circuit functions", *In proc.* XVI IMEKO World Congress, Vienna, Austria, 25-28 September 2000, Volume VI, pp. 29-32.

[2] Czaja Z., Zielonko R., "New methods of fault diagnosis in electronic circuits based on bilinear transformation in multidimensional spaces", *Metrology and Measurement Systems*, Vol. VIII. Number 3 (2001), Warsaw 2001, pp. 251-262.

[3] Microchip, "PIC16F87X Data Scheet", *PDF documentation from:* http://www.microchip.com.

[4] Atmel, "SPI Serial EEPROMs AT25HP256/512", *PDF documentation from:* http://www.atmel.com.

Author: Dr Zbigniew Czaja, Gdansk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Department of Electronic Measurements, ul. G. Narutowicza 11/12, 80-952 Gdansk, Poland, phone ++48 58 347-14-87,
fax ++48 58 347-22-55 and e-mail. zbczaja@pg.gda.pl