

REPLACING MICROPROCESSOR SIMULATORS: DEVELOPING HYBRID TECHNOLOGY IN MODERN EDUCATION

Panayotis Papazoglou

National and Kapodistrian University of Athens, Evripos Complex, Psachna 34400, Greece

Abstract

Simulators are the most popular tools for learning microprocessor architecture and programming in higher education. On the other hand, specialized training hardware platforms (known as hardware trainers) have been used from the late 70's. Simulators are mainly based on hypothetical microprocessor models, while hardware trainers are based on real microprocessors but are too complicated for inexperienced users. Moreover, simulators and hardware trainers have many limitations (e.g. microprocessor model, complexity, cost) and thus a new approach is needed to effectively face these issues. This paper presents a new simple and low-cost hybrid educational platform/tool for learning microprocessor architecture and programming using a real microprocessor and overcoming existing tools limitations. The developed educational tool is suitable for all students and teachers and can be easily reproduced for supporting many university classes. This paper also presents some preliminary results of the developed educational platform.

Keywords: Simulator, hybrid platform, microprocessor programming, educational tool

1. INTRODUCTION

Microprocessor Simulators (MSs) are assembly language simulators that constitute software implementations for studying a microprocessor model which is theoretical or real. The main advantage of MSs is the visual interface (GUI) where microprocessor operation can be observed without any additional hardware equipment [1-5]. Moreover, the MS developer can simplify the corresponding GUI for offering an abstract representation of the under-study microprocessor. Simulators offer a rich GUI that can be adapted to student needs during the development phase. Today, there is a great number of available cross-platform MSs that can be used for free. On the other hand, hardware-based microprocessor trainers (MT) are used from the late 70's. These MTs consist of a real microprocessor, a multiple seven segment display, a memory and I/O system and a keyboard [6]. The MT programming is performed by entering hexadecimal numbers that represent assembly instructions and memory addresses. MTs are not suitable for inexperienced students and the purchase cost for a university class is high. Moreover, MTs are not flexible and expandable and the corresponding technical support after some years is not possible especially when the manufacturer discontinues the MT model. Also, MT can not be adapted to new user needs and thus, a new massive MT purchase has to be made. Due to the above limitations, MT solution has been abandoned by the higher education institutes. MSs are low cost or free and many of them are cross platform. Some other approaches have been also proposed in the literature for facing effectively the MSs and MTs limitations. In [7] a novel object-oriented hybrid educational tool has been proposed for supporting real hardware view and custom microprocessor architecture, but the system implementation and reproduction is complicated, and more work has to be done mainly at GUI level for delivering a complete tool for the class. Despite the flexibility of MSs, a spherical view of a microprocessor which is crucial in engineering education is not supported (lack of hardware features, real instructions, clock synchronization, bus signals, real limitations).

2. REVIEWING EXISTING TOOLS

2.1. A microprocessor trainer (MT)

Figure 1 shows an old MT (Z80-based) which represents the philosophy behind this approach. The MT is not a simulator, and the programming targets a real microprocessor. Moreover, the MT is an autonomous system without the need of a PC. For limiting the corresponding complexity of the MT, the board contains the minimum necessary components for inserting and executing instructions (e.g. microprocessor, memory, keyboard, multiple seven segment display). Due to the above features, the assembly instructions are entered using the corresponding hexadecimal codes. As mentioned in introductory section, the MTs have been abandoned due to serious limitations (e.g. technical support, maintenance cost, complexity, inflexibility).



Fig. 1. Microprocessor Trainer

Source: <https://microdev.gr/sim>. Author's web site

2.2 A simple microprocessor simulator

Figure 2 shows the GUI of a simple assembly simulator (CPUMS-available for download from author's web site, implemented in Visual Basic) where a single abstract computer model is implemented and represents the philosophy of a MS. The CPUMS supports only twelve assembly instructions that can be combined in order to form an operational program. Through the GUI, instructions are inserted in memory using the corresponding codes and starting memory address. On the other hand, the CPU model consists of only three registers: General purpose registers A,B and a Program Counter (PC). Each instruction reserves four locations in memory for future expansion. The main areas of the CPUMS are: a) memory (where program instructions are stored), b) console (for I/O operations), c) Registers (for viewing contents during program execution) and d) instructions area (where instructions are entered). Students and teachers select the most appropriate instructions to form their program. Assume that a single program is developed to display the integer numbers 5 to 1. The initial number can be entered by the user using the RD instruction. This number is stored in register A and then is displayed in the console using the WR instruction. For decreasing the A register's contents, a DEC instruction is executed. The

above steps (from WR instruction) are repeated while the content of register A is not zero. All the above steps are implemented using the available assembly instructions as follows:

RD
start: WR
DEC A
JNZ start

Thus, the student implements the algorithm do-while by using only assembly instructions that are exclusively based on microprocessor (hypothetical) architecture. On the other hand, when the simulator software is under development, the programmer designs the needed assembly instructions based on the desired complexity and microprocessor model. A MS, is a flexible and expandable piece of software that can be adapted to educational needs and at the same time is portable with zero cost.

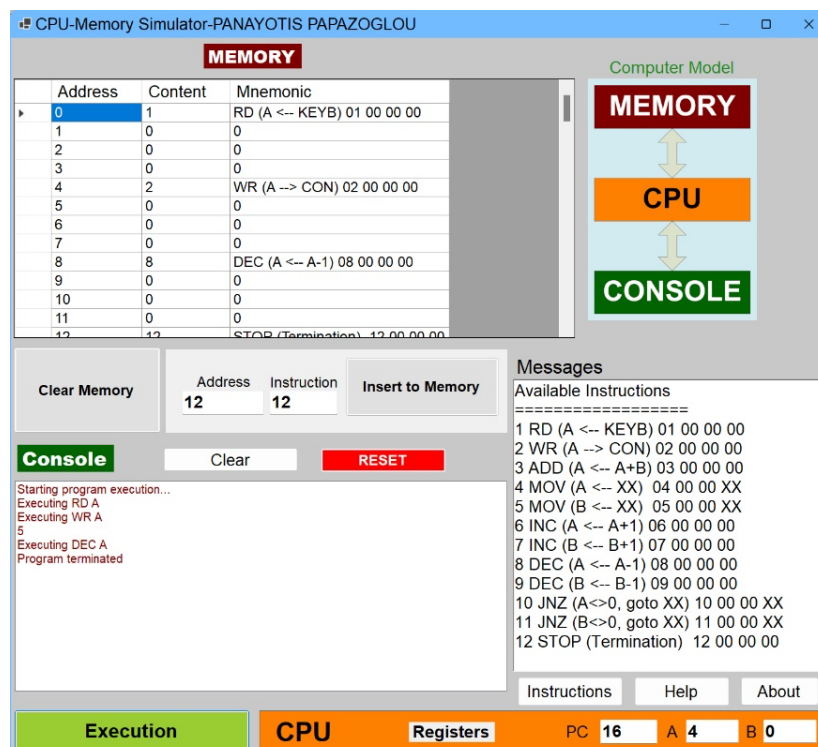


Fig. 2. MS GUI screenshot

Source: <https://microdev.gr/sim>. Author's web site

Using the MS, the student does not have any point of view of a real microprocessor and is trapped in the theoretical model. Moreover, the microprocessor study cannot be performed under real operation conditions.

3. THE PROPOSED HYBRID EDUCATIONAL PLATFORM/TOOL

3.1. The needed features

Both MTs and MSs have individual advantages that are not common between tools. MTs are ideal for studying the hardware aspects and the whole instruction set of a real microprocessor, while MSs are used for adapted theoretical education to student's needs. Based on the above features, there is an obvious need for a new tool in order to combine the advantages of the mentioned MTs and MSs. More precisely, the new tool has to support:

- a) a rich GUI that will represent programming and hardware aspects of the under-study microprocessor
- b) a real microprocessor
- c) expandability with external circuits

The new GUI has to support visually all the available functionality of the real microprocessor. On the other hand, selecting a modern microprocessor is a big problem, due to the corresponding complexity at programming and operation circuit level. That complexity makes any modern microprocessor inappropriate for education. Thus, a simpler microprocessor has to be selected. In the proposed tool, an old microprocessor has been selected because it has the following advantages:

- a) simpler architecture and simpler instruction set
- b) simpler supportive operational circuit
- c) simpler clock and control signals
- d) represents the fundamental architectural features of any microprocessor such as registers, ALU, etc
- e) is cheap

An old and simple (as compared to modern ones) microprocessor is the 6502 (introduced in 1975) which is available today in an updated version known as W65C02S (CMOS version). This model supports step by step clocking and is bug-free as compared to the original 6502, but for monitoring the pin's status within the new GUI, an intermediate circuit has to be connected between the microprocessor and PC where the GUI is implemented.

3.2. Multilayer architecture and implementation

Based on the above requirements, a multi-layered architecture has been designed for implementing a complete educational tool that supports all the features that offered individually by the MTs and MSs. Figure 3 shows the proposed architecture. The whole approach is hybrid because, a) an old technology (e.g. 6502) is combined with modern (e.g. PC and microcontroller for implementing control layer) b) a modern GUI is combined with an old technology and c) three different technologies (microprocessor, microcontroller and GUI) are unified for designing the desired educational tool.

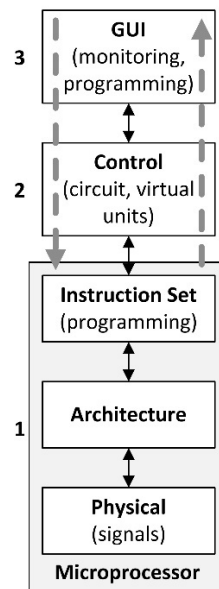


Fig. 3. Multi-layered architecture

The students program and study the microprocessor through the GUI (layer 3). A control signal or assembly instruction is generated within the GUI and the destination is the real microprocessor. Through the control layer, signal exchange is supported. The end user (e.g. student) controls the operation of the microprocessor using buttons. Microprocessor is controlled via operation signals (e.g. clock, bus state, interrupts) that are applied on the corresponding pins through the control layer (layer 2). On the other hand, the communication of the GUI with the microprocessor in terms of instruction and addresses is performed through the pins of address and data buses. Layer 1 is a solid layer because consists of the microprocessor itself. Due to that fact, the only access inside the microprocessor is achieved only via assembly instructions.

3.3 A new GUI

The developed GUI in the case of the proposed hybrid tool is more complicated as compared to the mentioned GUI in figure 2. This is because the microprocessor pins can be monitored at real time and additional signals for controlling the basic behavior can be also defined within the GUI. figure 4 shows the developed GUI of the proposed hybrid education tool.

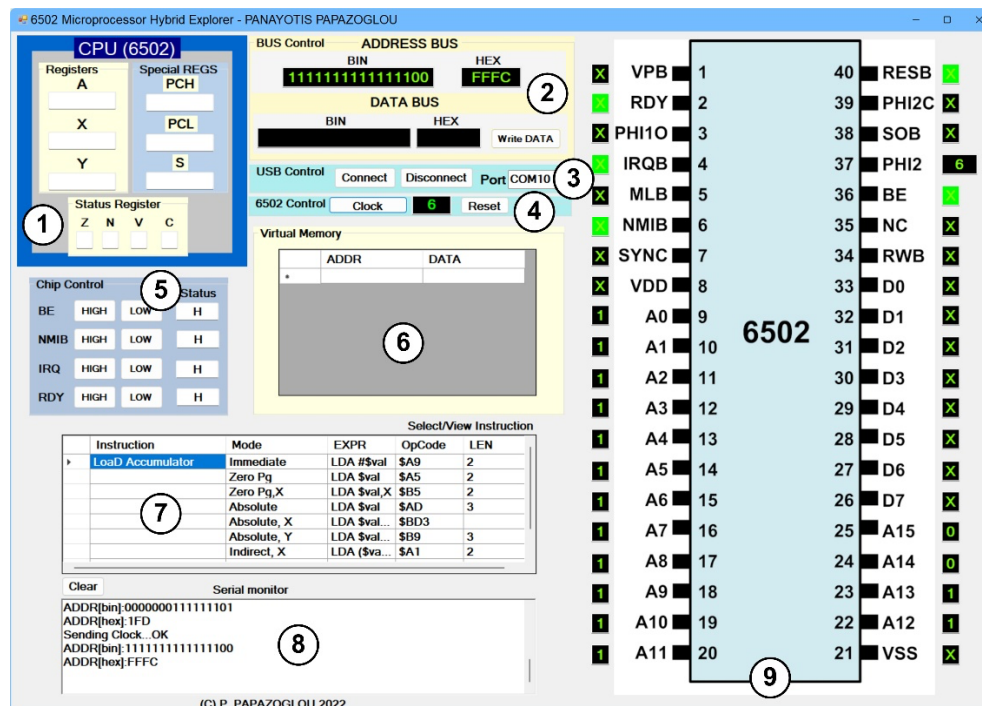


Fig. 4. The new GUI

The new GUI is organized in different areas, as follows:

- (1) CPU registers (general purpose: A,X,Y, special purpose: PC, S, and status register)
- (2) Contents of address and data bus both in binary and hexadecimal representation
- (3) USB control (port selection and connection activation). The control layer is implemented with an Arduino and thus, a USB port is needed
- (4) From this area, clock and reset signals are sent to the microprocessor pins
- (5) Critical control signals (e.g. BE-Bus Enable, IRQ-Interrupt request, RDY-Ready state) are set as HIGH or LOW by applying the corresponding signals to microprocessor pins
- (6) System virtual memory. The educational tool does not have real memory (will be explained later)
- (7) Available assembly instructions that can be chosen by the user (real instructions that are supported by the microprocessor)
- (8) USB connection monitoring for viewing data exchange between the GUI and the microprocessor
- (9) Microprocessor chip layout for monitoring pin signals at real time

The above GUI is based on the classic view of a MS which is enhanced with additional functionality due to the existence of a real microprocessor.

3.4 Comparing solutions

As mentioned before, the proposed tool unifies the advantages of two different tools in order to offer a new experience and functionality regarding the programming of a real microprocessor. Table 1 shows the basic features of the proposed hybrid tool as compared to existing educational tools.

Features	Simulators (MSs)	Trainers (MTs)	Hybrid tool
cost	free/low	high	low
complexity	low	high	low
maintenance cost	0	high	0
customization	yes	no	yes
microprocessor	hypothetical	real	real
reproduction	installation	no	easy
flexibility	yes	no	yes
visualization	yes	no	yes
GUI complexity	simple	-	simple/complex
real hardware	no	yes	yes
signal analysis	no	yes	yes
expendability	yes	no	yes
microprocessor features	fake	real	real
hardware access	no	yes	yes
external circuits	no	yes	yes
need for PC	yes	no	yes
online capability	yes	no	yes
home usage	yes	no	yes
Instruction development	yes	no	no
Instruction overhead	no	no	yes

Table 1. Comparing tools features

The proposed hybrid tool can be easily reproduced by teachers and students with a very low cost and complexity. Thus, any university class can be supported without maintenance cost. The physical implementation of the proposed tool requires a PC, an Arduino, a microprocessor and either a breadboard or a permanent PCB circuit for signal connections. The new GUI is developed in Visual Basic within the free Visual Studio Community and can be fully customized. The proposed tool is expandable within the GUI or with external circuits for experimenting with the real microprocessor. Due to the existence of a real microprocessor, a hardware point of view is possible as well as the manipulation of real signals.

3.5 Physical implementation

For implementing the proposed educational tool, three components are needed:

- a) 6502 microprocessor (in the W65C02S modern edition)
- b) Arduino MEGA 2560
- c) PC for running the GUI

Additionally, a breadboard and cables are needed for the corresponding connections or PCB for permanent connections (Figure 5).

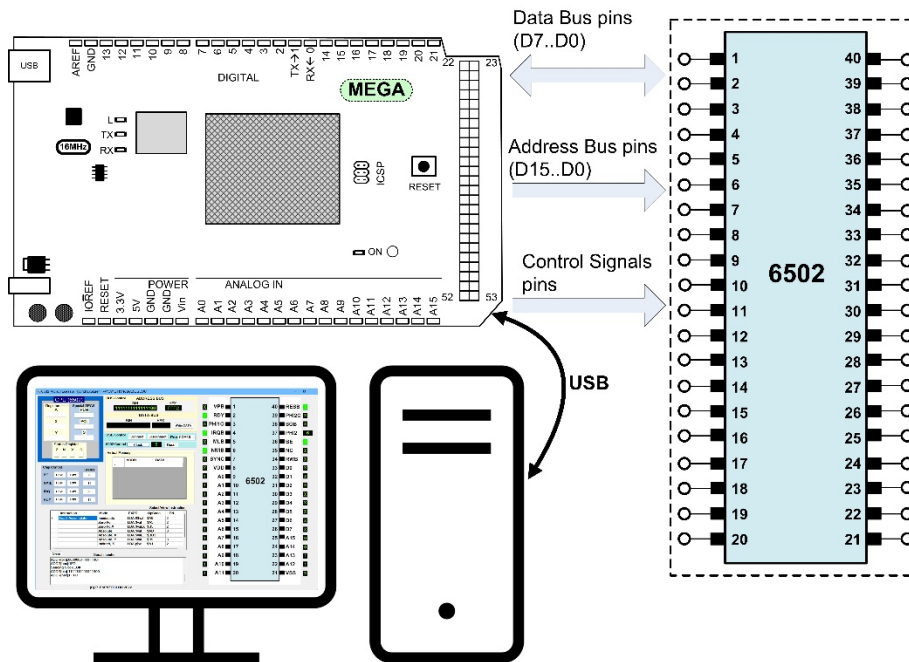


Fig. 5. Complete educational tool

Based on figure 5, there are two types of interaction:

- between Arduino and microprocessor
- between Arduino and PC

For supporting the needed interaction, two pieces of software have to be developed. The first one is for the GUI (running on PC) and the second one for reading/writing between Arduino and microprocessor pins (running on Arduino).

4. PRELIMINARY RESULTS

The developed hybrid tool so far, has been tested for basic operations.

4.1 Initializing the microprocessor

Due to real microprocessor existence, the corresponding initialization has to be performed according to manufacturer's guide as follows:

- Powering up W65C02S through Arduino
- Setting basic control signals state (e.g. BE, RDY, NMIB)
- Restarting (Reset) and execution stepping (Clock)
- Supporting the reset sequence by placing the Low and High byte of the program starting address. These bytes are requested from the addresses FFFC and FFFD respectively (based on the operation that is defined by the manufacturer)

Figure 6 shows the sequence of signals after chip reset, (addresses FFFC and FFFD as well as the result of placing a specific value on data bus as an answer to microprocessor request). The contents of figure 6 are copied from serial monitor of the PC Software GUI. Remember that serial connection, Reset and Clock signals are generated by user's actions.

```
CONNECTED
Reset Signal OK
Sending Clock...OK
ADDR[bin]:1111111111111111
ADDR[hex]:FFFF
Sending Clock...OK
ADDR[bin]:1111111111111111
ADDR[hex]:FFFF
Sending Clock...OK
ADDR[bin]:0000000111111111
ADDR[hex]:1FF
Sending Clock...OK
ADDR[bin]:0000000111111110
ADDR[hex]:1FE
Sending Clock...OK
ADDR[bin]:0000000111111101
ADDR[hex]:1FD
Sending Clock...OK
ADDR[bin]:1111111111111100
ADDR[hex]:FFFC
Sending Clock...OK
ADDR[bin]:1111111111111101
ADDR[hex]:FFFD

data=10101010
Sending Clock...OK
ADDR[bin]:1010101000000000
ADDR[hex]:AA00
```

Fig. 6. Initial sequence

Based on figure 6, the operation sequence is as follows:

- User selects serial port and activates the serial connection
- User presses Reset, and the microprocessor initialization starts
- After several clock signals (generated by user), the microprocessor reads the addresses FFFC and FFFD for finding the program starting address
- During the above sequence (reading from addresses FFFC and FFFD), the user sets the desired value on data bus as the answer from memory. In the example of figure 6, the value 10101010 (AA in HEX)

is placed as the desired value on Data Bus (the button Write Data must be pressed) and the microprocessor gets this value as memory content.

e) As shown in figure 6, the new content of address bus is 1010101000000000 and that means that the microprocessor recognizes the above value (10101010) as the real memory content for defining the High or Low byte of the program's starting address.

4.2 Executing instructions

As mentioned before, there is no real memory in the proposed tool. The expected memory data are placed directly to data bus using the connected Arduino. The desired value is defined through the GUI environment.

For example, assume that the instruction LDA #5 is stored in successive memory locations \$012C and \$012D and the microprocessor tries to read from the address \$012C. The content of \$012C is A9 (instruction opcode) and the content of \$012D is 05 respectively. For fetching instruction opcode, a memory read has to be performed and the microprocessor places the address \$012C (0000000100101100 in binary with 16bits) on address bus pins. Figure 7 shows the operation of reading two successive memory locations (\$012C and \$012D).

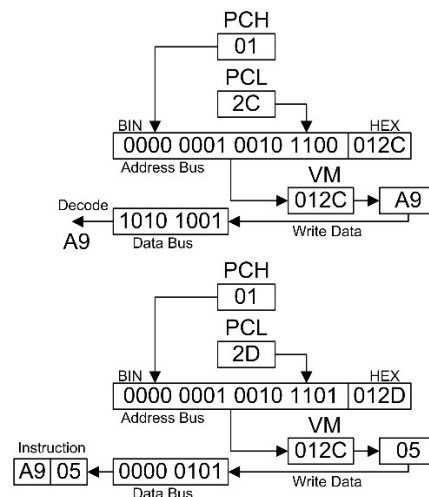


Fig. 7. Reading memory and instruction execution

The above operation is performed as follows:

- Program Counter contains the starting address (\$012C) of the instruction as a 16bit value (PCH-High byte, PCL-Low byte)
- Arduino reads the contents of address bus and places the desired answer (opcode A9) on data bus (through the GUI interface)
- Opcode A9 is now stored in Instruction Register of the microprocessor for decoding
- Program Counter contains now the next address (\$012D)
- The memory answer 05 is placed on data bus
- The instruction inside microprocessor is now completed (09 05) and the corresponding execution can be performed

4.3 Limitations

The proposed tool introduces some limitations that can be faced effectively only by programming approaches (in GUI). For example, it is not possible to develop new assembly instructions because only

the real microprocessor instructions are available. Moreover, an instruction overhead is needed for monitoring register's contents at real time because the only access to microprocessor internal architecture is achieved through the available assembly instructions. For example, when an instruction loads an integer number to a register, the only way to verify the corresponding content is to execute an additional instruction for reading that content and storing in memory. This approach introduces redundant code that is interleaved with the user code. For inserting automatically the needed redundant code, a demanding code development has to be made by the hybrid tool programmer.

5. CONCLUSIONS

In this paper, a new hybrid microprocessor technological tool has been presented. Several educational tools have been proposed in the literature but with significant limitations. Software tools (simulators) offer a rich GUI and many of them are free, but the students cannot approach the microprocessor from a real hardware point of view (e.g. signals, pins and buses). On the other hand, hardware trainers are too expensive and complicated for the students. Thus, a new tool is needed for effectively facing the mentioned limitations. The proposed tool is a low-cost solution (W65C02S costs \$10, Arduino Mega2560 costs \$20, breadboard costs \$2.5, jumper cables cost \$2.5, total cost \$35) and the corresponding connections are quite simple. Moreover, the proposed hybrid tool can be reproduced easily by teachers and students. Finally, there is more work to be done regarding the instruction execution for supporting all the microprocessor features through the PC Software GUI.

REFERENCES

- [1] B. Nova, J. C. Ferreira and A. Araújo, "Tool to support computer architecture teaching and learning," 2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE), 2013, pp. 1-8, doi: 10.1109/CISPEE.2013.6701965.
- [2] Shine V.J., Sathish P.K, "Teaching Computer Architecture Using Simulation Tools", International Journal of Computer Science and Information Technologies, Vol. 5 (2) , 2014, 1411-141
- [3] Topaloglu, Nurettin & Gürdal, Osman. (2010). A Highly Interactive PC based Simulator Tool for Teaching Microprocessor Architecture and Assembly Language Programming. Elektronika ir Elektrotechnika.
- [4] Mezei, Ivan. (2017). Cross-platform GUI for Educational Microcomputer Designed in Qt. 10.1109/EWDTS.2017.8110109.
- [5] P.W.C, Prasad & Alsadoon, Abeer & Beg, Azam & Chan, Anthony. (2015). Using simulators for teaching computer organization and architecture. Computer Applications in Engineering Education. 24. 10.1002/cae.21699.
- [6] Topaloglu, Nurettin. (2018). Comparing the Effectiveness of a Microprocessor Training System and a Simulator to Teach Computer Architecture. 34.
- [7] P.M.Papazoglou, A Hybrid Simulation Platform for Learning Microprocessors, Computer Applications in Engineering Education, 10.1002/cae.21921, (pp 655-674) WILEY, 2018