DEVELOPING A PROGRAMMABLE ELECTRONIC CIRCUIT FOR THE CONTROLLED IGNITION OF EXPLOSIVE MIXTURES

ZOLTAN VASS[1], MARIUS ȘUVAR[2], BOGDAN ȘIMON-MARINICĂ[3], LAURENȚIU MUNTEAN[4]

Abstract: The research performed in the domain of explosive gaseous mixtures, in particular of hydrogen-oxygen gaseous mixtures, is gaining ground due to their applications in various fields, such as power generation, propulsion systems and safety engineering. Due to the current realities of environmental and political considerations it is necessary to transition away from conventional carbon based sources of energy, to more clean sources of energy. In order to accomplish this goal, research is needed in the field of clean energy, in particular in the research of hydrogen. These research activities are performed under a strict laboratory environment where the conditions and requirements of the ignition of these gaseous mixtures are critical. In this context, the design and development of programmable electronic control circuit for the controlled ignition of explosive gaseous mixtures represents a significant advance. The objective is to create a system that not only is reliable, but also cost-effective and simple to use. In this paper we present a design and development of a digital electronic circuit.

Key words: hydrogen, mixture, ignition, control circuit.

1. INTRODUCTION

With the beginning of the first industrial revolution, which necessitated the rapid extraction and burning of fossil fuels, which in turn lead to the release of greenhouse gases into the atmosphere in quantities never seen before, global temperatures begin to rise, leading to increase in a number of negative effects, such as drought, high intensity storms, rise of the sea level, melting of glaciers. In order to mitigate these negative effects of global warming it is necessary to transition the economy from the consumption of fossil fuels to other, more cleaner forms of energy.

Energy is the basic element for human survival and human consumption of energy has increased dramatically due to the rapid development and progress of

[1] *Sc. Res., Ph.D.Eng., INCD-INSEMEX Petroșani, vass.zoltan@insemex.ro*
[2] *Sc. Res., Ph.D.Eng., INCD-INSEMEX Petroșani, marius.suvar@insemex.ro*
[3] *Sc. Res., Ph.D.Eng., INCD-INSEMEX Petroșani, bogdan.simon@insemex.ro*
[4] *Sc. Res., Ph.D.Eng., INCD-INSEMEX Petroșani, laurentiu.muntean@insemex.ro*

society. In particular, the consumption of large amounts of non renewable fossil energy has aggravated environmental degradation and energy crisis. Therefore, the development and construction of a clean, efficient, and renew-able new energy carrier is of great significance to achieving the goal of carbon neutrality. As a globally recognized clean energy carrier, hydrogen energy has the advantages of wide sources, high energy density, and stable properties at room temperature [1].

In the current political, social and political climate, where the governments of the countries of the world promote clean energy sources, hydrogen is one of the alternatives proposed to replace fossil fuels. Due to the different thermal, chemical properties of hydrogen, compared to fossil fuels, it is necessary to conduct research to determine the behavior and potential risks the use of hydrogen presents in the ordinary activities.

In the research on gas explosions, the emphasis has been and continues to be primarily on physical experiments conducted on various scaled-down models. Building models at actual size is often a resource-intensive task in terms of materials, time, and human resources. The rapid advancement of computational techniques has allowed, among other things, the transfer of gas explosion research into the virtual environment. For validating computerized simulations of this kind, physical experiments and specialized literature are still considered fundamental. However, one of the challenges posed by the virtualization process is the limitation of conducting simulations in fully or partially enclosed spaces, under initially imposed conditions, without the possibility of dynamically modifying these conditions based on the development of over-pressures generated by the virtual explosion. This paper details a computerized experiment where the boundary conditions were successfully transformed into predefined pressure threshold surfaces, transitioning from rigid surfaces to surfaces capable of releasing the over-pressures developed in fully or partially enclosed spaces. This approach aligns the results of these simulations with the real dynamic effects of gas explosion events [1].

The dangers associated with hydrogen mainly come from its wide flammability range, it's extremely fast burning rate (having much more aggressive explosive properties than methane gas) and the considerable amount of energy released when it burns or explode. The development of new applications that use hydrogen as clean energy is constantly increasing, thus, hydrogen can be used on a large scale. This paper presents the use of computational fluid dynamics techniques, regarding the linear propagation of the explosion of air and hydrogen mixture in closed spaces, the main purpose being the determination of the over-pressures to which a test stand of this processes is subjected [2].

These research activities are performed under a strict laboratory environment where the conditions and requirements of the ignition of these gaseous mixtures are critical. In this context, the design and development of programmable electronic control circuit for the controlled ignition of explosive gaseous mixtures represents a significant advance. The objective is to create a system that not only is reliable, but also cost-effective and simple to use [3].

Hydrogen will become a key player in transitioning toward a net-zero energy system. However, a clear pathway toward a unified European hydrogen infrastructure to support the rapid scale up of hydrogen production is still under discussion. This study explores plausible pathways using a fully sector-coupled energy system model [4].

Hydrogen together with methane, is one of the most common gaseous fuels that also exist in nature as the main part of the natural gas, the flammable part of bio-gas or as part of the reaction products from biomass pyrolysis. In this respect, the bio-gas and biomass installations are always subjected to explosion hazards due to methane. Simple methods for evaluating the explosion hazards are of great importance, at least in the preliminary stage [5].

2. COMPONENT ELEMENTS OF THE CIRCUIT

Due to the type of laboratory experiment performed, which consists of igniting gaseous mixtures of hydrogen and oxygen, it is necessary to employ the use of an electronic circuit that can provide a controlled and reliable way of igniting of these gaseous mixtures [6].

This electronic device is a digital programmable electronic circuit, with a custom instruction set.

The circuit is composed of multiple modules. These communicate with each other using a central bus.

Table 1. ***Component elements of the circuit***

1	Program memory
2	Instruction Decoder
3	Arithmetic and Logic Unit
4	Register file
5	Program Counter
6	IO Module

The program memory holds the instructions than will be executed by the electronic circuit.

There are two types of memory available. volatile memory which will lose all data when the memory is disconnected from the power source, and non-volatile memory, which will preserve the data stored in it even when is disconnected from the power source.

The instruction decoder is responsible with decomposing the instructions read from program memory and generating control signals for enable or disable different components of the circuit. This is necessary in order to avoid conflicting scenarios that could rise during the functioning of the electronic circuit.

Every computer has a word length that is characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as buses), for example, a computer whose registers and buses can store and transfer 8 to 32 bits. The characteristic 8-bit field is referred to as a byte. Each operation that the processor can perform is identified by a unique binary number known as an instruction code or operation code (OP code). An 8-bit word used as an instruction code can distinguish among 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. In the first, it transmits the address in its program counter to the memory. In the second, the memory returns the addressed byte to the processor. The CPU stores this instruction byte in a register known as the instruction register and uses it to direct activities during the remainder of the instruction execution. The mechanism by which the processor translates an instruction code into specific processing actions requires more elaboration than we can afford here. The data stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined coincidentally with selected timing pulses, to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the instruction decoder and by the associated control circuitry [7].

In computing, an arithmetic logic unit is a combinatorial digital circuit that performs arithmetic and bit-wise operations on integer binary numbers. This is in contrast to a floating-point unit, which operates on floating point numbers. It is a fundamental building block of many types of computing circuits, including the central processing unit of computers, floating-point unit, and graphics processing units.

The inputs to an Arithmetic and Logic Unit are the data to be operated on, called operands, and a code indicating the operation to be performed; the Arithmetic and Logic Unit's output is the result of the performed operation. In many designs, the Arithmetic and Logic Unit also has status inputs or outputs, or both, which convey information about a previous operation or the current operation, respectively, between the Arithmetic and Logic Unit and external status registers [8].

The arithmetic and logic unit is responsible with performing logic and arithmetic operations such as logic operations and arithmetic operations. Taking into account the purpose and the intended use of this electronic circuit, the implementation of logic operations was deemed as not necessary.

The register file is the component part of the electronic circuit that consists of a number of fast memory modules, which are used to store data. The data stored in these memory modules can be accessed by using control signals that allow us to use one or multiple registers simultaneously.

A register file is an array of processor registers in a central processing unit. The instruction set architecture of a central processing unit will almost always define a set of registers which are used to stage data between memory and the functional units on the chip. The register file is part of the architecture and visible to the programmer.

In simpler central processing units, these architectural registers correspond one-for-one to the entries in a physical register file within the central processing unit. More complicated processors use register renaming, so that the mapping of which physical entry stores a particular architectural register changes dynamically during execution.

Modern integrated circuit-based register files are usually implemented by way of fast static RAMs with multiple ports. Such RAM memories are distinguished by having dedicated read and write ports, whereas ordinary multi-ported S-RAM memory will usually read and write through the same ports. Register banking is the method of using a single name to access multiple different physical registers depending on the operating mode [9].

The program counter is a component part of the electronic circuit which ensures that the instructions are loaded from program memory into the decoder sequentially. This module is also responsible for the execution of jump instructions, using a number of control signals provided by the decoder module.

The output module is used to allow for the circuit to communicate with other devices. The output module, is designed in such a way that allow multiple ignition devices to be controlled simultaneously.

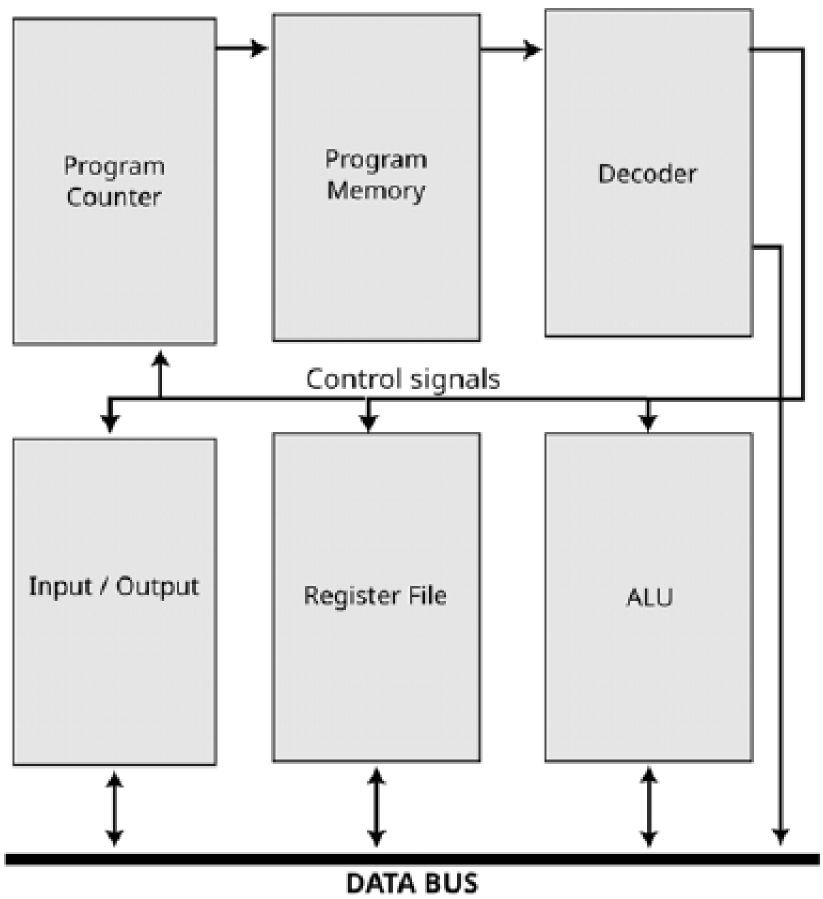

Fig. 1. Block diagram of the electronic circuit

The Input / Output module of can be used to control a number if external devices, such as:

- Data acquisition systems: i/o module can be used in data acquisition systems to acquire and process data from various sensors and input devices. the i/o processor can handle high-speed data transfer and perform real-time processing of the acquired data.

- Industrial control systems: i/o module can be used in industrial control systems to interface with various control devices and sensors. the i/o processor can provide precise timing and control signals, and can also perform local processing of the input data.
- Multimedia applications: i/o module can be used in multimedia applications to handle the input and output of multimedia data, such as audio and video. the i/o processor can perform real-time processing of multimedia data, including decoding, encoding, and compression.
- Network communication systems: i/o module can be used in network communication systems to handle the input and output of data packets. the i/o processor can perform packet routing, filtering, and processing, and can also perform encryption and decryption of the data.
- Storage systems: i/o processors can be used in storage systems to handle the input and output of data to and from storage devices. the i/o processor can handle high-speed data transfer and perform data caching and pre-fetching operations. [10]

3. INSTRUCTION SET

In order to be useful, a programmable electronic circuit must have a set of instructions that will allow to perform useful tasks.

The number of instructions a processor is capable of executing, varies depending on the environment in which this processor will be used. Therefore if the programmable electronic circuit is used in a special circumstance that require only a limited functionality, it is not necessary to implement a full set of instructions. However if the programmable circuit is used in a more complex environment, implementing a full set of instructions may be necessary.

All processors carry out their require operations by executing sequences of instructions. Each instruction defines a simple operation, for example, simple Arithmetic and Logic Unit operation, data access to the memory system, program branch operation, etc.

For the processor, it takes instructions in form of binary code and decodes them in internal hardware (instruction decoder), then passes on the information about the decoded instruction to the execution stage. In simple processor designs, for minimum the following types of instructions are required:

- Data processing(arithmetic and / or logic operations);
- Memory access operations(read from memory, write to memory);
- Program flow control instructions(branches, conditional jump instructions, function calls);
- Access to registers;
- Other operations [11].

In our case, the programmable circuit is used in a laboratory setup, to control the ignition of explosive gaseous mixtures, therefore it is not necessary for us to implement a large set of instructions.

In the table below are listed all the instructions that our programmable circuit can execute.

Table 2. ***List of instructions of the programmable circuit***

Opcode	Instruction name:	Instruction binary map
00000	NOP	0000000000000000
11111	HLT	-----------ooooo
00001	JR	mmmmmmmm---ooooo
00010	JA	mmmmmmmm---ooooo
00011	JC	mmmmmmmmfffooooo
00011	JZ	mmmmmmmmfffooooo
00011	JE	mmmmmmmmfffooooo
00011	JL	mmmmmmmmfffooooo
00011	JG	mmmmmmmmfffooooo
01001	ADD	--bbbaaadddooooo
11001	SUB	--bbbaaadddooooo
01011	INC	-----aaadddooooo
11011	DEC	-----aaadddooooo
11000	CMP	--bbbaaa---ooooo
10110	MOV	-----aaadddooooo
10101	LDI	iiiiiiiidddooooo

Legend:
o – opcode
m – memory address
a – first operand register address
b – second operand register address
d – destination register address
i – immediate value
f – function select

3.1. NOP instruction

The purpose NOP instruction its to allow for the processor to pause execution for a limited amount of time. This time can be controlled by executing one or more NOP instructions sequentially.

Table 3. ***Binary representation of the NOP instruction***

15 5	4				0
	0	0	0	0	0

The fields from 0 to 4 is used to encode the binary representation of the NOP instruction. As it can be observed from the table above, this instruction is encoded with zero values.

The rest of the fields from 5 to 15 are unused.

3.2. HLT instruction

The The purpose of the HLT (halt) instruction is to terminate execution of the program. This instruction is very important. Without this instruction the programmable circuit will continue to run out of control, which could lead to errors.

Table 4. ***Binary representation of the HLT instruction***

15 5	4				0
	1	1	1	1	1

From the table above, the binary encoding of the HLT instruction can be observed.

The fields ranging from 0 to 4 is used to encode the binary representation of the HLT instruction.

The fields from 5 to 15 are unused.

3.3. JUMP instructions

Our programmable circuit is capable to execute special instructions that allow it to load instructions from the program memory from any arbitrary memory address. These instructions can be used to provide a greater flexibility to our programmable circuit. Our circuit that we develop is capable of a number of seven jump instruction: relative jump, absolute jump and five conditional jump instructions.

In the table below are presented the binary representation of the jump instructions.

Table 5. ***Binary representation of the relative and absolute jump instruction***

15							8	7		5	4				0
m	m	m	m	m	m	m	m	f	f	f	o	o	o	o	o

From From table 5 above can be observed the binary encoding of the jump instructions.

The fields ranging from 0 to 4 is reserved for encoding of the jump instruction.

The fields ranging from 5 to 7 is reserved to encode the type of the jump instruction.

The fiends ranging from 8 to 15 is reserved to encode the memory location of the jump instruction.

It can be observed from the above table that in fact the HLT instruction is a special case of the absolute jump instruction.

3.4. Arithmetic instructions

Arithmetic instructions are an important part of the functioning of the programmable circuit that we design. These instruction allow us to perform simple arithmetic instructions such ad adding or subtracting two values, incrementing or decrementing a value, and store the result in a specified registry location.

Table 6. ***Binary representation of the arithmetic instructions***

15		13		11	10		8	7		5	4				0
-	-	b	b	b	a	a	a	d	d	d	o	o	o	o	o

In table above we can observe the binary encoding of the addition and subtract instructions.

The fields ranging from 0 to 4 is used to encode the binary code of the instruction.

The fields ranging from 5 to 7 id used to encode the address of the registry where the result of the instruction will be stored.

The fields ranging from 8 to 10 is used to encode the address of the registry where the value of the first operand is stored.

The fields ranging from 11 to 13 is used to encode the address of the registry where the value of the second operand is stored.

Table 7. ***Binary representation of the arithmetic instructions***

15	10		8	7		5	4				0
	a	a	a	d	d	d	o	o	o	o	o

In table above we can observe the binary encoding of the increment and decrement instructions.

The fields ranging from 0 to 4 is used to encode the binary code of the instruction.

The fields ranging from 5 to 7 id used to encode the address of the registry where the result of the instruction will be stored.

The fields ranging from 8 to 10 is used to encode the address of the registry where the value of the operand is stored.

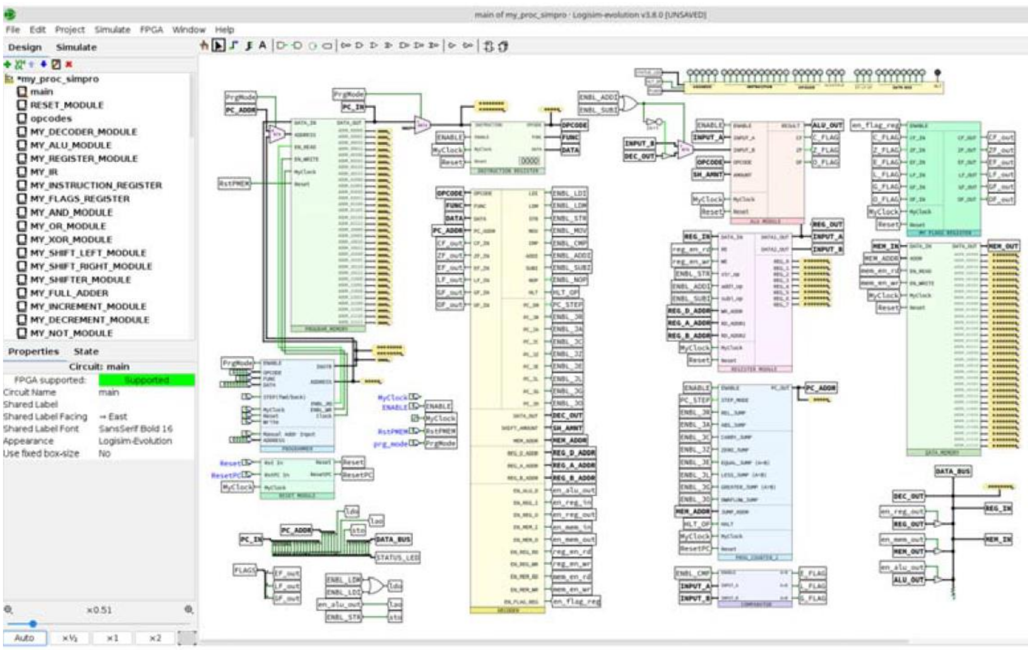

Fig. 2. Screen capture of the simulation of the electronic circuit

This work was carried out through the"Nucleu" Program within the National Plan for Research, Development and Innovation 2022-2027, with the support of the Romanian Ministry of Research, Innovation and Digitalization, project no PN-23 32 02 01., title: Development of the knowledge level of the explosion parameters of gaseous and hybrid mixtures based on hydrogen, involved in the processes of obtaining and storage in order to reduce the explosion risk.

4. CONCLUSIONS

The development of a programmable electronic circuit, for the ignition of explosive gaseous mixtures, can lead to an increased accuracy of the experiments, and also to the improvement on the field of safety. Using a programmable electronic circuit allow us to use ignition patterns that are unavailable or are difficult to implement using traditional technologies.

REFERENCES

[1]. Long J.J., Wu H-C., Liu Y-T., Ding Y-Y., Yao Q-L., Metin O., Lu Z-H., *Hydrogen production from chemical hydrogen storage materials over copper-based catalysts*, Youke Publishing Co., Ltd., 2024.

[2]. Florea G., Petrilean D., *Computational Simulation of Methane Ignition in Enclosed Spaces.* Mining Revue, 30(1), pp. 58-67., 2024.

[3]. Șimon-Marinică A.B., Manea F., Vass Z., Păsculescu V.M., *Virtual experiments about hydrogen explosions with linear propagation,* MATEC Web of Conferences 2024.

[4]. Șimon-Marinică A.B., Manea F., Gaman G.A, s.a. *Development of a Programmable System Used for the Preparation of a Mixture of Flammable/Explosive Gases.* Automation 5, pp. 246 -258, 2024.

[5]. Kountouris I., B Ramstoft R., Theis Madsen T., Gea-Bermúdez J., s.a. *A unified European hydrogen infrastructure planning to support the r apid scale-up of hydrogen production,* Nature Communications, vol. 15, 2024.

[6]. Prodan M., Ghicioi E., Laszlo R., s.a. *Experimentaland Numerical Study of Ignition and Flame Propagation for Methane–Air Mixtures in Small Vessels,* MDPI, 2021.

[7]. Samuel C. L., *Encyclopedia of Physical Science and Technology* ,Third Edition, 2003.

[8]. https://en.wikipedia.org/wiki/Arithmetic_logic_unit

[9]. https://en.wikipedia.org/wiki/Register_file

[10]. https://www.geeksforgeeks.org/introduction-of-input-output-processor/

[11]. Yiu J., *The Definitive Guide to Arm® Cortex®-M0 and Cortex-M0+ Processors* (Second Edition), 2015.

[12]. Malvino A. P., Ph.D. Jerald A. Brown – Digital Computer Electronics – 3rd Edition, 1994.

This article was reviewed and accepted for presentation and publication within the 11th edition of the International Multidisciplinary Symposium "UNIVERSITARIA SIMPRO 2024".