Designing Digital Circuits in Multi-Valued Logic

Alessandro Simonetta[#], Maria Cristina Paoletti

[#] *Department of Enterprise Engineering, University of Rome "Tor Vergata", Via del Politecnico, 1, Rome, 00133, Italy*
E-mail: alessandro.simonetta@gmail.com

***Abstract*— In the last few decades we have witnessed an increase in CPU performance, which has been made possible thanks to the increase in the clock frequency and the increase in the number of transistors in the unit of space. In the last few years, however, we reached the limit for the clock and for the miniaturization of the transistor grid. Beyond this growth new problems arose such as the disposal of the produced heat and the minimum distance to be respected between elements for the electrical signals transfer. So the chip makers, to further increase the processing power of the processors, started to insert more cores on the same chip. The presence of several cores undoubtedly improves performance and improves consumption, but the ability to transfer data between cores and components remains limited by the number of pins of the cores themselves. Furthermore, it is necessary to manage the synchronization between cores during the access to common resources and all those multi-core architectures typical problems. This article provides a different approach to improve the computing capacity of the CPUs that is based on the extension of the binary system in a multi-value coding system or, commonly, called MVL. Although this direction has already been explored, the idea behind the study is in the representation of the generic function in the MVL domain. This representation has a link to the binary system and a surprisingly greater simplicity of the corresponding digital circuits (combinatorial and sequential). A different mathematical approach is thus provided for the realization of the multivalue logic gates. This could enable the use of different data encoding systems no longer linked to the voltage value of a signal but to other physical quantities as it happens at present, for example, in the world of telecommunications.**

***Keywords*— multi-valued logic; circuit design; computer architecture; fuzzy system.**

I. INTRODUCTION

The chip makers have always looked to improve performance by trying to make increasingly powerful and faster CPUs by focusing on increasing the clock frequency and inserting a greater number of transistors in the unit of space. In recent decades, the number of CPU transistors followed the famous Moore law. Unfortunately, due to the level of miniaturization achieved, it was no longer possible to increase neighter the density of the transistors, nor the clock frequency (another factor that influences the computing power), because new problems arose, in addition to the ability to operate at inertial distances, linked to the signal transfer speed between components and also to the heat produced by the components overheating (see [1]).

As the increase in the density of transistors on the surface had reached the limit, some have hypothesized to increase the number through the thickening of the die. However, having a greater thickness of the die can lead to a significant increase in costs, for this reason some studies have shown that the number of transistors can be increased by using a double layer of the die to form a crystal grid in whose nodes [transistors are located. This approach, duplicates the number of transistors and makes the die thicker [2].

However, the strategy adopted by the chipmakers has been to improve the computing power by constructing CPUs with multiple cores within the same chip, enabling processing parallelism. This approach undoubtedly improves performance but introduces new problems because that processing is distributed on the cores. These cores must be synchronized to access common resources or collaborate to a common goal. Moreover, the management of interrupts is more complex than a single-core. Finally, the limit of the transfer capacity of the cores cannot be exceeded, because it depends on the number of connections and the clock frequency. Power management also becomes a crucial factor when building high-performance architectures, as the articles [3], [4] and [5] demonstrates.

In this article we will discuss a different approach based on the possibility of building a high performance computer architecture by making digital components that work intrinsically in multivalue logic (MVL). The need to build efficient machines that worked in MVL, with low consumption was considered in [6].

A ternary computer hypothesis has been historically treated in [7] where the author observed that if the circuit complexity depended roughly on the product of the base size used (R) multiplied by the number of digits of the maximum representable number (N), the economically best basis was *e*=2,718. The article pointed out that if it were possible to build components that when increasing the representation base (R) size complexity and cost don't grow, the best choice would be to adopt the largest possible base.

The idea of designing circuits that work in MVL has recently been treated from the canonical point of view, using post-order algebras of degree greater than or equal to two, also in [8]. Some studies showed that the methods used in the Boolean algebra, such as the Quine Mc-Cluskey method shown in [9], are applicable. However, the idea that inspired this work is the scalability of the solution with respect to the chosen base, the simple realization of the circuits and the maintenance of a close relationship with the binary system. The ability to build any circuit in MVL will allow to build processors, memories and I/O devices able to operate, with the same number of connections, at a greater throughput compared to the binary case. The ternary logic was the first studied extension of the binary algebra ([10] and [11]), but also inspired the realization of processors working on the base 3 [12]. Also the quaternary logic, being power of two, inspired many research works ([13] and [14]). In the literature we also find valuable contributions on MVL as [15] and [16], also from the point of view of the verification of hardware circuits [17].

However, with the present work we will demonstrate how it is possible to define a reduced set of mathematical operators that are able to perform any function in the chosen domain independently from the base, similar to what happens for universal operators (NAND and NOR) in the case of Boolean algebra. The proposed idea is based on the use of one-digit arithmetic operations (multiplication and sum) together with the functions reported in the binary domain (the selectors). Multivalue operators will be described in Session II from the external point of view, as if they were black boxes, without considering the internal functionality or the modes of transferring the signals.

Research and technological innovation will provide, in the near future, the best answers for the realization of these components. Session III is organized in two parts: the first concerns the process of creating an example of combinational circuit (half-adder); the second one is the construction of a memory element for MVL information. Section IV gives the conclusion and oulines the future research.

II. Material and Method

Without losing generality we can consider the algebra in base 3 and then we can extend the operators to any domain with n-values. Although we have extended the discrete domain by a single value, for example a word of only 10 digits passes from 1024 combinations in the binary number system to about 59k in the ternary one.

A. Ternary algebra

Consider a T domain consisting of the three values {0,1,2}. On this domain we can define unary functions $F(I_0)$: $T \rightarrow T$, with $I_0 \in T$; binary functions $F(I_1, I_0)$: $T \times T \rightarrow T$, with $I_0 \in T$, $I_1 \in T$; and, generically, functions with p operands $F(I_{p-1}, I_{p-2},...,I_1, I_0)$: $T \times ... \times T \rightarrow T$, with $I_0 \in T$,..., $I_{p-1} \in T$.

So we can imagine a function like a black box that receives input values I_0, I_1,..., I_{p-2}, I_{p-1} and returns an output U that represents the value assumed by the function at the inputs (combinatorial circuit).

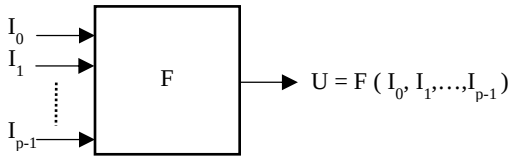

Fig. 1 Example of function as a black box

With p inputs we have $x=3^p$ combinations of the inputs that originate 3^x different functions. In a generic algebra with n values we will have $x=n^p$ e n^x different functions.

1) Unary functions: in the context of the n^n possible unary functions (27 in the ternary case) we consider n functions, which reduce the n-ario domain into the binary one. In particular we will call them selection functions, or selectors, and we will indicate them with the letter S.

In an algebra with n values there are n selectors each one for any symbol of the domain. If $c \in T$ the selection function $S_c(I)$ answers the value 1 when in input (I) the value c is present and zero in the other cases.

In the ternary case the functions S_0, S_1 and S_2 are:

TABLE I
SELECTOR FUNCTIONS (N=3)

I	$S_1(I)$	$S_1(I)$	$S_2(I)$
0	1	0	0
1	0	1	0
2	0	0	1

2) Binary functions: the functions with two values in the ternary domain are 3^9 (19.683) and can be described by a quintuple of functions: the three selectors plus two functions (op_1 e op_2). Entering in the value table the input selectors (S_0, S_1 and S_2) and the main arithmetic functions: product, sum, minimum and maximum:

TABLE II
SOME BINARY FUNCTIONS (N=3)

i	I_1	I_0	$S_0(I_1)$	$S_1(I_1)$	$S_2(I_1)$	$S_0(I_0)$	$S_1(I_0)$	$S_2(I_0)$	$I_1 \cdot I_0$	I_1+I_0	min (I_1,I_0)	max (I_1, I_0)
0	0	0	1	0	0	1	0	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	1	0	1
2	0	2	1	0	0	0	0	1	0	2	0	2
3	1	0	0	1	0	1	0	0	0	1	1	1
4	1	1	0	1	0	0	1	0	1	2	1	1
5	1	2	0	1	0	0	0	1	2	0	1	2
6	2	0	0	0	1	1	0	0	0	2	0	2
7	2	1	0	0	1	0	1	0	2	0	1	2
8	2	2	0	0	1	0	0	1	1	1	2	2

Any function F with two values in the ternary domain can be written as a linear combination of the selectors of the input variables in the following way:

$$F = [\, k_0\ op_1\ S_0(I_1)\ op_1\ S_0(I_0)\,]\ op_2\ [\, k_1\ op_1\ S_0(I_1)\ op_1\ S_1(I_0)\,]\ op_2 \ldots op_2\ [\, k_m\ op_1\ S_2(I_1)\ op_1\ S_2(I_0)\,] \quad (1)$$

We want to create a normal form that, similarly to the case of the min-term or max-term of Boolean algebra, considers groups of selectors of the input variables, modulated by the constant corresponding to the row, joined by an aggregation fuction .

The properties required for the two operators op_1 e op_2 are:

$$X\ op_1\ 0 = 0 \quad (2)$$

$$X\ op_1\ 1 = X \quad (3)$$

$$X\ op_2\ 0 = X \quad (4)$$

Although there are various functions that satisfy the properties (2) and (3), restricting the field of interest to the four arithmetic operations above, a possible candidate for op_1 that satisfies the properties (2) and (3) is the multiplication operation.

TABLE III
CANDIDATE FUNCTIONS FOR OP_1

I_1	I_0	$\max(I_1,I_0)$	I_1+I_0	$\min(I_1,I_0)$	$I_1 \cdot I_0$	op_1
0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	2	2	2	0	0	0
1	0	1	1	1	0	0
1	1	1	2	1	1	1
1	2	2	0	1	2	2
2	0	2	2	0	0	0
2	1	2	0	1	2	2
2	2	2	1	2	1	X

With regard to the second op_2 operator, similarly to what was done for op_1, various functions with the propriety (4) can be used; in our subset of interest: the arithmetic sum and the maximum.

TABLE IV
CANDIDATE FUNCTIONS FOR OP_2

I_1	I_0	$\max(I_1,I_0)$	I_1+I_0	$\min(I_1,I_0)$	$I_1 \cdot I_0$	op_2
0	0	0	0	0	0	0
0	1	1	1	0	0	1
0	2	2	2	0	0	2
1	0	1	1	1	0	1
1	1	1	2	1	1	X
1	2	2	0	1	2	X
2	0	2	2	0	0	2
2	1	2	0	1	2	X
2	2	2	1	2	1	X

Summarizing, in the ternary number system we can represent any two-input function (F) using its description of the truth table. The method is similar to the binary case: we have to consider all possible combination of the input variables and for each of them consider the corrisponding value of the function.

TABLE V
TRUTH TABLE OF A GENERIC FUNCTION F

i	I_1	I_0	F
0	0	0	k_0
1	0	1	k_1
2	0	2	k_2
3	1	0	k_3
4	1	1	k_4
5	1	2	k_5
6	2	0	k_6
7	2	1	k_7
8	2	2	k_8

we can write that F is calculated as the union of the nine exclusive and not overlaped cases:

$$F = k_0 \cdot S_0(I_1) \cdot S_0(I_0) + k_1 \cdot S_0(I_1) \cdot S_1(I_0) + \ldots + k_8 \cdot S_2(I_1) \cdot S_2(I_0) \quad (5)$$

that is:

$$f(I_1, I_0) = \sum_{i=0}^{8} k_i \cdot S_{c_0(i)}(I_0) \cdot S_{c_1(i)}(I_1) \quad (6)$$

where:

- k_i is the value assumed in the row corresponding to the number *i*, with $k_i \in \{0,1,2\}$;
- $c_j(i)$ is the j-th digit of the number i, represented in base n = 3, with $j \in \{0,1\}$:

$$i = \sum_{j=0}^{p-1} n^j \cdot c_j = 3^1 \cdot c_1(i) + 3^0 \cdot c_0(i) \quad (7)$$

TABLE VI
VALUES OF COEFFICENTS C_0 AND C_1

i	$c_0(i)$	$c_1(i)$
0	0	0
1	0	1
2	0	2
3	1	0
4	1	1
5	1	2
6	2	0
7	2	1
8	2	2

- $S_{c_j(i)}(I_j)$ is the selector of the $c_j(i)$ value applied to the operand I_j with $j \in \{0,1\}$.

The proof of the validity of the formula is simple: only one group of selectors can obtain the value 1 at a time, as only one configuration of the inputs is possible, being discrete. The other groups will get value 0.

The group of selectors corresponding to the input configuration will be multiplied by the related value k_i (property (1)) which, added to the others with null value groups (property (2)), will be returned as output (property (3)).

B. N-ary algebra

The interesting thing is that we can go further by generalizing the representation base (n) and describing with

the same method any function (F) with a predefined number of operands (p) within the set of n^x possibile functions, with $x=n^p$. Also in this case we can write the table of values:

TABLE VII
TRUTH TABLE OF A GENERIC FUNCTION F IN GENERAL DOMINION

i_n	I_{p-1}	I_{p-2}	...	I_1	I_0	F
0	0	0	0	0	0	k_0
1	0	0	0	0	1	k_1
2	0	0	0	0	2	k_2
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n-1	0	0	0	0	n-1	k_{n-1}
n	0	0	0	1	0	k_n
n+1	0	0	0	1	1	k_{n+1}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n^p-1	n-1	n-1	n-1	n-1	n-1	k_{n^p-1}

In a similar way to what has already been seen in the case n = 3, it is possible to represent F according to the inputs I_0, I_1, ...,I_{p-1}:

$$F(I_{p-1},\dots,I_1,I_0)=\sum_{i=0}^{n^p-1} k_i \cdot \prod_{j=0}^{p-1} S_{c_j(i)}(I_j) \qquad (8)$$

where:

- k_i is the value assumed in the line corresponding to the number *i*, with $k_i \in \{0,1,...,n-1\}$;
- $c_j(i)$ is the *j*-th digit of the number *i*, represented in the base *n* with $j \in \{0,1, ..., p-1\}$;
- $S_{c_j(i)}(I_j)$ is the $c_j(i)$ value selector applied to the I_j operand with $j \in \{0,1,...,p-1\}$.

III. RESULT AND DISCUSSION

A. The realization process of a combinational circuit in MVL

In this section we will show the algorithm to design the multivalue circuit corresponding to a generic function in the multi-value domain. The proposed algorithm is based on four sequential steps:

1) building the truth table that describes exhaustively the function to be implemented through all the possible combinations of the inputs (n^p, logic with *n* values and function with *p* operands),

2) for each non-zero element of the column that describes the function in the truth table, multiply the value for the row selector group,

3) the function is given by the sum of the groups of terms identified in point 2).

To simplifity the use of the algorithm, and in analogy with [8] in order to grasp the differences between different representation systems, we will implement the two-digit half-hadder circuit (I_1 and I_2) in base 4.

Initially we can consider the circuit as a black box in which we have 2 inputs and 2 outputs:

Fig. 2 Half-adder analyzed as a black box

The behavior of the circuit is described exhaustively by the truth table, in this case with p = 2 inputs we will have 4^2 = 16 combinations (ie the numbers from 0 to 15, column *i*):

TABLE VIII
HALF-ADDER TRUTH TABLE

i	I_1	I_0	f_1	f_0
0	0	0	0	0
1	0	1	0	1
2	0	2	0	2
3	0	3	0	3
4	1	0	0	1
5	1	1	0	2
6	1	2	0	3
7	1	3	1	0
8	2	0	0	2
9	2	1	0	3
10	2	2	1	0
11	2	3	1	1
12	3	0	0	3
13	3	1	1	0
14	3	2	1	1
15	3	3	1	2

According to the illustrated methodology, the output functions are:

$$\begin{aligned} f_1 = {} & 1\cdot S_1(I_1)\cdot S_3(I_0) + 1\cdot S_2(I_1)\cdot S_2(I_0) + 1\cdot S_2(I_1)\cdot S_3(I_0) + \\ & 1\cdot S_3(I_1)\cdot S_1(I_0) + 1\cdot S_3(I_1)\cdot S_2(I_0) + 1\cdot S_3(I_1)\cdot S_3(I_0) = \\ = {} & S_1(I_1)\cdot S_3(I_0) + S_2(I_1)\cdot S_2(I_0) + S_2(I_1)\cdot S_3(I_0) + \\ & S_3(I_1)\cdot S_1(I_0) + S_3(I_1)\cdot S_2(I_0) + S_3(I_1)\cdot S_3(I_0) \end{aligned} \qquad (9)$$

$$\begin{aligned} f_0 = {} & 1\cdot S_0(I_1)\cdot S_1(I_0) + 2\cdot S_0(I_1)\cdot S_2(I_0) + 3\cdot S_0(I_1)\cdot S_3(I_0) + \\ & 1\cdot S_1(I_1)\cdot S_0(I_0) + 2\cdot S_1(I_1)\cdot S_1(I_0) + 3\cdot S_1(I_1)\cdot S_2(I_0) + \\ & 2\cdot S_2(I_1)\cdot S_0(I_0) + 3\cdot S_2(I_1)\cdot S_1(I_0) + 1\cdot S_2(I_1)\cdot S_3(I_0) + \\ & 3\cdot S_3(I_1)\cdot S_0(I_0) + 1\cdot S_3(I_1)\cdot S_2(I_0) + 2\cdot S_3(I_1)\cdot S_3(I_0) \end{aligned} \qquad (10)$$

Transforming these expressions into the corrisponding digital circuits is a simple process, as happens in the binary case.

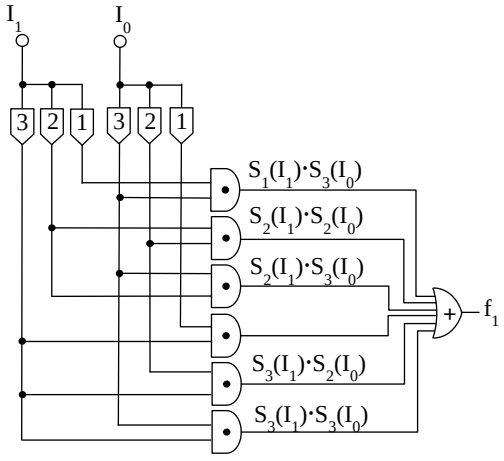

Fig. 3 MVL circuit that implements the function f_1

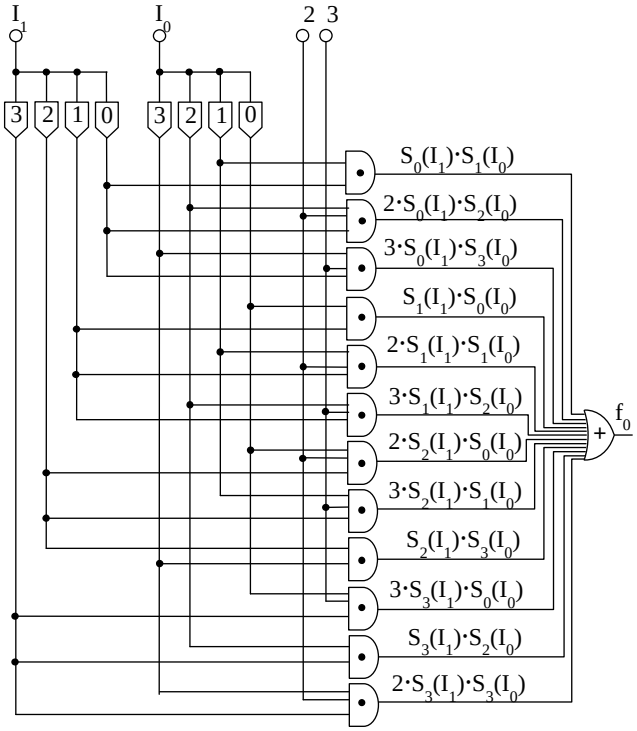

Fig. 4 MVL circuit that implements the function f_0

The corresponding to f_0 circuit will be implemented using the set of base functions explained but in this specific case, remembering the semantic of our operators, we can also calculate f_0 as:

$$f_0 = I_1 + I_0 \tag{11}$$

B. Basic element for an MVL memory

In this section, we present our design of a D flip-flop which is based on an extension of binary D flip-flop. In a D flip-flop the next state Q(t+1) is characterized by a function of both the current state Q(t) and the D data input. The next state Q(t+1) could be defined by:

$$Q(t+1)=D\cdot Q(t)+D\cdot\overline{Q(t)} \tag{12}$$

or

$$Q(t+1)=\left(D+Q(t)\right)\cdot\left(D+\overline{Q(t)}\right) \tag{13}$$

these two equations can be transformed in a fuzzy domain by replacing the binary operators by fuzzy operators as shown in [20][6]. Using min-max type operation and fuzzy negation we can write the following transformation:

$$Q(t+1)=\{[1-Q(t)]\wedge D\}\vee[D\wedge\overline{Q(t)}] \tag{14}$$

$$Q(t+1)=\{[1-Q(t)]\vee D\}\wedge[D\vee\overline{Q(t)}] \tag{15}$$

The symbol $\wedge$ represents min operation and $\vee$ represents max operation. Because these equations do not transform D flip-flop to the fuzzy domain, the authors proposed a different equation. The proposed circuit, however, is not simple to realize, therefore starting from the assumption that normally to construct memory elements, a clock is used that allows to restrict the sampling interval of the input D. In this case the transfer function can be written through the operators we have defined in Session II:

$$Q(t+1)=D\cdot S_1(CLK)+S_0(CLK)\cdot Q(t) \tag{16}$$

It can be understood easily that working with the flip-flop shown in Fig. 3, the value of the input D is posted to the output Q(t) when the CLK values is 1, otherwise (CLK=0) the circuit store the previous value: Q(t+1)=Q(t).

The corresponding digital circuit will then be:

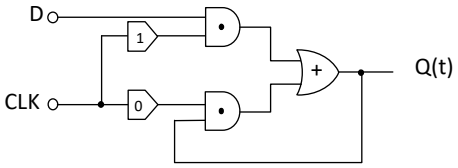

Fig. 5 D-Type Flip-Flop with clock signal (CLK)

Excitation table for this circuit is shown in Table IX.

TABLE IX
SIMULATION OF THE HALF-ADDER CIRCUIT

t	CLK	D	Q(t)	Q(t+1)
0	0	0	0	0
1	1	0	0	0
2	0	0	0	0
3	1	0	0	0
0	0	0	1	1
1	1	0	1	0
2	0	0	0	0
3	1	0	0	0
0	0	0	2	2
1	1	0	2	0
2	0	0	0	0
3	1	0	0	0
0	0	0	3	3
1	1	0	3	0
2	0	0	0	0
3	1	0	0	0
0	0	1	0	0
1	1	1	0	1
2	0	1	1	1
3	1	1	1	1
0	0	1	1	1
1	1	1	1	1
2	0	1	1	1
3	1	1	1	1
0	0	1	2	2
1	1	1	2	1
2	0	1	1	1
3	1	1	1	1
0	0	1	3	3
1	1	1	3	1
2	0	1	1	1
3	1	1	1	1
0	0	2	0	0
1	1	2	0	2

2	0	2	2	2
3	1	2	2	2
0	0	2	1	1
1	1	2	1	2
2	0	2	2	2
3	1	2	2	2
0	0	2	2	2
1	1	2	2	2
2	0	2	2	2
3	1	2	2	2
0	0	2	3	3
1	1	2	3	2
2	0	2	2	2
3	1	2	2	2
0	0	3	0	0
1	1	3	0	3
2	0	3	3	3
3	1	3	3	3
0	0	3	1	1
1	1	3	1	3
2	0	3	3	3
3	1	3	3	3
0	0	3	2	2
1	1	3	2	3
2	0	3	3	3
3	1	3	3	3
0	0	3	3	3
1	1	3	3	3
2	0	3	3	3
3	1	3	3	3

To simulate the behavior of this flip-flop, we investigate our design using a simple java program. The table 1 has been calculated importing the CSV output file generated by the class JMAT into a spreadsheet application like MS Excel or OO Calc.

TABLE X
JAVA EXAMPLE CODE FOR SIMULATE D-TYPE FLIP-FLOP CLOCKED

```
package jmat;
import java.io.IOException;
import java.io.PrintWriter;
public class JMAT {
   final static int N = 4; // The domain dimension
   public static void main(String[] args) throws IOException {
      int saveQt; // used to save the state Q(t)
      int CLK=0; // the square wave of the clock
      PrintWriter writer = new PrintWriter("logging.csv", "UTF-8");
      //heading of CSV file
      writer.println("t,CLK,D ,Q(t),Q(t+1)");
      for (int D = 0; D < N; D++) {
         for (int Qt = 0; Qt < N; Qt++) {
            // save actual state Q(t)
            saveQt = Qt;
            for (int t = 0; t < N; t++) {// t stands for time
               CLK= t % 2; //the clock
               // writing the row inside the CSV file
               writer.println("" + t + "," + CLK + "," +D + "," +
                              Qt + "," + FlipFlopD (CLK, D, Qt));
               Qt = FlipFlopD (CLK , D, Qt); //the next state for Q(t)
            }
            // resume original Q(t)
            Qt = saveQt;
         }
      }
      // Close writer
      writer.close();
   }
   public static int FlipFlopD (int CLK, int D, int Qt) {
      // this function realize the flip-flop's behaviour
      int x1 = MULTgate(SELgate(1, CLK), D);
      int x2 = MULTgate(SELgate(0, CLK), Qt);
      Qt = SUMgate(x1, x2);
      return Qt;
   }
   public static int SELgate(int S, int x) {
      return (x == S ? 1 : 0);
   }
   public static int MULTgate(int x, int y) {
      return (x * y) % N;
   }
   public static int SUMgate(int x, int y) {
      return (x + y) % N;
   }
}
```

As you can see, we have realized a simple method (*SELgate()*) that implements the 3 selectors and other 2 methods (*SUMgate()* and *MULTgate()*) that implement the binary functions SUM and MULT.

To simulate the operation of the sequential circuit we use grafted loops that allow you to vary D, Qt and the clock (CLK) in the set of possible values. We are interested in seeing the operation of the circuit both in the ability to keep the information stored when CLK=0, and to transfer D in the internal memory (Qt) if CLK=1.

IV. CONCLUSION

This article provides a different evolutionary line to improve the computing capacity of the CPUs that is based on the extension of the binary system in a multi-value coding system or, commonly, called MVL.

Although this direction has already been explored, the idea behind the study is in the representation of the generic function in the MVL domain. This representation has a link to the binary system and a surprisingly greater simplicity of the corresponding digital circuits (combinatorial and sequential). A different mathematical approach is thus provided for the realization of the multivalue logic gates. This could enable the use of different data encoding systems no longer linked to the voltage value of a signal (as seen in [18]) but to other physical quantities as it happens at present, for example, in the world of telecommunications.

The other important aspect is the scalability of the solution: this study illustrates a methodology that is independent of the basis of the adopted domain and could even be extended to fuzzy logic [8].

The proposed solution is not opposed to multi-core architectures, since it describes how the internal operating logic of a future CPU could be and therefore nothing prevents the creation of multi-core architectures with MVL.

REFERENCES

[1] D. Etiemble, *45-year CPU Evolution: one law and two equations*, Second Workshop on Pioneering Processor Paradigms, Vienna, February 2018,

[2] Haissam El-Aawar, *Increasing the transistor count by constructiong a two-layer crystal square on a single chip*, International Journal of Computer Science & Information Technology (IJCSIT) Vol 7, No 3, June 2015

[3] X. Chen, Y. Wardi, S. Yalamanchili, *Power regulation in high performance multicore processors*, Decision and Control (CDC) 2017 IEEE 56th Annual Conference on, pp. 2674-2679, 2017.

[4] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M., Silvestri, F., Spanò, S. *Energy consumption saving in embedded microprocessors using hardware accelerators*, Telecommunication Computing Electronics and Control (Telkomnika), 16 (3), pp. 1019-1026, 2018.

[5] Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Re, M., Lee, R.B. *Integration of butterfly and inverse butterfly nets in embedded processors: Effects on power saving*, Conference Record - Asilomar Conference on Signals, Systems and Computers, art. no. 6489268, pp. 1457-1459, 2012

[6] Adib Kabir Chowdhury, Nikhil Raj and Ashutosh Kumar Singh. *Design of Low Power MAX Operator for Multi-Valued Logic System*, Procedia Computer Science, pages 428 – 433, 2015.

[7] W. Alexander, *The ternary computer*, Electronics and Power, pages 36-39. February 1964.

[8] Ben Choi and Kankana Shukla, *Multi-Valued Logic Circuit Design and Implementation*, International Journal of Electronics and Electrical Engineering Vol. 3, No. 4, August 2015.

[9] Prashant S. Wankhade, Gajanan Sarate. *Minimization of Multiple Value function using Quine Mc-Cluskey Technique*. International Journal of Computer Applications (0975 – 8887) Volume 143 – No.7, June 2016.

[10] Israel Halpern and Michael Yoeli. *Ternary arithmetic unit*, Proceedings of the Institution of Electrical Engineers, Volume 115, Issue 10, October 1968

[11] Dhande A.P., Ingole V.T. and Ghiye V.R., *Thernary Digital System: Concepts and Applications*, SM Online Publishers LLC, ISBN: 978-0-9962745-0-0, October 2014.

[12] Satish Narkhede, *Design and Implementation of an Efficient Instruction Set for Ternary Processor*, International Journal of Computer Applications (0975 – 8887) Volume 83 – No.16, December 2013.

[13] Nayan Kumar, Naware Deepti, S. Khurge and S.U.Bhandari, *Review of Quaternary Algebra & its Logic Circuits*, International Conference on Computing Communication Control and Automation, pages 969-973, 2015.

[14] Ifat Jahangir, Dihan Md. Nuruddin Hasan, Shajid Islam, Nahian Alam Siddique, Md. Mehedi Hasan. *Development of a Novel* Quaternary Algebra with the Design of Some Useful Logic Blocks, Proceedings of 2009 12th International Conference on Computer and Information Technology (ICCIT 2009), Dhaka, Bangladesh, , pages 197 – 202, December 2009.

[15] Miller D.M. and Thornton M.A., *Multiple Valued Logic: Concepts and Representations*, Digital Circuits and Systems, Vol. 2, No. 1 , Pages 1-127, 2007.

[16] L. P. Nascimento, "An Automated Tool for Analysis and Design of MVL Digital Circuits", in 14th Symposium on Integrated Circuits and Systems Design, Pirenópolis-GO-Brazil, 2001.

[17] Amnon Rosenmann, *A Multiple-Valued Logic Approach to the Design and Verication of Hardware Circuits*, Journal of Applied Logic, Volume 15 Issue C, pages 69-93, May 2016

[18] B. Srinivasa Raghavan and V.S Kanchana Bhaaskaran, *Design of Novel Multiple Valued Logic (MVL) Circuits*, International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS^2 2017) Chennai, India 23-25 March 2017.