THE ANNALS OF "DUNĂREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2020, VOL. 43, NO. 1, ISSN 2344-4738, ISSN-L 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS
Article DOI: https://doi.org/10.35219/eeaci.2020.1.03

HARDWARE IMPLEMENTATION OF A LOW COST MATH MODULE USING MULTIFUNCTIONAL REGISTERS WITH DECODED MODE INPUTS

Grigore Mihai TIMIS*, Alexandru VALACHI*

**Technical University "Gh.Asachi"Iasi, Faculty of Automatic Control and Computer Engineering (e-mail: mtimis@ tuiasi.ro, avalachi@tuiasi.ro).*

Abstract: In the present paper, we propose a low cost algorithm of a math module and the implementation using multifunctional registers with decoded mode inputs. The proposed math module algorithm will be implemented using the transition matrix method. According with taxonomy of the algorithms, we use the functional iteration one. It is found in specific literature that it can provide the lowest latency and greatest reliability. Compared with CORDIC math module which is based on the hardware iteration algorithm with design implemented in FPGA (which is more expensive and slow than a dedicate hardware), our proposed math module algorithm use less hardware, means the chip area is minimized, working at a high speed rate. There, will be proved that implementation of the digital automaton can be reduced to a combinational one, this will lead to the economical implementation.

Keywords: math module algorithm, automaton execution elements, multifunctional registers, finite state machine.

1. INTRODUCTION

In today devices, it is well known that Processor Floating Point Units – FPU, which handle huge amount of data, are used at the maximum capacity. Based on this fact, we propose an optimal low cost method for a hardware implementation of a math module between two unsigned integer numbers. This represents a small part from the FPU's logic. The mathematical computations can be found inside of Intel 80x86, AMD 80x86, Sun, Sparc, IBM logic cores architectures.

The calculation operations of the FPU processor must be always done in fast and precise way to avoid system crash and errors with the state-of-the-art of the computer technology.

The proposed algorithm is based on functional iterations, using only simply counting and decrementing operations. Based on specific literature (Premys et al., 2011), (Yi-Jun et al., 2011), (Ranjan Kumar et al., 2016), (Mihailov et al., 2011), (Morris et al., 2006), (Morris et al., 2005), (Skylarov et al., 2005), it is found that for low-cost implementations where chip area must be minimized, the iteration algorithms are suitable.

It is well known that although is used an optimal algorithm, the resulted synthesis can be non-optimal, that is why the pipelined combinational logic will be optimised.

The detailed outline of the paper: Section II, details about the binary matrices operations; Section III, describes the state diagram functional representation based on the transition matrix method; Section IV,

outlines the global synthesis for the math module, experimental results and implementation costs; Section V, Further work; Section VI, final remarks.

2. BINARY MATRICES OPERATIONS

We define the transition matrix, noted with $T_{r\_matrix} = (t_{ij})_{MxM}$ which is build using the state machine diagram, where M represents the numbers of machine states.

The pure binary codification for distinct operations is used like in (Valachi et al., 2010), (Morris et al., 2006). In order to obtain the transition matrix coefficients, it's considered the next operations:

- Count Up, $t_{i+1,i} = 1$, makes the transition from state i to state $i+1$
- Count Down, $t_{i-1,i} = 1$, makes the transition from state i to state $i-1$
- Hold, $t_{ii} = 1$, the automaton remains in state *i*, makes the transition from state *i* to state *i* means that remains in the same state
- Shift, $t_{ij} = 1$, if $i = 2 \cdot j$, with $(2 \cdot j \le M)$ - Shift Left or if $i = \left[\frac{j}{2}\right]$ - Shift Right
- Reset, $t_{1j} = 1$, $j \ne 1$, makes the transition from state j to initial state $i = 1$, $i \ne j$
- Parallel Load, $t_{ij} = 1$, $i \notin \{1, j+1, j-1, 2 \cdot j, \left[\frac{j}{2}\right], j\}$, makes the transition from state *j* to state *i* where [x] represents *x* integer part.

Considering $C = (C_{ij})_{pxp}$, a binary matrix, we note with $\overline{C} = (\overline{c_{ij}})_{pxp}$, the complement of that matrix.

For example:

$$(1)\quad C = \begin{bmatrix} 1 & b \\ c & 0 \end{bmatrix},\ \overline{C} = \begin{bmatrix} 0 & \overline{b} \\ \overline{c} & 1 \end{bmatrix}$$

For the two binary matrices logic multiplication with the same dimension, we propose the following matrices $B = (b_{ij})_{mxn}$, $C = (c_{ij})_{mxn}$. The logic multiplication result noted with $R = B \cdot C = (b_{ij} \cdot c_{ij})_{mxn}$

Ex. $B = \begin{bmatrix} z & 1 \\ d & 0 \end{bmatrix}, C = \begin{bmatrix} x & s \\ 1 & e \end{bmatrix}$

$B \cdot C = \begin{bmatrix} z \cdot x & s \\ d & 0 \end{bmatrix}$

2.1. The two-matrices multiplication product algorithm

For the two matrices multiplication, let us consider $S = (s_{ij})_{mxp}$, $L = (l_{ij})_{pxn}$

Considering these two matrices, by multiplication of them, it will define the W matrix, as: $W = S \otimes L = (c_{ij})_{mxn}$, where $c_{ij} = \sum_{k=1}^{p} s_{ik} \cdot l_{kj}$, Σ - represents the logic adder.

For example:

$$(2)\quad S = \begin{bmatrix} x & 1 & a \\ 0 & b & 1 \end{bmatrix},\ L = \begin{bmatrix} 1 & 1 \\ a & u \\ c & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} x + a + a \cdot c & x + u + 0 \\ 0 + a \cdot b + c & 0 + b \cdot u + 0 \end{bmatrix} = \begin{bmatrix} x + a & x + u \\ a \cdot b + c & b \cdot u \end{bmatrix}$$

3. THE STATE DIAGRAM FUNCTIONAL REPRESENTATION BASED ON THE TRANSITION MATRIX METHOD

Considering a digital automaton that compute the math module for two unsigned integers numbers: $x = x[7:0], y = y[7:0]$, where x represents the first operand and y represents the second one. The processing results will be $Q = Q[7:0]$.

The following assignments were used: $xNull = (x = 0)$ and $yNull = (y = 0)$.

This algorithm can be used with good results for any operand size – 8, 16, 32 bits.

The logic diagram with the iteration operation algorithm:

Algorithm description – math module $|x-y|$:

Step 1: Load $x = x[7:0], y = y[7:0]$;
Step 2: If x is not zero (*xNull* =0) and y is not zero (*yNull* =0) then go to Step 3;
Step 3: $x = x-1, y = y-1$;
Step 4: If x is zero or y is zero, go to Step 5;
Step 5: If x value is zero then the result is stored in y;
Step 6: If y value is zero, then the result is stored in x;
Step 7: Read the results;
Step 8: Stop algorithm.

Fig. 1. Logical description of the math module algorithm

3.1. Algorithm description

After loading two operands $x = x[7:0]$ and $y = y[7:0]$, the algorithm will start: if $xNull$ and $yNull$ are not active, means operands are not zero, the following algorithm will be executed: x value and y values are decrementing in parallel, till one of them becomes zero. If the x value becomes null, this means the math module result is stored in y. Also, if the y value becomes null, this means the math module result is stored in x.

Observation: the logical description steps from the multiplication algorithm - Fig.1 are not the same as the states from functional organizational chart - Fig.3, but the proposed algorithm is the same.

3.2. Description of the Automaton Execution Elements (EEA)

The logic circuits used are the multifunctional registers with decoded mode signals and synchronized with a clock signal. The imposed priority levels are: Reset has the highest priority level, increment/decrement has the lower priority level. The input command signals $\overline{LDx}$, $\overline{LDy}$ are asynchronous and has higher priority level than the other signals.

R_x register functionalities: asynchronous store the $x = x[7:0]$ operand, load of data $D[7:0]$, synchronous reset and decrement on the positive edge of the h_1. The signal $xNull = 1$ shows that the content of the R_x register is null.

R_y register functionalities: asynchronous store the $y = y[7:0]$ operand, load of data $D[7:0]$, synchronous reset and decrement on the positive edge of the h_1. The signal $yNull$ shows if the content of the register is null.

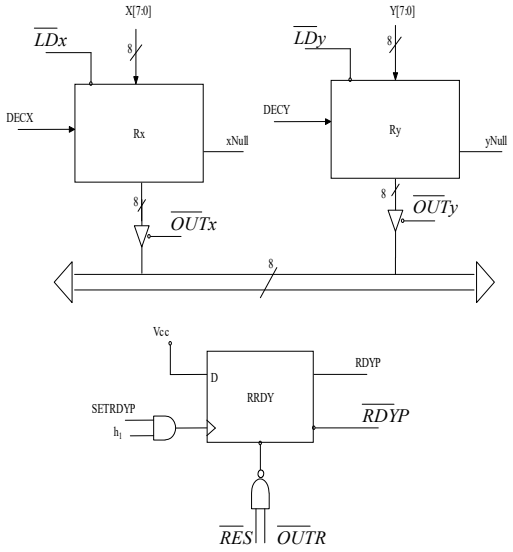

Fig. 2. Automaton Execution Elements - EEA for the math modulo algorithm

The D Flip-Flop RDYP functionalities: at the end of the processing, will provide the RDYP signal.

The reset of the CBB-D is done using the hardware reset $\overline{(RES = 0)}$ or at the reading of the result $\overline{(OUTR = 0)}$.

The Three State Control Buffer (TSC) BUFOUT: using two consecutive READ operations it allows the transfer of the result.

ST Start Signal: is set only after the loading command of two data operands was received. On asynchronous hardware reset or after the final result was read, it is deactivated.

Considering the states attached code $y_n = \{y_2 y_1 y_0\}_n$, as:

$y_n = 000 \leftrightarrow s_0, 001 \leftrightarrow s_1, 010 \leftrightarrow s_2, 011 \leftrightarrow s_3, 100 \leftrightarrow s_4, 101 \leftrightarrow s_5$

The code matrix, noted with C, (3):

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (3) \qquad (3)$$

D_i represents the *i* range input data in current states code register and is used in order to obtain the column vector [2], fig.4.

The modules algorithm description:

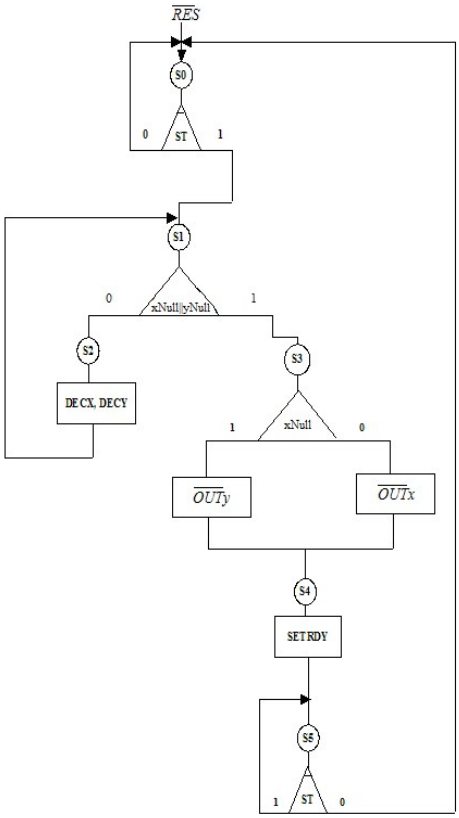

Fig. 3. Functional organizational chart

$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}_{n+1} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \end{bmatrix} = (C \otimes T) \otimes S$, S represents the column vector, $S = \begin{bmatrix} s_0 \\ s_1 \\ . \\ . \\ . \\ s_5 \end{bmatrix}$

From the functional organizational chart, it is deducted the transition matrix, (4):

$$(4) \quad T = \begin{bmatrix} \overline{ST} & 0 & 0 & 0 & 0 & \overline{ST} \\ ST & 0 & 1 & 0 & 0 & 0 \\ 0 & \overline{(x=0)+(y=0)} & 0 & 0 & 0 & 0 \\ 0 & (x=0)+(y=0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & ST \end{bmatrix}$$

After a series of successive computing, is obtained relations (5):

$$(5) \quad C \otimes T = \begin{bmatrix} ST & (x=0)+(y=0) & 1 & 0 & 1 & ST \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & ST \end{bmatrix}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = (C \otimes T) \otimes S =$$

$$= \begin{bmatrix} S_0 \cdot ST + S_1 \cdot [(x=0)+(y=0)] + S_2 + S_4 + S_5 \cdot ST \\ S_1 \\ S_3 + S_4 + S_5 \cdot ST \end{bmatrix}$$

The relations from (5) represents the SLC1 (Combinational Logic) equations. The architecture for the hardwired sequencer, implemented with codified sequences, is shown in figure 4. The command signals relations are presented in equations (6).

$$(6) \quad DEC_x = DEC_y = S_2$$

$$\overline{OUT_y} = S_3 \cdot (x=0)$$

$$\overline{OUT_x} = S_3 \cdot (\overline{x=0})$$

$$SETRDYP = s_4$$

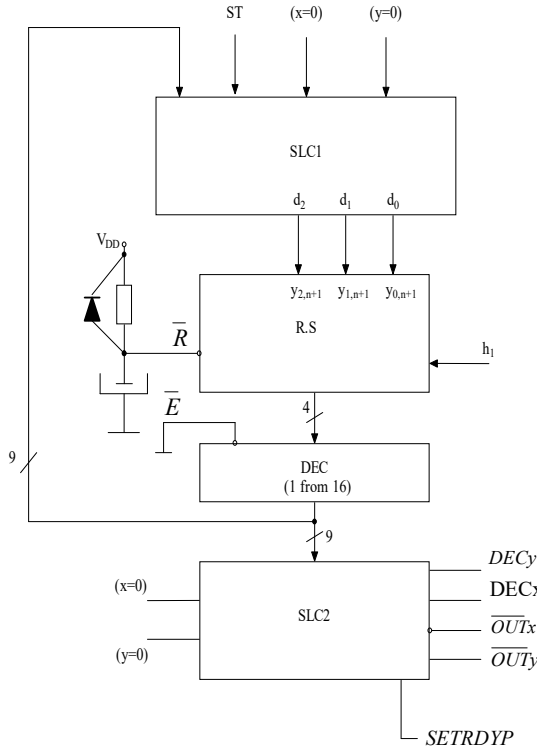

Fig. 4. Hardwired Sequencer

The final implementation cost charts for the SLC2 will not taken into account, because this synthesis remains the same in every design implementation.

The first proposed implementation of design from fig.4 is like in fig.5.

The following relations (7), are deducted from (3):

$$(7)\quad \begin{aligned} y_0 &= S_0 \cdot ST + S_1 \cdot [(x=0)+(y=0)] + S_2 + S_4 + S_5 \cdot ST \\ y_1 &= S_1 \\ y_2 &= S_3 + S_4 + S_5 \cdot ST \end{aligned}$$

The total cost of the SLC1 represents the total number of logic gates multiplies by the inputs. Thus, the implementation is calculated as in equation (8):

$$(8)\quad \begin{aligned} &C_1(SLC_1) = C \cdot (D_0) + C \cdot (D_1) + C \cdot (D_2) = \\ &= 14 + 0 + 5 = 19 \end{aligned}$$

4. MATH MODULE COMPLEX AUTOMATON SYNTHESIS USING MULTIFUNCTIONAL REGISTERS, IMPLEMENTATION COSTS AND EXPERIMENTAL RESULTS

The novelty of the proposed method represents the separate synthesis for the digital logic system which generates the $\overline{SR}$ reset signal, $\overline{PL}$ parallel load signal, INC increment signal, DEC decrement signal etc.

All the logic function has the corresponding transition matrices TR, TP, TI .

Relation involved:

$$(9)\quad T(F) = I(F) \cdot T$$

Operation identification matrix is noted with $I(F)$.

Priority orders: Rest - high priority, INC / DEC low priority.

The validation matrices for Reset, Parallel Load, Increment operations are noted with IR, IP, II , where Φ symbol means indifferent values (nor 0 logic or 1 logic), are shown in (10).

$$(10)\quad IR = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$IP = \begin{bmatrix} 0 & \Phi & \Phi & \Phi & \Phi & \Phi \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$II = \begin{bmatrix} 0 & \Phi & \Phi & \Phi & \Phi & \Phi \\ 1 & 0 & \Phi & \Phi & \Phi & \Phi \\ \Phi & 1 & 0 & \Phi & \Phi & \Phi \\ \Phi & \Phi & 1 & 0 & \Phi & \Phi \\ \Phi & \Phi & \Phi & 1 & 0 & \Phi \\ \Phi & \Phi & \Phi & \Phi & 1 & 0 \end{bmatrix}$$

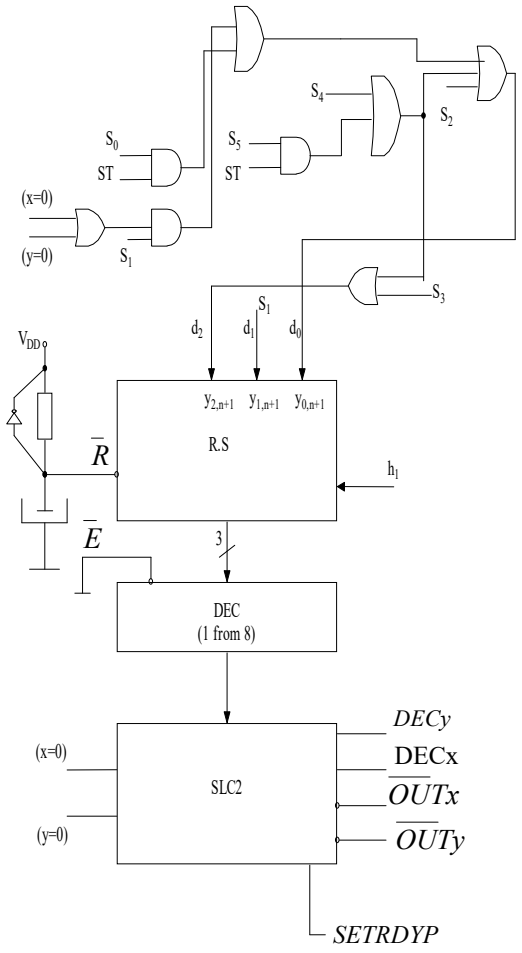

Fig. 5. Hardwired sequencer

- II is computed as: $t_{ii}=0, t_{ik}=1$ for $i=k+1$ - for hold state and $t_{ij}=\Phi$ for the rest (increment operation has the lowest priority order).
- IP is computed as: $t_{1k}=\Phi$, for $k\neq 1$ (reset operation has highest priority), $t_{ii}=0, t_{ik}=1$ for $i\neq k+1$

IR is computed as: $t_{1k}=1$, for $k\neq 1$, transitions on first state.

The specific transition matrices are shown in (11), (12), (13):

$$(11)\quad TR=IR\cdot T=\begin{bmatrix}0&0&0&0&0&\overline{ST}\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

$$(12)\quad TP=IP\cdot T=$$

$$\begin{bmatrix}0&0&0&0&0&\overline{ST}\cdot\Phi\\0&0&1&0&0&0\\0&0&0&0&0&0\\0&(x=0)+(y=0)&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

$$(13)\quad TI=I(INC)\cdot T=$$

$$=\begin{bmatrix}0\cdot\overline{ST}&0&0&0&0&\overline{ST}\cdot\Phi\\ST&0&1\cdot\Phi&0&0&0\\0&1\cdot\left[\overline{(x=0)+(y=0)}\right]&0&0&0&0\\0&\left[(x=0)+(y=0)\right]\cdot\Phi&0&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}$$

For the $\overline{SR}$, $\overline{PL}$, INC relations it will be used the line vector [111111].

In order to simplify the computations of the column vector sum, is used the following relation: $[T(F)\otimes S]$.

$$(14)\quad \overline{\mathrm{SR}}=\overline{[111111]\otimes[\mathrm{TR}\otimes\mathrm{S}]}=\overline{s_5\cdot\overline{ST}}$$

$$\overline{\mathrm{PL}}=\overline{[11111]\otimes[\mathrm{TP}\otimes\mathrm{S}]}=$$

$$=\overline{S_5\cdot\overline{ST}\cdot\Phi+S_2+S_1\cdot\left[(x=0)+(y=0)\right]}=$$

$$=\overline{S_2+S_1\cdot\left[(x=0)+(y=0)\right]}$$

$$\mathrm{INC}=[111111]\otimes[\mathrm{TI}\otimes\mathrm{S}]$$

The transition matrices for the INC function, (15):

$$(15)\quad TI\otimes S=\begin{bmatrix}S_5\cdot\overline{ST}\cdot\Phi\\ST\cdot S_0+S_2\cdot\Phi\\S_1\cdot\left[\overline{(x=0)+(y=0)}\right]\\S_1\cdot\left[(x=0)+(y=0)\right]\cdot\Phi\\S_3\\S_4\end{bmatrix}$$

The final INC equation:

$$(16)\quad \begin{aligned} &INC = [111111] \otimes [TI \otimes S] = \\ &= s_5 \cdot \overline{ST} \cdot \Phi + s_0 \cdot ST + S_2 \cdot \Phi + S_1 + S_3 + S_4 \end{aligned}$$

In order to result an optimal expression, the values for Φ were chosen as preferable, equation (17):

$$(17)\quad INC = s_0 \cdot ST + S_1 + S_3 + S_4$$

For the next transition sequence, there are deducted the following equation, (18):

(18)

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = [C \otimes TP] \otimes S \quad , \quad \begin{bmatrix} \overline{d_0} \\ \overline{d_1} \\ \overline{d_2} \end{bmatrix} = [\overline{C} \otimes TP] \otimes S$$

The code sequences matrix and it's complement are noted with $C, \overline{C}$, (19):

$$(19)\quad \begin{bmatrix} d_0 \\ d_1 \\ d_0 \end{bmatrix} = \{ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \otimes T\Pi \} \otimes S =$$

$$= \begin{bmatrix} S_1 \cdot [(x=0)+(y=0)] + S_2 \\ S_1 \cdot [(x=0)+(y=0)] \\ 0 \end{bmatrix}$$

The optimal implementation, (20):

$$(20)\quad \begin{aligned} d_0 &= S_1 \cdot [(x=0)+(y=0)] + S_2 \\ d_1 &= S_1 \cdot [(x=0)+(y=0)] \\ d_2 &= 0 \end{aligned}$$

The implementation of the hardwired sequencer is shown in figure 7.

The implementation cost for the SLC1, (21):

$$(21)\quad \begin{aligned} &C_1(SLC_1) = C(\overline{SR}) + C(\overline{PL}) + \\ &C(INC) + C(d_3 d_2 d_1 d_0) = \\ &= 2 + 6 + 6 + 6 = 20 \end{aligned}$$

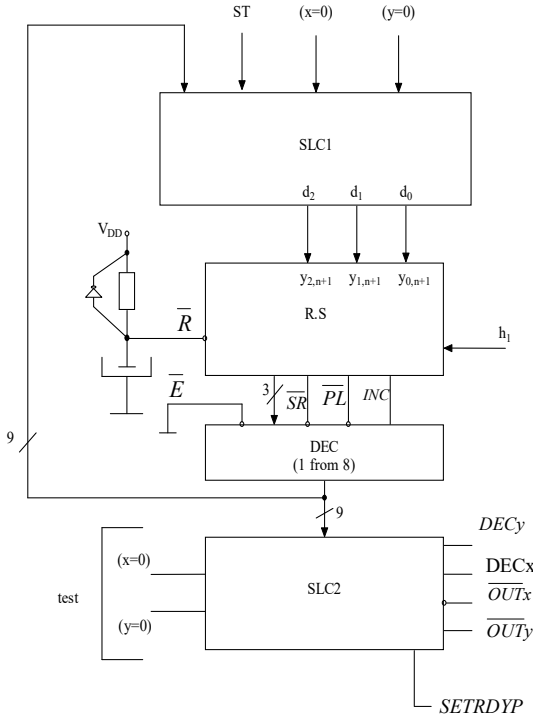

Fig. 6. Hardwired sequencer

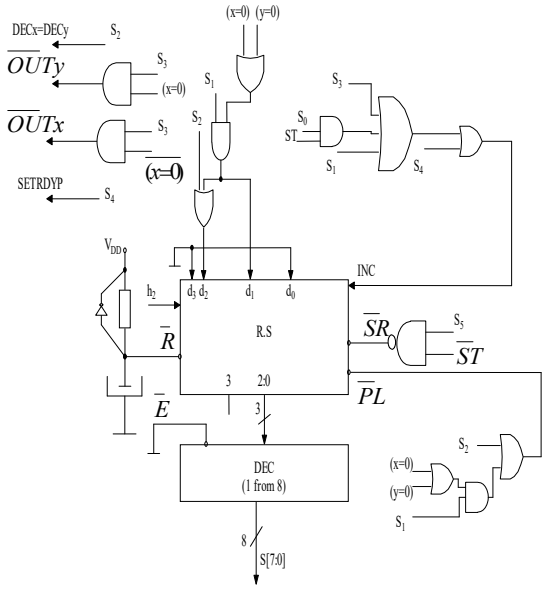

Fig. 7. Implementation of the hardwired sequencer

FUTURE WORK

This implies the low cost implementation and synthesis for a multifunctional digital device with arithmetic and logic operations. This will simulate a relative huge portion from the FPU wafer. Also, we will study timing in digital systems because fault free

digital circuits may malfunction when asynchronous inputs have critical timing combinations.

5. CONCLUSIONS

According with taxonomy of the algorithms, we use the functional iteration one. It is found in specific literature (Mihailov et al., 2010), (Teodorescu et al., 2010), (Valachi et al., 2010), (Morris et al., 2006), (Morris et al., 2005), (Skylarov et al, 2005), (Peng et al., 1987), (Ursaru et al., 2009), (Rodriguez et al., 2008) that it can provide the lowest latency and greatest reliability.

For the proposed algorithm, we show in our paper that the synthesis with multifunctional registers simplify the FPU digital hardware logic. Balanced with the math module algorithms and synthesis methods available in references, our iterations based algorithm works fine with integers numbers with 8,16,32 bits, this means low cost implementation and reliability. For 64bits integer numbers, due latency, a variable number of digital slices should be used accompanying by a digital arbiter.

Based on available papers listed in the references section, we proved that our research is an actual one.

Compared with CORDIC math module (Muhammad et al., 2013) which is based on the hardware iteration algorithms with design implemented in FPGA (which is more expensive and slow than a dedicate hardware), our proposed math module algorithm use less hardware, means the chip area is minimized and works at a high speed rate for the integers numbers with 8, 16, 32 bits. There, was proved that implementation of the digital automaton can be reduced to a combinational one, which leads to the economical implementation.

As a final conclusion, we proposed two methods for synthesis the digital device: first proposed method has a smaller implementation cost – C=19, than the second proposed method – C=20 and those described in specific literature from references. This leads to a small number of the logic gates and digital logic that is used. Low cost means the FPU logic core is much faster and the responses timing are short. Moreover, it's about green architectures which means less power consumed.

6. REFERENCES

Ranjan Kumar Barik; Itishree Samal; Manoranjan Pradhan, "Efficient hardware realization of signed arithmetic operation using IEN" in Power, Communication and Information Technology Conference (PCITC), 2015 IEEE, 15-17 Oct. 2015, DOI: 10.1109/PCITC.2015.7438171, INSPEC Accession Number: 15885905, Date Added to IEEE Xplore: 24 March 2016, IEEE Conference Publications.

Muhammad Nasir Ibrahim; Chen Kean Tack; Mariani Idroas; Siti Noormaya Bilmas; Zuraimi Yahya, "Hardware Implementation of Math Module Based on CORDIC Algorithm Using FPGA", in International Conference on Parallel and Distributed Systems, 2013, Pages: 628 - 632, DOI: 10.1109/ICPADS.2013.112, IEEE Conference Publications.

D. Yi-Jun, B. Zhuo, "CORDIC algorithm based on FPGA", Journal of Shanghai University, vol. 15, no. 4, pp 304-409, Aug 2011.

Premysl Sucha; Zdenek Hanzalek; Antonirn Hermanek; Jan Schier, "Efficient FPGA Implementation of Equalizer for Finite Interval Constant Modulus Algorithm" in Industrial Embedded Systems, 2011. IES '06. International Symposium on Industrial Embedded Systems, 18-20 Oct. 2011, DOI: 10.1109/IES.2006.357480, INSPEC Accession Number: 9551929, Date Added to IEEE Xplore: 07 May 2011, IEEE Conference Publications.

Dmitri Mihailov; Valery Sklyarov; Iouliia Skliarova; Alexander Sudnitson, "Parallel FPGA-Based Implementation of Recursive Sorting Algorithms", in 2010 International Conference on Reconfigurable Computing and FPGAs, Date of Conference: 13-15 Dec. 2010, Date Added to IEEE Xplore: 20 January 2011, INSPEC Accession 11791744, DOI: 10.1109/ReConFig.2010.30,Publisher: IEEE.

Horia-Nicolai Teodorescu, Mircea Hulea, "Improving time measurement precision in embedded systems with a hybrid measuring method", in Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2010 IEEE 6th International Conference, Volumul 1, Pag. 59-64, Editor IEEE.

Al. Valachi, M.Timis, S.Tarcau, B.Aignatoaie, "Orders Priorities Settings Criteria for Multifunctional Registers" in Electronics and Electrical Engineering. Intl. Journal of Electronics and Telecommunications Kaunas: Technologija, 2010.

Ovidiu Ursaru, Cristian Aghion, Mihai Lucanu, Liviu Tigaeru, "Pulse width Modulation Command Systems Used for the Optimization of Three Phase Inverters", Advances in Electrical and Computer Engineering Journal. Suceava, Romania, 2009, pag.22-27.

M.R.D. Rodrigues; J.H.P. Zurawski; J.B. Gosling, "Hardware evaluation of mathematical functions", in Computers and Digital Techniques - Volume: 128, Issue: 4, Date of Publication: 11 November 2008, Page(s): 155 – 164, Print ISSN: 0143-7062, DOI: 10.1049/ip-e:19810029, Published in: IEE Proceedings.

Lin Yuan, Gang Qu, Villa, T., Sangiovanni-Vincentelli,A., "An FSM Reengineering Approach to Sequential Circuit Synthesis by State Splitting", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(Volume:27,Issue: 6), may 2008, pages 1159-1164.

Gerald R. Morris; Viktor K. Prasanna; Richard D. Anderson, "A Hybrid Approach for Mapping Conjugate Gradient onto an FPGA-Augmented Reconfigurable Supercomputer", in 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Date of Conference: 24-26 April 2006, Date Added to IEEE Xplore: 11 December 2006, INSPEC Accession Number: 9274737, DOI: 10.1109/FCCM.2006.8.

G.R. Morris; V. K. Prasanna, "An FPGA-based floating-point Jacobi iterative solver", in 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05), Date of Conference: 7-9 Dec. 2005, Date Added to IEEE Xplore: 16 January 2006, Print ISBN: 0-7695-2509-1, INSPEC Accession Number: 8846596, DOI: 10.1109/ISPAN.2005.18, Publisher: IEEE.

V. Skylarov; I. Skilarova; B. Pimentel, "FPGA-based implementation and comparison of recursive and iterative algorithms", in International Conference on Field Programmable Logic and Applications, 24-26 Aug. 2005, Date Added to IEEE Xplore: 10 October 2005, INSPEC Accession Number: 8813928, DOI: 10.1109/FPL.2005.1515728, Publisher: IEEE.

Victor Peng, Sridhar Samudrala, Moshe Gavrielov., Sangiovanni- Vincentelli, A., "On the implementation of shifters, multipliers, and dividers in VLSI floating point units", in Computer Arithmetic (ARITH), 1987 IEEE 8th Symposium, pages 95-102.