Study of 32-bit RISC Processor Architecture and VHDL FPGA Implementation 32-bitMatrix Manipulation

Vandana S. Shah
Asstitant Professor, E&C Dept, SCET, Surat (India)

Dr. R. V. Kshirsagar
Professor (E&C Dept), Dean (Acad.), PCEA, Nagpur and Dean (Engineering Faculty), STM Nagpur University, Nagpur (India)

ABSTRACT
Present title discloses a distinctive method to cram the processor behavior while dealing with the multifaceted task of matrix manipulation. System facilitates this distinct feature by allowing user to input the data in suitable form and observe the output using suitable display devices. [5]

System is build around high performance VLSI technology. Matrix manipulation is on the whole parallel architecture of logical expressions. VLSI when implemented using High Performance Gate Arrays becomes most suitable for implementing parallel architecture. [6]

Keywords
VLSI, Gate Arrays, Parallel Architecture, Matrix Manipulation.

1. INTRODUCTION
Advanced RISC Machine (ARM) is the Implementation result of the RISC microprocessor architecture. Reduced Instruction Set Computer (RISC) basically represents a processor architecture design methodology, emphasizing the insight that highly simplified instructions that perform fewer tasks can be used to achieve superior performance when executed at higher rate. Fundamental of executing the small instructions at higher speed is also called LOAD – STORE architecture. Such small instructions are of fixed length helps to implement the Pipelined architecture. RISC family includes ARM, AVR, MIPS, ARC, ALPHA, Power Architecture, SPARC etc. [1]

RISC introduces simpler hardware processor architecture because Fixed Length Instruction format with the opcode in the same bit position requires less decoding, allowing any register to be used in any context simplifies the compiler design and Complex addressing is performed through sequences of arithmetic and/or load store operations.

Different manufactures that develops ARM architecture are Advanced Micro Devices, Atmel Corporation, Fujitsu, IBM, Intel, Kawasaki LSI, Microchips, NVIDIA, Texas, etc.

Even though such powerful architecture from different manufacturers provides best architecture solution, the proposed co-processor design having VHDL – FPGA implementation provides the arrangement with which one can manually apply the input stimulus (Input Values) of the matrix using the keypad , select the operation to be perform (multiplication, transpose, inverse) and finally observe the manipulated data on the display screen.

The proposed system is best option to analyze any higher level application, at bit level, in which complex process of mathematics such as matrix multiplication, transpose and inverse is used. E.g. in Image Processing.

Same design can be then converted to operate at higher operating speed, with fewer modifications, which can be used in different applications design such as Image processing, as a Math Co – Processor and etc. [7]

2. SYSTEM BLOCK DIAGRAM
The matrix manipulation system can be implemented using high performance Field Programmable Gate Arrays **(FPGA)**. Figure1 shows the system block diagram for Matrix Manipulation using FPGA.

Front end designers support different versions of the FPGA, like Xilinx XC2S100/200/300/400, XC3S100/200/300/400, or XC4S100/200/300/400 and so on, for the design under considerations XC2S100 is the best suitable flavor. It supports 16-bit LUT RAM, In-System Programming **(ISP)**, Boundary Scan and Read Back Ability that makes it most suitable for the said applications.

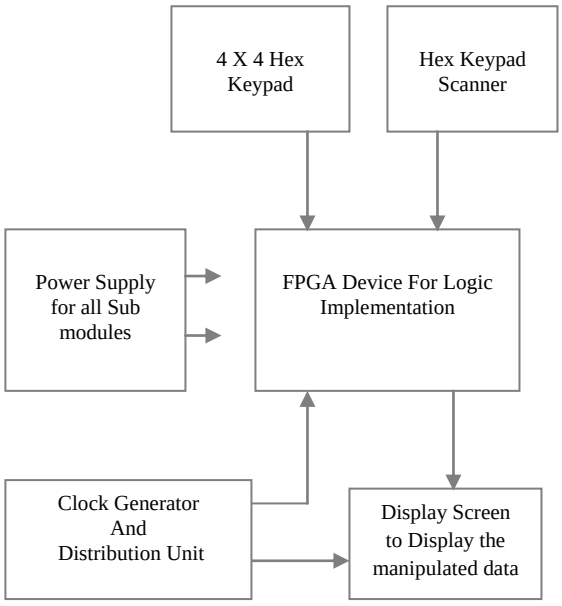

Figure1. System Block Diagram

3. GATE ARRAYS

Many forms of gate arrays are available like PLA, PAL, PLD and FPGA. These devices can be differed based on their granularity, ascending order of their appearance, and the programming methodology of the individual. PLA, PAL and PLDs are basically implementation of simple SOP, POS or combination of both representations and are smaller in capacity as compared to the FPGA devices.

FPGA devices are based on SRAM technology, it consist of Configurable Logic Blocks (CLBs), Programmable I/O Blocks, Delayed lock loops (DLL) and ISP Supports. CLBs are the unit that holds the user logic, and can be configured by application program created by the user. Application programs are created by certain Hardware Description Language (HDL).

3.1 Look up table

Look Up table is an implementation of universal device i.e. multiplexer. Figure2 depicts architecture of look-up table.

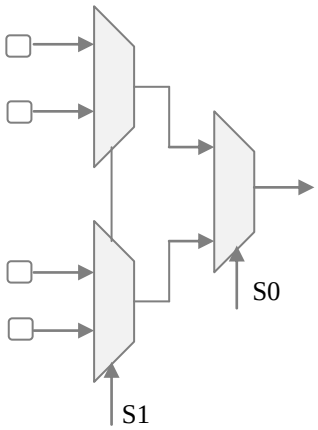

Figure2. Look Up Table.

Look up table is much similar to the logic expression implementation using decoder structure, as shown in figure 2, combination of three multiplexer implements any boolean expressions having four inputs and single output. But as the multiplexes are volatile device, data will lose when power is down; the Look up table can hold the data for power up period.

Field Programmable Gate Array consists of enormous LUTs, that, when described using hardware description language, implements the required logic.

3.2 Configurable Logic Blocks

The application program can be described by the user using hardware description language, like Very High Speed Integrated Circuit Hardware Description Language (VHDL) or VeriLog.

The hardware description of the application is converted to physical level that is represented using Boolean expressions that may be optimized highly considering factors like speed or area.

The Boolean expression that is to be implemented using physical level architecture may be purely combinational or purely sequential or combination of both of the techniques. The configurable logic block is responsible for successful implementation of Boolean expressions. Figure 3 shows generalized architecture for the configurable logic blocks.

As per the given logic circuit, if the circuit is purely combinational then the D-FF is bypassed and the logic output is produced at the output. But in case of implementing the sequential circuit logic the D-FF is engaged and the final output is computed. This option can be implemented by using one 2:1 multiplexer; by adjusting the select line appropriate logic can be implemented.

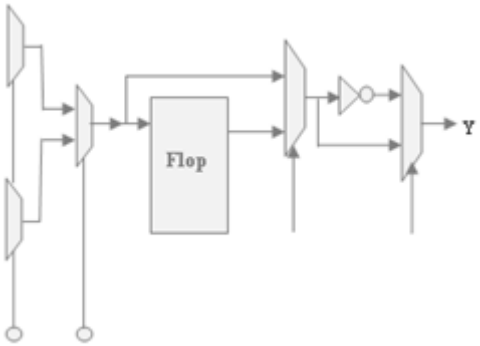

Figure 3: Configurable Logic Block

Again the output may be true or complemented; to have this astonishing feature 2:1 multiplexer can be used with one true input and the second one as inverted input. By adjusting the select input line implementation of the desired logic pattern is possible.

3.3 Buffers

Field Programmable Gate Arrays (FPGAs) are usually programmed and used to have higher speed execution per unit time. While implementing such high speed logic and to have read back facility they are equipped with high performance bidirectional buffers. Figure 4 depicts the architecture of 1-bit bidirectional buffer.

They are intentionally used for signal flattening and boundary scanning process. Boundary scanning is a process like emulation which is performed to check the hardware functionality of the FPGA device; secondly it is used for signal flattening.

For emulation process, the buffers are interfaced in queue, the first input of the buffer is used to input the test pattern and is called as Test Data Input (TDI), and the output of the last buffer is used to collect the data pattern sent from the TDI, called as Test Data Output (TDO).

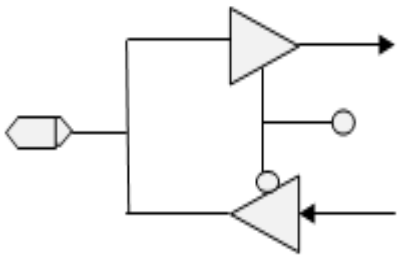

Figure 4: Input Output Buffer.

If the data pattern sent from the TDI pin is received successfully at the TDO pin the it is test successful but in case if mismatch occurs between the data at TDI and the TDO then test fail occurs and further it can be concluded that device is either failure or there is certain power supply related problem.

3.4 In-System Programming

With the help of the In-System Programming it is possible to make the device program when it is interfaced in the application. The modern technology makes the memory technology more user friendly by the advent of the USB technology which is also called as "Plug and Play Technology". In the present context, by the advent of ISP technology the Gate Arrays like FPGA or CPLD are called as Designer **Friendly Gate Arrays.**

In this technique the device itself generates the internal voltage for programming the unit. With such unusual feature it is possible for the user to integrate the programming and testing into single phase.

4. HEX KEY-PAD

Hex keypad is the matrix arrangement of 4x4 press to on switches. With 4x4 arrangements it becomes 16 possible values that can be encoded and designer can encode the values as per their convenience. For entering the values for matrix manipulation it is required to values to be input form the 4x4 matrix.

4.1 Hex Key-pad Scanning

For encoding the values from the 4x4 hex keypad matrix, keypad scanning routine needs to be executed repeatedly, keypad can be scanned by enabling only one row **(or column)** at a time and checking the individual column **(or row).** For each of the column in an enabled row, different values can be encoded. Figure 5 shows the matrix like structure of the Hex-keypad.

j / i	R1	R2	R3	R4
C1	A11	A12	A13	A14
C2	A21	A22	A23	A24
C3	A31	A32	A33	A34
C4	A41	A42	A43	A44

Figure 5: Hex Key-Pad like structure

Same process can be repeated for remaining rows; for first row four values can be allotted, similarly, for four rows, 16 different values can be allotted and encoded by the designer.

For matrix manipulation hexadecimal values, 0 to 9 and A - H values are used and hence are encoded from the keypad scanning accordingly.

5. DISPLAY UNIT

The manipulated data by the FPGA can be displayed on the various display units like 7 segment displays, matrix display units, CRT displays and etc. Out of these display devices Liquid Crystal Display Unit (LCD) is best suitable for the said application.

LCDs are available in different shape and sizes like 16x2 and 16x4 characters, 20x1 and 20x2 characters and 40x2 characters. Typically 16x2 character is most suitable for the matrix manipulation.

Basically, LCD is consisting of two special purpose registers, called as Data Register and the Control register. The LCD device, being intelligent device, needs to be initialized as per the requirement of the data to be displayed. Figure 6 gives detailed table of mode selection.

LCD device can be initializing to different modes like automatic display address increment or decrement to position the next data or digit or character. Further it can be adjusted to display the data item on single line or dual line, size of the character to be displayed and so on.

Once the pattern of the data to be displayed is decided, the data can be loaded into the control register by making the Register Select (RS) pin Low, on each downbeat of signal on Enable (E) pin, the data is stored into the Control Register.

Command	Binary								Hex
	D7	D6	D5	D4	D3	D2	D1	D0	
Clear Display	0	0	0	0	0	0	0	1	01
Display & Curser Home	0	0	0	0	0	0	1	X	02 OR 03
Character Entry Mode	0	0	0	0	0	1	I/D	S	04 OR 07
Display On/Off & Curser	0	0	0	0	1	D	U	B	08 OR 0F
Display/Curser Shift	0	0	0	1	D/C	R/L	X	X	10 OR 1F
Function Set	0	0	1	8/4	2/1	10/7	X	X	20 OR 3F
Set CGRAM address	0	1	A	A	A	A	A	A	40 OR 7F
Set Display Address	1	A	A	A	A	A	A	A	80 OR FF

I/D: 1 = Increment, 0 = Decrement. R/L: 1 = Right Shift, 0 = Left Shift.
S: 1 = Display Shift on, 0 = off. 8/4: 1 = 8-bit Interface, 0 = 4-bit Interface.
D: 1 = Display on, 0 = off. 2/1: 1= 2 Line Mode, 0 = 1 Line Mode.
U: 1 = Curser Underline on, 0 = off. 10/7: 1 = 5x10 dot Format, 0 = 5x7 dot Format.
B: 1 = Curser Blink on, 0 = off.
D/C: 1 = Display Shift, 0 = Curser Move. 'X' => Don't care

Figure 6: Functional Table

After successful initialization of the LCD, the data can be displayed by making RS High on each downbeat of signal on Enable **(E)** pin; the data is displayed on the screen.

6. PROPOSED ALGORITHM

The process of matrix manipulation can be initiated by initializing the LCD display unit to following mode:

a. 8-bit Interface Mode,

b. Double Character Line Mode with 5x7 Dot Matrix

c. Display Curser On and

d. Increment next character display address automatically.

For initializing the LCD unit to the above mode data pattern of 38H, 0CH and 06H can be input to the control register with Register Select pin as Low. The above pattern can be loaded into the RS Register by each trailing edge of the pulse on Enable (E) pin. After such initialization the LCD unit is ready for the data to be displayed.

The proposed system is designed around FPGA, considering the hardware limitations it is possible to perform following four operations:

a. Transpose
b. Addition
c. Multiplication and
d. Subtraction.

Any of the above modes can be selected to perform the operation. For performing the matrix transpose it is required to enter only one matrix but for remaining all three operations it is required to enter both the matrix as an input.

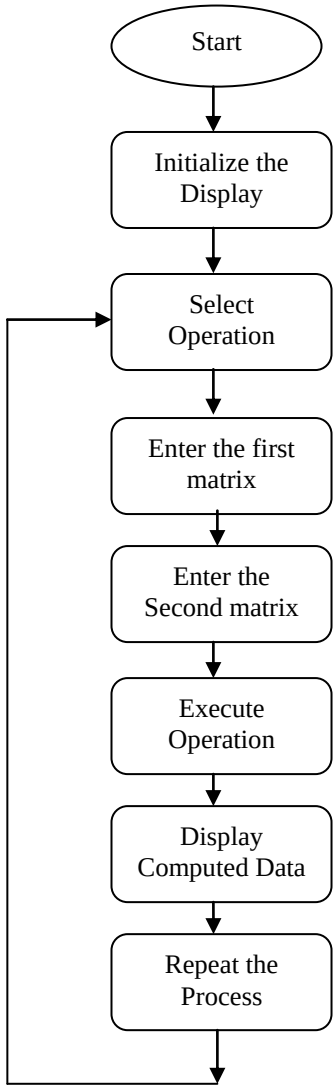

Figure 7: Proposed Algorithm

The data once available inside the FPGA, it can be operated to get the desired operation by operating it using suitable algorithm for matrix manipulation.

Algorithm can be designed to operate the Hex data directly or data can be first converted to decimal values and then operate on the converted data. It is also possible to convert the incoming data into binary format or octal format and then operate on the new data.

For any of the converted data manipulation algorithm data needs to be converted into ASCI value for displaying on the LCD unit.

7. OBSERVATIONS

7.1 Key-Pad Scanning

```
if c = "1110" then
        reg <= "00110000";
elsif c = "1101" then
        reg <= "00110001";
elsif c = "1011" then
        reg <= "00110010";
else
        reg <= "00110011";
end if;
```

7.2 Encoder

```
when "00110000" =>
            bcd <= "0000";
when "00110001" =>
           bcd <= "0001";
when "00110010" =>
             bcd <= "0010";
when "00110011" =>
             bcd <= "0011";
when "00110100" =>
             bcd <= "0100";
when "00110101" =>
             bcd <= "0101";
when "00110110" =>
             bcd <= "0110";
when "00110111" =>
             bcd <= "0111";
```

7.3 Stack Unit

```
when 0 =>
             regnumber <= "0011000000110000";
when 1 =>
             regnumber <= "0011000000110001";
when 2 =>
             regnumber <= "0011000000110010";
when 3 =>
             regnumber <= "0011000000110011";
when 4 =>
  regnumber <= "0011000000110100";
when 5 =>
             regnumber <= "0011000000110101";
when 6 =>
             regnumber <= "0011000000110110";
when 7 =>
             regnumber <= "0011000000110111";
```

7.4 LCD Initialization

```
when 0 =>
   lcd <= "00111000";
               rs <= '0';
               en <= '1';
               state <= 1;
when 1 =>
               en <= '0';
               state <= 2;
when 2 =>
   lcd <= "00001100";
               en <= '1';
               state <= 3;
when 3 =>
               en <= '0';
   lcd_state <=4;
when 4 =>
   lcd <= "00000110";
               en <= '1';
               count <= 0;
               state <= 5;
when 5 =>
               en <= '0';
               count <= 0;
               state <= 6;

when 6 =>
   lcd <= "00000001"; -- reset;
               en <= '1';
               lcd_state <= 7;
when 7 =>
               en <= '0';
               lcd_state <= 7;
end case;
```

7.5 Synthesis Results

Target Device: xc2s100-5-pq208

Summary:
- inferred 2 Finite State Machine(s).
- inferred 1 ROM(s).
- inferred 3 Counter(s).
- inferred 39 D-type flip-flop(s).

Macro Statistics
- ROMs: 1
- 32x16-bit ROM: 1
- Counters: 3
- 19-bit up counter: 2
- 5-bit up counter: 1
- Registers: 7
- 1-bit register: 3
- 16-bit register: 1
- 4-bit register: 1
- 8-bit register: 2

Optimization of FSM state on signal state [1:34] with One-hot encoding.

State	Encoding
000000	00000000000000000000000000000001
000001	00000000000000000000000000000010
000010	00000000000000000000000000000100
000011	00000000000000000000000000001000
000100	00000000000000000000000000010000
000101	00000000000000000000000000100000
000110	00000000000000000000000001000000
000111	00000000000000000000000010000000
001000	00000000000000010000000000000000
001001	00000000000000100000000000000000
001010	00000000000001000000000000000000
001011	00000000000010000000000000000000
001100	00000000000100000000000000000000
001101	00000000001000000000000000000000
001110	00000000010000000000000000000000
001111	00000000100000000000000000000000
010000	00000001000000000000000000000000
010001	00000010000000000000000000000000
010010	00000100000000000000000000000000
010011	00001000000000000000000000000000
010100	00010000000000000000000000000000
010101	00100000000000000000000000000000
010110	01000000000000000000000000000000
010111	10000000000000000000000000000000
011000	00000000000000000000000100000000
011001	00000000000000000000001000000000
011010	00000000000000000000010000000000
011011	00000000000000000000100000000000
011100	00000000000000000001000000000000
011101	00000000000000000010000000000000
011110	00000000000000000100000000000000
011111	00000000000000001000000000000000
100000	00000000000000010000000000000000
100001	00000000000000100000000000000000

LUT RAM:

ram_type	Block
Port A	
aspect ratio	32-word x 16-bit
mode	write-first
clkA	connected to signal <clk> rise
weA	connected to internal node high
addrA	connected to signal <count>
diA	connected to internal node
doA	connected to signal <regnumber>
optimisation	speed

Advanced HDL Synthesis Report

Macro Statistics
- FSMs: 2
- RAMs: 1
- 32x16-bit single-port block RAM: 1
- Counters: 3
- 19-bit up counter: 2
- 5-bit up counter: 1
- Registers: 58
- Flip-Flops: 58

8. CONCLUSIONS

Considering the market segment, different manufacturers supports different processors or co-processors that can be used to compute the complex algorithms. As compared to that the present system is more users friendly and demonstrates each and every step of execution of the complex algorithm.

System is designed around high computing, high performance gate arrays that are extensively used for integrating parallel architecture and furnishes outstanding performances when operated at frequencies in the range of and above 100 MHz.

As per the discussion under the section 7 it is obvious that the proposed system uses minimum hardware resources. The first-rate programming techniques results One-Hot encoding that reduces switching of state to certain states.

9. REFERENCES

[1] Kuan Jen Lin, Yi Tang Chiu and Shan Chief Fang,"Design Optimization and Automation for Secure Cryptographic Circuits", 22nd International conference on VLSI, 5-9 Jan.2009 , New Delhi,India.

[2] FPGA prototyping by VHDL Examples: Xilinx Spartan-3 Version, e-book, 2008.

[3] Arifur Rahman, "FPGA based design and applications", Springer edition, July 2008.

[4] Morris Mano, "Digital Logic and Computer Design-Advanced Digital Design fundamentals and Issues" Prentice Hall , 2003.

[5]http://www.xilinx.comPreliminaryproductspecificationDS077-1(v1.0).

[6] XILINX available at http://www.xilinx.com DS312.

[7]XILINX available at http://www.xilinx.com, JTAG Programmers Guide.

[8] ALLDATASHEET.COM is the biggest online electronic component datasheets search engine. http://www.alldatasheet.com/datasheetpdfpdf/197436/XILINX/XCF01S.html.

[9]http://www.national.com In System Programming Communication Protocol, revision 2.

[10] http://www.itu.dk/courses/ISOM/E2005/ARMv6 Architecture.pdf

[11]Matrix Operations for Image Processing http://www.graficaobscura.com/matrix/index.ht ml

[12] klabs.org http://klabs.org/richcontent/Tutorial/MiniCourses/architecture_logic_mapld2001/Architecture Section/07_PLD_Architecture.PDF

[13] PLA (programmable logic array) available at http://tams-www.informatik.uni hamburg.de/applets/hades/webdemos/42-programmable/10-pla/pla.html

[14] The Datasheet Archive http://www.datasheetarchive.com/XC2S100PQ2 08-5C-datasheet.html

[15] http://www.national.com In System Programming Communication Protocol, revision 2.

[16] PIC Tutorial Nine - HEX Keypad available at http://www.winpicprog.co.uk/pic_tutorial9.htm

[17] Dot Matrix Liquid Crystal Display Controller/Driver http://www.adafruit.com/datasheets/HD44780.pdf