International Journal of Innovation Scientific Research and Review
Vol. 05, Issue, 09, pp.5124-5131, September 2023
Available online at http://www.journalijisr.com
SJIF Impact Factor 2023: 6.599

Research Article

ISSN: 2582-6131

DESIGNING AND IMPLEMENTING A SIGNED MULTIPLIER RADIX-2 USING BOOTH'S ALGORITHM

[1], *Ali Othman Albaji, [2]Ali Hakami , [3]Marwan Ali Abu Aneiza, [4]Husam Mohammed Ahmed Banwair, [5]Abdelnaser Omran , [6]Abdusalama Daho

[1]Dept. of Electronics and Telecommunications at the higher college of science and technology which belongs to the ministry of technical & vocational Education, Souk Al-Jumaa, Tripoli, Libya. And a member in the Dept. of Telecommunication Software and Systems (TeSS) Research Group. Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia.
[2]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia And Computer Tech, Samath College of Technology, Vocational Training Corporation(TVTC), Saudi Arabia.
[3]Teaching assistant engineer at the Higher Institute of Science and Technology Msallatah.
[4]Master of philosophy . Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia.
[5]Faculty of Engineering Sciences, Bright Star University, Brega city, Libya.
[6]Dept. of Electrical and Electronic, Sebha University Faculty of Engineering.

Received 09th July 2023; Accepted 10th August 2023; Published online 25th September 2023

ABSTRACT

One of the most critical components of electronic design is the computation of multiplication. Numerous techniques are used to design multipliers, which can provide high speed and low power consumption, especially in signal processing. The booth multiplier is commonly utilized to increase the performance of a digital system while reducing the number of parts produced. This research aims to design and implement a digital system binary model machine This module is to convert BCD input from the keyboard to Binary. It stores the two numbers to be multiplied in registers X & Y. When the equal button is pressed it gives a start signal to Booth's Multiplier to start calculating the result. This is the part that multiplies the binary numbers in X & Y and puts the result in the ANS register. Its FSM and other things are explained in a later section. The output interface is to put the binary number in the ANS register on the LCD in BCD format. The booth algorithm is used to design the 16-bit hardware for the multiplier in this study. In addition, the LCD controller and PS/2 keyboard interface have been successfully tested.

Keywords: LCD, BCD, ANS, Booth Multiplier, Digital System, FSM.

INTRODUCTION

The algorithm created by Booth is a double-complemental multiplication method [1], that multiplies two binary numbers in 2's complement notation. He used desk calculators to increase the speed of his algorithm, which is of interest to those studying computer architecture. The Booth algorithm can simplify the process of multiplying binary numbers in signed 2's complement by minimizing the number of additions and subtractions required. It works by shifting strings of 0 in the multiplier without adding any additional value. The multiplier requires no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m. The booth algorithm must examine the shifting and addition of the partial product's multiplier bits. Multiplicands may be added, subtracted, or left unchanged depending on the rules [2].

1. When a partial product has encountered the first least significant value in a string of 1's in the multiplier, the multiplicand will be subtracted from it.
2. When the first 0 of the partial product is encountered, the multiplicand is added. This is done if there was a previous "1" in the string of 0s in the multiplier.
3. The product does not change even if the multiplier bit is the same as the previous one.

***Corresponding Author: Ali Othman Albaji,**
1Dept. of Electronics and Telecommunications at the higher college of science and technology which belongs to the ministry of technical & vocational Education, Souk Al- Jumaa, Tripoli, Libya. And a member in the Dept. of Telecommunication Software and Systems (TeSS) Research Group. Faculty of Engineering, Universiti Teknologi Malaysia, Johor Bahru, Malaysia.

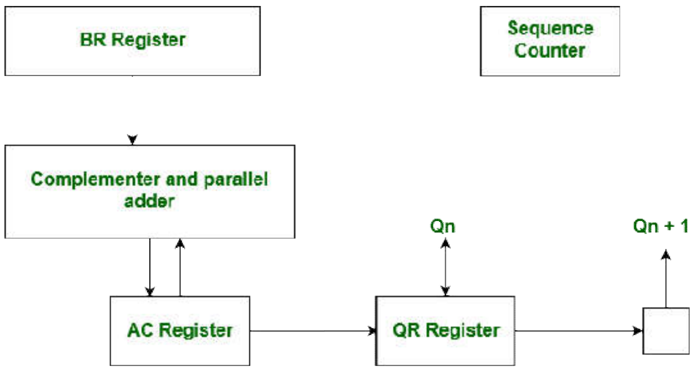

Figure 1. Booth's Algorithm Flowchart

Basically, this research focuses on designing an FPGA implementing a 16-bit integer multiplier using the radix-2 booth algorithm. An external PS/2 board and a 16x2 LCD are used to design the integer multiplier hardware for real-time verification. Here, the PS/2 keyboard serves as an input interface, allowing the user to instruct the machine to carry out tasks like loading data, multiplying, and displaying. Meanwhile, the LCD serves as an output interface, displaying helpful messages and then displaying the necessary information. Binary multiplication is one of the important arithmetic operations in digital circuits like microprocessors, microcontrollers, or FPGA devices. Different multiplications techniques are used for signed/unsigned multiplication like sequential, Shift and add, Array and Booth multiplier, etc. The booth Algorithm is one most used algorithms for the implementation of two's complement signed multiplication. In this approach, it computes multiplication using additions/subtractions and arithmetic shift operations. The radix-2 Booth multiplier flow chart is shown in Figure 1 above [3,4].

BOOTH MULTIPLIER

A lot of high-performance electronic components, such as digital signal processors, microprocessors, and filters, rely on the ability of multipliers to perform at a high level. The performance of a system is affected by the multiplier's efficiency. In addition, it is typically the slowest component in the system, and it consumes a lot of area. Due to the complexity of the design of the multiplicand, optimizing its speed and area is a major challenge. This issue usually occurs when the two constraints are conflicting. One of the most common ways to improve the performance of the multiplicand is by implementing a new algorithm called the Booth multiplier. This method takes advantage of the multiple advantages of the conventional multiplicand algorithm, such as the ability to reduce the number of steps required to produce the result. PS/2 is an IBM Personal System interface protocol for keyboards and is compatible with computer systems connected via a 5-pin or 6-pin connector shown in Figure 2. The keyboard is powered with a 5V source and ground connection along with a clock and a data line for communication protocol. After powering on, the keyboard goes with a self-initialization of internal data. After initialization, the keyboard can communicate pressed key information over the PS/2 interface [5].

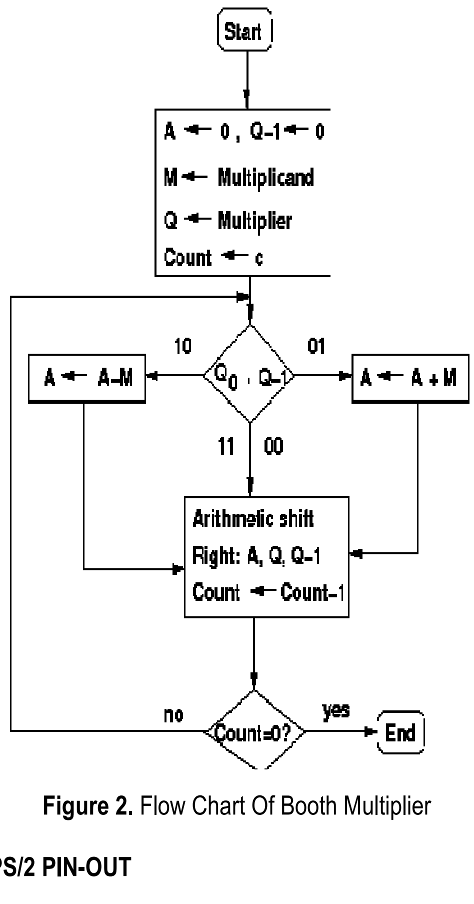

Figure 2. Flow Chart Of Booth Multiplier

A. IBM PS/2 PIN-OUT

The longevity of IBM's PC architecture can be attributed to its ability to support various software updates. This has played a significant role in the evolution of both server and personal computer systems [2].A keyboard has a large matrix of keys that are monitored by an onboard computer. The specific processor that is used to control the keyboard varies depending on the model. The goal of this process is to monitor which key is being pressed and which is being released and send the appropriate information to the host. The main processor of the computer is responsible for debouncing and buffering the data in the 16-byte buffer. The keyboard controller is also responsible for decoding all of the data that the keyboard sends to the host. All of this is done through a protocol known as IBM as shown in Figure 3 below the IBM PS/2 PIN-OUT [6,7].

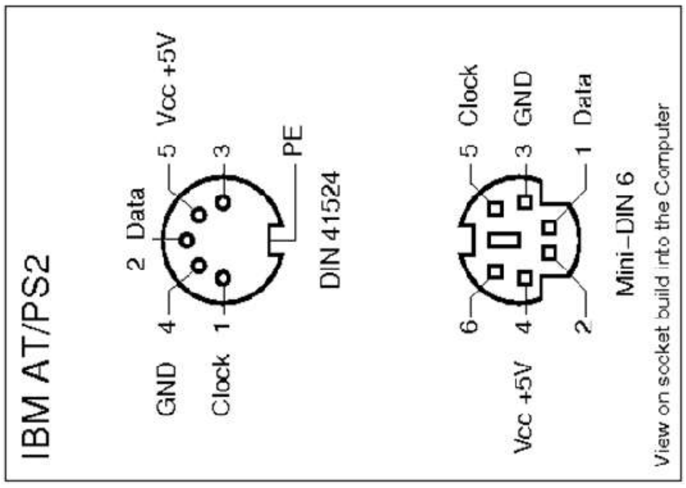

Figure 3. IBM PS/2 PIN-OUT

B. TIMING DIAGRAM OF PS/2 KEYBOARD

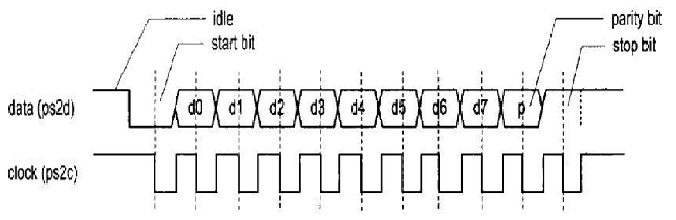

Figure 4. Timing Diagram Of PS/2 Keyboard

The data transaction format is shown above[8,9], the timing diagram in Figure3 Both clock and data signals are logic level high when the keyboard is inactive. Any key on the keyboard is pressed, and then it transmits 11-bit data. The data begin with a start bit (logic low), followed by one byte of data, a parity bit, and finally a stop bit (logic high). First, LSB data is transmitted, and then each bit should be read on the falling edge of the clock signal. After completion of pressed key information, both the clock and data signals return to logic high. The PS/2 Keyboard transmission diagram is shown below. The 8-bit data byte represents part of a keyboard output scan code and returns to a high value after key release. The PS/2 keyboard scan codes are given in Figure 4 above.

C. MAKEUP CODE OF PS2 KEYBOARD

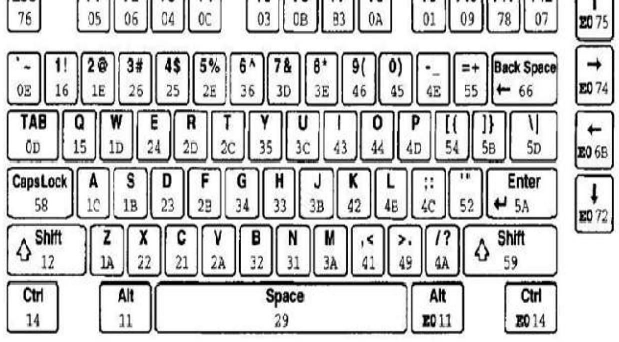

Figure 5. Makeup Code Of PS2 Keyboard

LCD is Liquid Crystal Display it will display the character data and 16x2 LCD is shown in Figure 5. Input to the LCD is the ASCII (American Standard Code for Information Interchange) value of the character. This ASCII data is 8bit. Any 16×2 LCD can display in two rows and 16 characters per row. The 16×2 LCD can be interfaced with FPGA to control the display. LCD has 8 input bits and 3 control signals (RW, RS, En), and VCC and GND [10,11,12].

D. 16x2 LCD DISPLAY

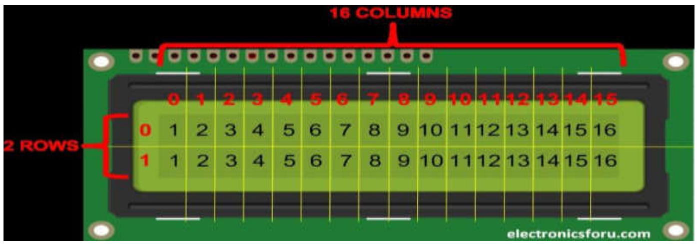

Figure 6. 16x2 LCD Display [13]

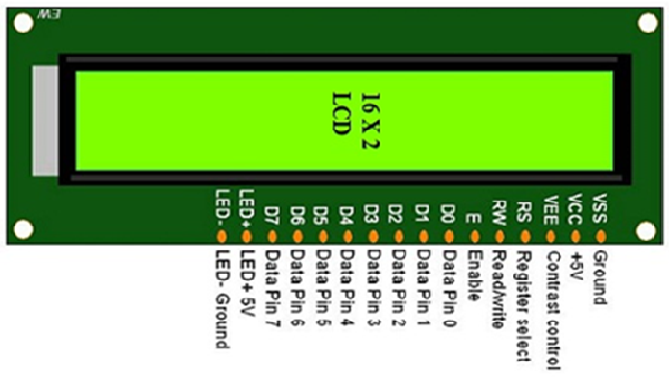

Figure 7. 16x2 LCD Display [14]

LCD display works in two modes: 8-bit and 4-bit. In 8-bit display mode, It uses 8 data pins for receiving data. In 4-bit display mode, it uses only 4 data pins for receiving data. In LCD, it has two registers: data and command register. The data-on-data pins will store in the data or command register depending on RW pin data. If RW=0, it selects the command register. And RW=1 for the data register. The command register stores the commands given to LCD. Based on the command, it will perform specific tasks like initializing, clearing, and setting the position of the display. Hex codes of the different commands are shown in Table 1. The ASCII information of character data in data registers will display on LCD. ASCII codes of the numerical characters are also shown in Figures 6 and 7 above [15].

STANDARD LCD COMMANDS AND FUNCTIONALITY

Below is a list of standard LCD commands. V0 is used to set the contrast. A potentiometer and variable resistor can be attached to this pin to set the exact contrast. To illuminate the LCD's backlight, supply +5v to ground and LED+ as can be seen in Table 1.

Table 1. Standard LCD Commands And Functionality

S. N	HEX CODE	COMMAND TO LCD INSTRUCION REGISTER
1	0x01	Clear display screen
2	0x02	Return home
3	0x04	Decrement cursor (shift cursor to left)
4	0x06	Entry mode: Increment cursor (shift the cursor to the right)
5	0x05	Shift display right
6	0x07	Shift display left
7	0x08	Display off, cursor off
8	0x0A	Display off, cursor on
9	0x0C	Display on, cursor off
10	0x0E	Display on, cursor blinking
11	0x0F	Display on, cursor blinking
12	0x10	Shift the cursor position to the left
13	0x14	Shift the cursor position to the right
14	0x18	Shift the entire display to the left
15	0x1C	Shift the entire display to the right
16	0x80	Force cursor to beginning (1st line)
17	0xC0	Force cursor to beginning (2nd line)
18	0x30	8-bit 1 Line and 5x7matrix
19	0x38	8-bit 2 Line and 5x7matrix
20	0x20	4-bit 1 Line and 5x7matrix
21	0x28	4-bit 2 Line and 5x7matrix

DESIGN DESCRIPTION

The design of the research's model can be explained in the following steps in sequence

A. TOP-LEVEL IOBD DIAGRAM

The top-level input and output interface diagram is shown in Figure 7. The PS2 Keyboard receiver module receives the data from the PS/2 keyboard. The output of the PS2 keyboard is loaded into a shift register serially at negedge clk and then data remains unchanged until the key is pressed. The shift register has a scam code and converts it into BCD format and it into the X & Y registers. First data is stored in the X register if * is pressed and Second data is stored in the Y register if # is pressed on the keyboard. Then Booth Multiplier module is used to multiply the data in X & Y registers. When the equal to the button is pressed, Booth's Multiplier receives a start signal and starts the multiplication using the Booth algorithm. LCD Display Controller takes the X, Y, and Output of Booth multiplier and sends them to the LCD display unit for display. 16x2 LCD Display Controller unit initializes the LCD module and also displays the data. Display data selection is based on Control signals from the controller unit. It also converts 16-bit X, Y, and 32-bit Z data into an 8-bit binary ASCII data format for the LCD display. It controls the display unit using RW, RS, and E control signals.

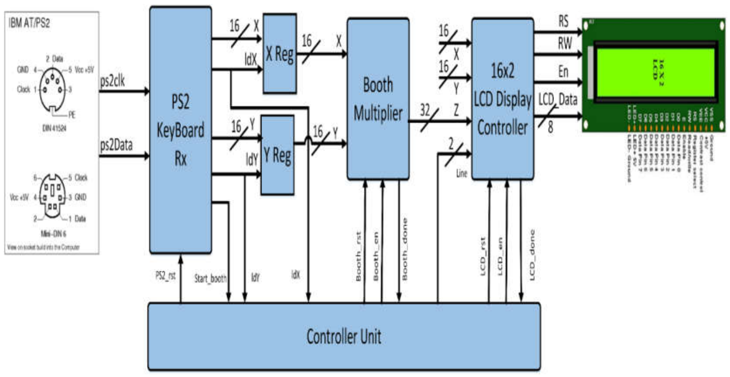

Figure 8 Top-Level IOBD Diagram

B. BOOTH MULTIPLIER ARCHITECTURE

The booth multiplier algorithm ASM chart machine is shown in Figure 8. This booth algorithm is an iterative algorithm and iterates for no. of input bit times. Its hardware requires addition, subtraction, and arithmetic right shift also counter. The 16-bit inputs init X and init Y are appended with zeros as per the ASM chart and are stored into 33-bit registers X and Y. Final output of the multiplication is 32-bit which is X[32:1]. The LSB (Least Significant Bit) bit of X is not considered in the final result. The data path unit and Control logic unit is designed for the radix-2 booth algorithm. The control flow of hardware checks the LSB two bits X1 and X0. If the two bits are the same (00 or 11) then all of the bits of Accumulated data in x in are shifted 1 bit to the right (Arithmetic right shift) as shown in the ASM chart. If two bits are not the same and if the combination is 10 then the Y is subtracted from X and if the combination is 01 then the Y is added with X. 4-input is used along with an adder and a subtractor is used to perform this operation. In both cases, results are loaded into register Q or x in and after the addition or subtraction operation, the Arithmetic right shift is performed on Q. In the arithmetic right shift operation where the LSB bit Q1 is shifted into Q1 and also A32. The final result of the multiplication will update in the Q [16,17].

ALGORITHM DESCRIPTION (ASM)

The design of finite state machines using the ASM method is commonly used to visualize the various components of digital circuits. Although it is similar to a state diagram, the ASM method is less formal and easier to understand. A typical flowchart is used to represent decision paths and procedural steps in algorithms, though time relations are not included. A state machine diagram is a representation of the sequence of events that happen as a sequential controller moves between steps. An algorithmic state machine diagram provides several advantages over a conventional diagram.

1. ASM diagrams, are easier to interpret in larger state diagrams,
2. Conditions for a proper state diagram, are usually automatically satisfied.
3. ASM diagrams can be easily converted to other forms.

One of the most important features of ASM diagrams is that they do not display all the possible outputs and inputs of a given state. This ensures that the correct inputs and outputs are identified as shown in Figure 9.

- The positive logic signals that are high are asserted.
- The negative logic signals that are low are asserted.

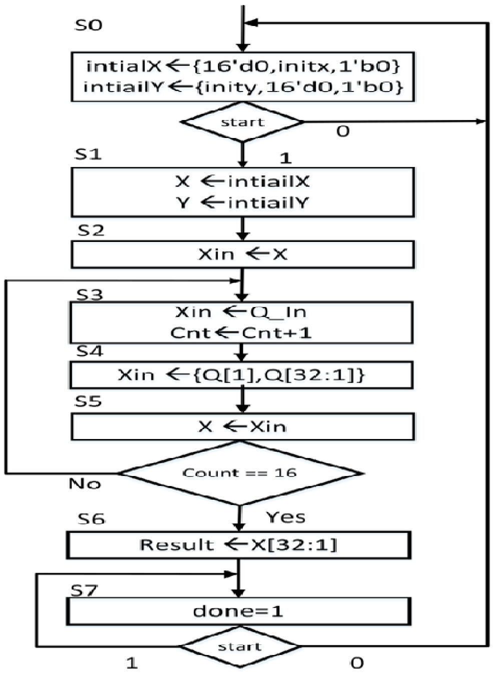

Figure 9 Algorithm Description (ASM)

STATEMACHINEDIAGRAM

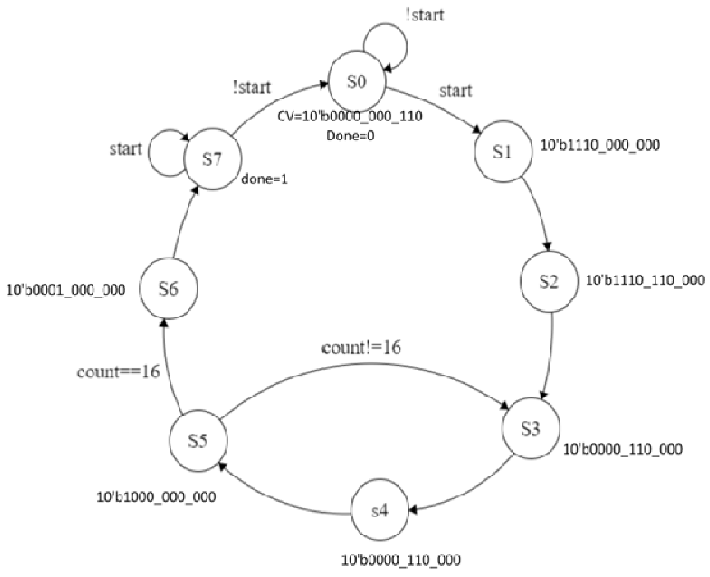

Figure 10 State Machine Diagram

Booth's algorithm can be implemented in many ways. The hardware is designed using a controller and a Data path are shown in Figure 9. The operations in the data path unit are controlled by the control signal and control signals are generated from the controller unit (FSM). The State Machine diagram is shown in figure 10 above.

DATA AND CONTROL PATH UNITS

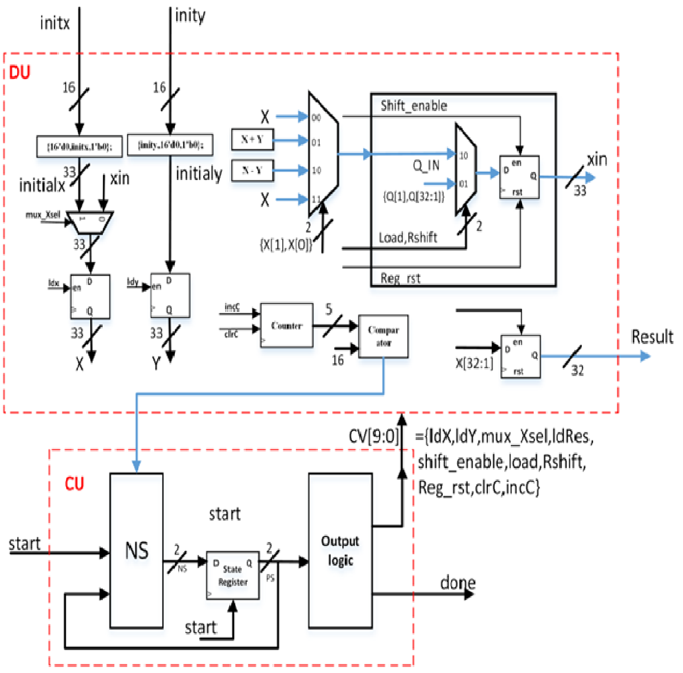

Figure 11 Data And Control Path Units

LdX, ldY and ldRes are used to update data into X, Y, and Result registers. Reg_rst is used to reset X, Y, Result, and Shift Register Output Q/Xin. Shift_enable, Load, Right_shift are control signals for the Right shift register. If shift enable and Load is high and then input Q_IN loaded into Q. If shift enable, Load is low and Right_shift is high then Arithmetic right shift is performed on Q. clrc and incC are Counter control signals. After Count=16 the result is updated with X [32:1].

PS/2 KEYBOARD HARDWARE

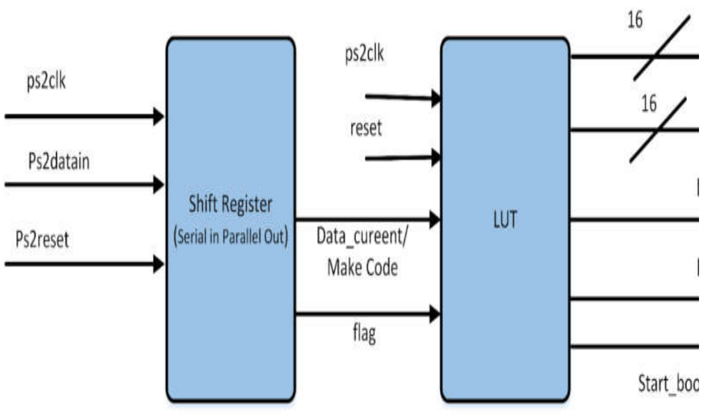

Figure 12 PS/2 Keyboard Rx Interface & Logic Architecture

This PS2 keyboard Rx takes the input from the PS/2 keyboard serially and stores data in the output register X, Y. The PS/2 Keyboard Rx interface & logic hardware architecture is shown in Figure 10. It has a serial-in parallel-out shift register, a Lookup table, two 16-bit registers, and three 1-bit registers. The serial-in parallel-out shift register takes serial data at the negative edge of ps2clk and stores data after a 1-bit shift operation. PS/2 keyboard scan code is available in the shift register. LUT converts this scan code i.e., shift data into BCD data and stores information at the relevant register.

The PS2 Keyboard communication protocols timing diagram is shown in Figure 3. When any is pressed then the Keyboard sends CLK and data serially. PS2 Keyboard Rx ASM chart is shown in Figure 11. PS2 keyboard ASM chart represents the communication protocol. It has 3 states: START, DATA, and PARITY state. When reset is pressed then it enters into the START state and waits for keyboard press i.e., negative edge CLK & ps2datain==0(start bit). If any key is not pressed then ps2clk & ps2datain are maintained high continuously. When the key is pressed then ps2clk==0 and ps2datain==0 and stater enter into the DATA state. And the state is in the DATA state for up to 11 clock cycles (i.e., 8 clock cycles). During the DATA state, ps2datain loads into the serial-in parallel-out shift register at each negedge of ps2clk. The state remains in a DATA state for 11 clock cycles. The counter is used in the design to count the number of ps2clk clock pulses. After 9 clock cycles, the state enters into PARITY state. The Scan code is fully loaded into the shift register and connected to the LUT for scan code to BCD format conversion. Also, it checks for which *, #, and = keypress to store and to state booth multiplier. And then in the next clock (11th clock cycle), the state goes to START state and again waits for a keypress.

PS/2 KEYBOARD RXASM CHART

Figure 13 below illustrates the steps of the PS/2 KEYBOARD RX ASM CHART

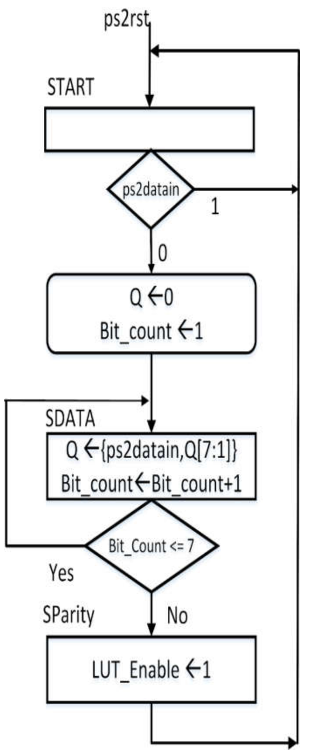

Figure 13 PS/2 Keyboard RX ASM Chart

LCD DISPLAY CONTROLLER HARDWARE

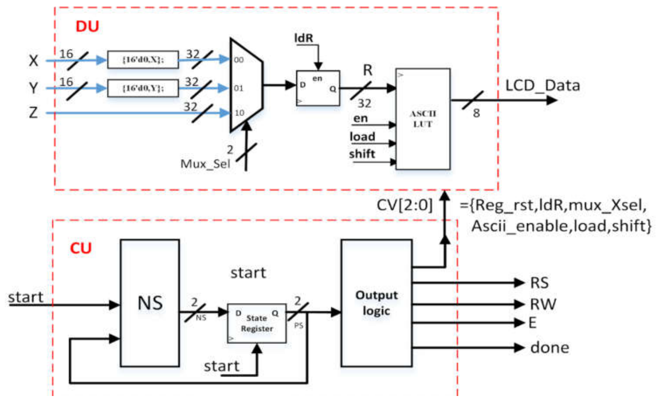

Figure 14 Lcd Display Controller Hardware

The LCD display controller hardware architecture is shown in figure 10. It has a multiplexer to select input data X, Y, and Z for the LCD display. Shift register selects LSB 4bits i.e., BCD digits data and shifts right by 4bits. The ASCII Lookup module converts it into 8-bit ASCII data format. In any two LCD data i.e., LCD command or character, a delay is required. To generate the delay, the Counter is designed and controlled using control signals generated from the controller. The controller is designed using a Finite state machine.

LCD display controller has 6 states: IDLE, SEL_IN, LCDCMD_INIT, WAIT, LCDCMD_LINE, LCD_DISPLAY. By reset, FSM state enters into IDLE state and stays same state until lcd_start signal is high. In the SEL_IN state, multiplexer selects among the X, Y and Z based on the key pressed information. In the LCDCMD_INIT state, LCD display initializes by sending command like clear the display (01h), Return to home (02h), entry mode (06h), display on and cursor blinking (0Eh), selection of 2-line LCD 8-bit, display data in Line 1(80h) and display data in Line 2(C0h). If LCD is not displaying anything then it is in WAIT state until receive display or command signal. TO send the LCD command RS=0(LCDCMD_INIT state), RW=0 and En=1. To display data in LCD, RS=1(LCD_DISPLAY state), RW=0 and En=1. When E=1 then wait for 10us delay and then En=0 again wait for 1000 us to send new data or command as shown in figure 14 above.

SIMULATION RESULTS AND ANALAYSIS

A. BOOTH'S ALGORITHM

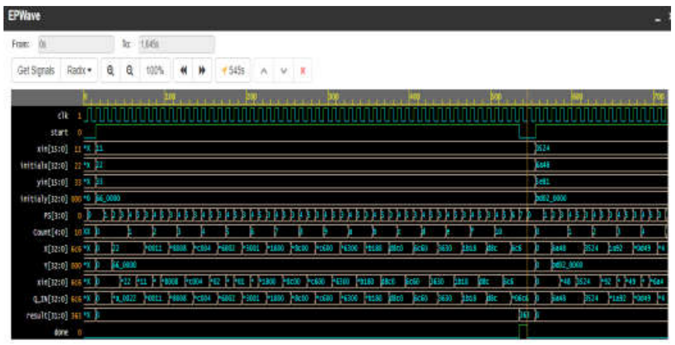

Figure 15 Simulation Results Of Booth's Algorithm

For the multiplier first Input X=17=16'h0011=0000_0000_0001_0001 then 33-bit x as Initialx={16'd0,initx,1'b0} = 0,0000, 0000, 0000, 0000, 0000,0000,0010,0010=0 0000 0022

For the multiplier second Input Y=51=16'h0033= 0000_0000_0011_0011 then 33-bit Y as initialy={inity,16'd0,1'b0} =0,0000,0000,0110,0110,0000,0000,0000,0000=0 0066 0000.

Table 2. Output Of Hardware Simulation Results

Counter	X	X1X0	Add/Sub	Y	Shift Input	Shifter Output
1	0 0000 0022	10	sub	0 0066 0000	1 FF9A_0022	1 FFCD_0011
2	1 FCD_0011	01	Add	0 0066 0000	0 0033_0011	0 0019_8008
3	0 0019_8008	00	Add(Only subtraction)	0 0000 0000	0 0019_8008	0 000C C004
4	0 000C C004	00	Add	0 0000 0000	0 000C C004	0 0006 6002
5	0 0006 6002	10	Sub	0 0066 0000	1 FFA0_6002	1 FFD0_3001
6	1 FD0_3001	01	Add	0 0066 0000	0 0036 3001	0 001B 1800
7	0 001B 1800	00	Add	0 0000 0000	0 001B 1800	0 000D 8C00
8	0 000D 8C00	00	Add	0 0000 0000	0 000D 8C00	0 0006 C600
9	0 0006 C600	00	Add	0 0000 0000	0 0006 C600	0 0003 6300
10	0 0003 6300	00	Add	0 0000 0000	0 0003 6300	0 0001 B180
11	0 0001 B180	00	Add	0 0000 0000	0 0001 B180	0 0000 D8C0
12	0 0000 D8C0	00	Add	0 0000 0000	0 0000 D8C0	0 0000 6C60
13	0 0000 6C60	00	Add	0 0000 0000	0 0000 6C60	0 0000 3630
14	0 0000 3630	00	Add	0 0000 0000	0 0000 3630	0 0000 1B18
15	0 0000 1B18	00	Add	0 0000 0000	0 0000 1B18	0 0000 0D8C
16	0 0000 0D8C	00	Add	0 0000 0000	0 0000 0D8C	0 0000 06C6
17	0 0000 06C6					

when counter = 17, the 33-bit X has loaded with 0 0000 06C6 and Final result is 32-bit output and Result = 0000, 0000, 0000, 0000, 0000,0011,0110,0011= 0000 0363.In the above booth multiplier simulation figure, the output of hardware simulation results are matched with the theoretical step-by-step result.

B. PS2 KEYBOARD

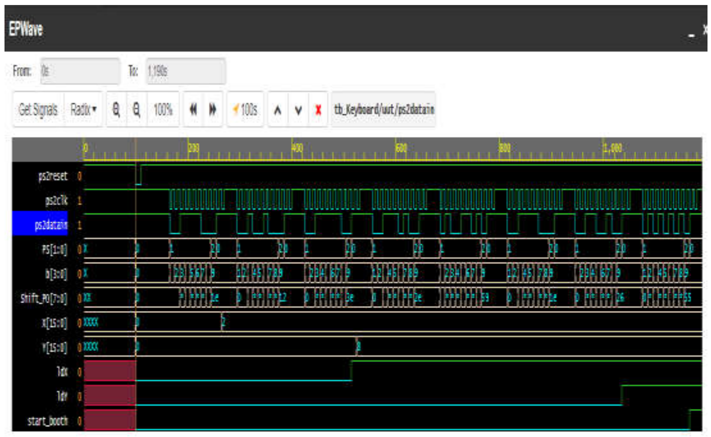

Figure 16 Simulation Results Of Ps2 Keyboard

The PS2 keyboard receiver simulation waveform is shown in the above figure. Hardware is designed to detect the key press and to check valid keys. It will detect the following key: 0,1,2,3,4,5,6, 7,8, 9,A,B,C,D,E,F keys along with *,#, and = keys. If any other key is pressed then it will not detect means it will not a valid data and not be stored in any register. After pressing the first key (0,1,2,3,4,5, 6,7, 8, 9,A,B,C,D,E,F), Rx waits for the * key to be pressed then it stores into the X memory. After the second key press, the Rx waits for the # key. After Y stores into the register, waits for the = key press. If = key is pressed then the controller unit generates the booth multiplier hardware start signal.

C. LCD DISPLAY CONTROLLER

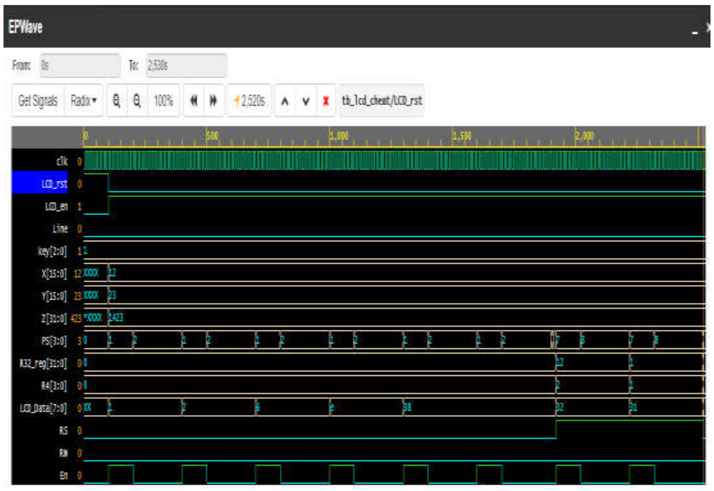

Figure 17 Simulation Results Of Lcd Display Controller

LCD display controller simulation waveform is shown in the above figure 1.16 Hardware is designed to initialize the LCD display and to display the BCD data. LCD display initializes by sending commands like clear the display (01h), Return to home (02h), entry mode (06h), display on and cursor blinking (0Eh), selection of 2-line LCD 8-bit, display data in Line 1(80h) and displays data in Line 2(C0h). During LCD command mode RS=0, RW=0, and E=1. If C is pressed on the keyboard, then C equivalent BCD data 12 is stored into X. Then it will display 12 data by sending ASCII codes of 1 and 2 as 8’h31 and 8’h32. During LCD display mode, RS=1, RW=0, and E=1.

LCD display BCD data

I. Ex. Z=3456
II. It first send’s 3, ascii code of 3=33
III. And 4, ascii code of 3=34
IV. And 5, ascii code of 3=35
V. And 6, ascii code of 3=36
VI. LCD display controller first initializes the LCD

   1. During initialization, RS=0, RW=0, and En=0
   2. display (01h),
   3. Return to home (02h),
   4. entry mode (06h),
   5. display on and cursor blinking (0Eh),
   6. selection of 2-line LCD 8-bit,
   7. display data in Line 1(80h) and
   8. displays data in Line 2(C0h).

VII. While displaying the data input in X is displayed in LCD

   9. X=12
   10. First data loaded into R32_reg. First 1 is extracted from R32_reg
   11. Data in R4 is compared in ASCI_LUT module
   12. And corresponding character ASCII code is transferred to the LCD
   13. While displaying the LCD data
       1. RW=0
       2. RW=1
       3. En=1

D. TOP-LEVEL DESIGN

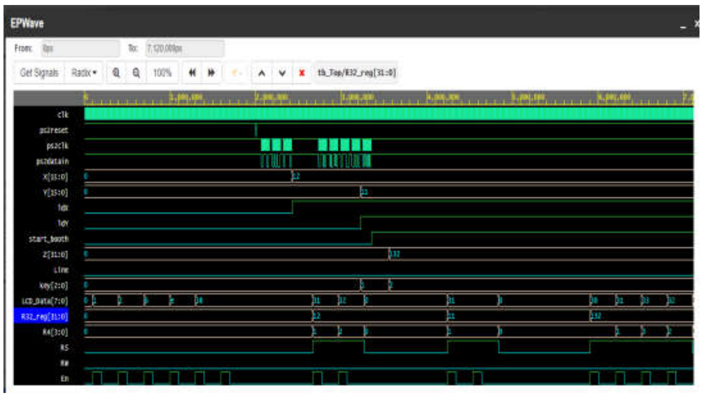

Figure 18 Simulation Results Of Top-Level Design For Multiplier Hardwar

The top-level design for multiplier hardware using the radix-2 booth algorithm simulation waveform is shown in above figure 16. Hardware is designed to initialize the LCD display and to display the BCD data. LCD display initializes by sending commands like clear the display (01h), Return to home (02h), entry mode (06h), display on and cursor blinking (0Eh), selection of 2-line LCD 8-bit, display data in Line 1(80h) and displays data in Line 2(C0h). During LCD command mode RS=0, RW=0, and E=1. If C is pressed on the keyboard, then C equivalent BCD data 12 is stored in X. Then it will display 12 data by sending ASCII codes of 1 and 2 as 8’h31 and 8’h32. During LCD display mode, RS=1, RW=0, and E=1.

a. Initially wait for LCD initialization
b. Next, if any key is pressed then data is loaded into X, Y
c. The control unit will identify the keypress
d. Then it sends the start signal to the booth multiplier to start multiplication
e. And sends the control signals to the LCD display controller to which data is to be displayed in LCD.

CONCLUSION

The standards of LCD commands have been used in this research. V0 is used to set the contrast. A potentiometer and variable resistor can be attached to this pin to set the exact contrast. To illuminate the LCD's backlight, supply +5v to ground and LED+ Multiplication is a vital component of electronic design. Various techniques are utilized to design multipliers, and they can provide high power consumption and speed. The booth multiplier can also be used to improve the performance of digital systems. The goal of this study is to develop a digital system binary machine. It converts the input BCD from the keyboard to binary and stores the two numbers in the registers X and Y. The equal button can be pressed to start the calculation of the result. The booth multiplier is a component that converts the binary numbers in X and Y into a result in the ANS register. The output of this device is sent to the LCD in a format known as BCD. The design for the 16-bit hardware is based on the booth algorithm. The integer 16-bit multiplier hardware is designed using the booth algorithm. Also, the PS/2 keyboard input interface and LCD controller are designed and tested successfully.

REFERENCES

[1] shrivastava, s., singh, j., & tiwari, m. (2011). implementation of radix-2 booth multiplier and comparison with radix-4 encoder booth multiplier. international journal on emerging technologies, 2(1), 14-16.
[2] kawahito, s., kameyama, m., & higuchi, t. (1990). multiple-valued radix-2 signed-digit arithmetic circuits for high-performance vlsi systems. ieee journal of solid-state circuits, 25(1), 125-131.
[3] vlăduţiu, mircea. computer arithmetic: algorithms and hardware implementations. springer science & business media, 2012.
[4] roth, charles, lizy k. john, and byeong kil lee. digital systems design using verilog. cengage learning, 2015.
[5] http://vlabs.iitkgp.ernet.in/coa/exp7/index.html
[6] https://cse.iitkgp.ac.in/~chitta/coldvl/booth.html
[7] https://www.allaboutcircuits.com/technical-articles/how-to-interface-mojo-v3-fpga-board-16x2-lcd-block-diagram-verilog-code/
[8] https://www.futurlec.com/led/lcd16x2bla.shtml
[9] https://ieeexplore.ieee.org/abstract/document/1291430
[10] https://ieeexplore.ieee.org/abstract/document/1291430
[11] https://ieeexplore.ieee.org/abstract/document/5228240
[12] http://www.wseas.us/e-library/conferences/malta2001/papers/193.pdf
[13] https://d1wqtxts1xzle7.cloudfront.net/43080016/fpga-with-cover-pagev2.pdf?expires=1643637911&signature=oa6m5nltydo049ht1suufoxuyzrjmnhmgnk8beibco1srn4ueoc9pxiuxdp8shciuje1cnn5v7pkkd64hjd4x75kv4a8qd5s2bgb3aqjz-31dby-skzzar7pkv3cijf66lj3hsqfkcqgtylp3cctufalkcm0pyezqf5okhcyzmodmfacaxahuuxz3pxqsry1ulmy3oxkfovgcq2rruqsvnagzw8krmytl9r7gpacqhhdpuzmlt~ntn~zumozle5-~fxuisqzcra17tulatt4rxia74cqly8ym16byzb2r7slv9yoj-ugkgjysy3v1zdevzqj~l7ctd-xme2l41kuq__&key-pair-id=apkajlohf5ggslrbv4za

[14] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.300.3261&rep=rep1&type=pdf

[15] https://www.researchgate.net/profile/akkureshi/publication/296673364_hardware_implementation_of_configurable_booth_multiplier_on_fpga/links/56d7bbcc08aebabdb4030bfd/hardware-implementation-of-configurable-booth-multiplier-on-fpga.pdf

[16] https://books.google.com.my/books?hl=en&lr=&id=bphudf6clgc&oi=fnd&pg=pp8&dq=booth%27s+algorithm+hardware+implementation.+advanced+digital+design&ots=9rvpzcmv0u&sig=lmqjinnwpka-m1n9cwziz1zasgs&redir_esc=y#v=onepage&q&f=false

[17] https://ieeexplore.ieee.org/abstract/document/7020607

****