Design and implementation of a fast sequential multiplier based on iterative addition architecture

Ahmed Salah Hameed¹,² and Marwa Jawad Kathem¹

¹Department of Computer Engineering, College of Engineering, University of Diyala, Baquba, Iraq
²E-mail: ahmedhameed_eng@uodiyala.edu.iq

Abstract. In this paper, a fast design and implementation for sequential multiplier is presented. The suggested approach of implementation incorporates a definition of iterative addition that reduces the number of additions required in calculating the product of two binary numbers. The proposed implementation of sequential multiplier eliminates all shift operations required by conventional sequential multiplier to only one shift operation with the final accumulated result. Proposed and conventional designs of sequential multiplier are simulated in Quartus II synthesis software tool using Verilog implementation. According to the simulation results, the proposed implementation of sequential multiplier is better than conventional implementation in terms of delay time and power consumption. The proposed sequential multiplier shows an average improvement of 17.15% in delay time compared to conventional sequential multiplier.

1. Introduction
Multipliers are one of the basic units used in implementing different simple and complicated digital circuits. Digital multipliers are the heart of many devices and applications used in our daily life. Since multipliers consume large area and power of implementation, optimization in their design can play a key role in optimizing the speed and area of digital circuits such as Digital Signal Processing DSP digital communication systems and any other circuit use multiplier in its structure [1, 2]. Multipliers occupied the core of different operations such as convolution, cross correlation, and filter implementation which are mainly used in DSP processes and applications [3, 4]. Different algorithms have been used in implementing the circuit of multiplication. Some of the algorithms simulate the process of doing the multiplication by hand and other uses special algorithms to implement the process of multiplication.

The hardware implementation of digital multiplier can be classified in to combinational multiplier design and sequential multiplier design. Combinational multipliers are the direct and basic version of multiplication. Most of combinational multipliers such as Array multipliers, Wallace Tree Multiplier, and Booth multipliers mimic the basic definition of multiplication in which a number of add and shift operations to find the final product. Sequential multipliers are basically using a single circuit of addition to accumulate the product of multiplication. Sequential multipliers are smaller in area than combinational multipliers.

The aim of this research is to provide a fast and low power design for sequential multiplier based on a new look to the basic definition of multiplication. The rest of the paper is organized as follows. Section 2 presents some of the standard and basic designs of multipliers. The proposed
implementation of sequential multiplier is described in Section 3. Finally, in section 4 the simulation results are discussed followed by conclusions in section 5.

2. Existing digital multipliers
In this part some of the standard and basic designs of multipliers are presented. Multipliers such as basic array multiplier, wallace tree multiplier, booth multiplier, and conventional sequential multiplier could be considered as the base technique in developing the implementation of digital multipliers.

2.1. Basic Array multiplier
Array multipliers employ an array of adders to represent the direct implementation of manual multiplication. Figure 1 shows the general implementation for 4-bit basic array multiplier. A straightforward layout can be easily generated to represent array multipliers for hardware implementation in which the basic add and shift algorithm is used in the design. This type of multipliers is easy in implementation, but requires a big area of design that increases proportionally with the increase in size of the multiplied operands.

2.2. Wallace Tree Multiplier
Wallace Tree Multiplier is a tree based implementation for multiplication operation. It uses carry save addition CSA to reduce the delay time of multiplication. In tree multiplier, the multiplication is done with three stages. It starts with bit products calculation, then the minimization of bit products rows into two rows through the use of CSA, and end with calculating the final result of multiplication by adding the two bit products rows generated with CSA [2, 5].

2.3. Booth multiplier
Booth algorithm is one of the famous approaches used in the design of a reduced area, high speed, and low power design for multipliers. Three units are used in booth multiplier to complete the process of multiplication, which are: the decoder unit, the unit for generating partial product, and adder unit.
Although the use of booth algorithm participates in enhancing the multiplier design in term of speed, a large number of partial products are required to complete the process of multiplication [6 - 9].

2.4. Conventional sequential multiplier

Although the combinational multipliers have an easy structure of implementation and take less time to complete the process of multiplication, their area of design increases with the increasing in size of multiplied numbers. Sequential multipliers are the implementation option to have a reduced area design for multiplier.

Conventional version of sequential multiplier is implemented through the generating of partial products of multiplication with sequenced steps in a way mimics the manual implementation of multiplication. Sequential steps are used to do an accumulative addition to partial products generated by shift left (with the multiplicand) and shift right (with the multiplier) [10]. Figure 2 shows the general datapath structure for 8-bit sequential multiplier in its conventional implementation [11].

![Figure 2. Datapath structure for 8-bit sequential multiplier (conventional implementation) [11].](image)

A controller unit is used to generate all signals used by sub-modules of datapath unit. Even though the implementation of sequential multiplier reduces the area of design by avoiding the use arrays of adders, but the process of multiplication is still slow. The time consuming process in conventional version of sequential multiplier is related with the number of sequential shift operations required to find the product of two numbers.

3. Proposed sequential multiplier

The idea behind the suggested implementation of binary multiplier is using the very basic definition of multiplication. The multiplication can be basically defined as a repetitive addition in which one of the two multiplied numbers (multiplicand) will be added to itself a number of times equal to the value of the second number (multiplier). Using the definition as it requires a number of additions equal to the value of the multiplier and that is a time consuming operation. The number of repetitive additions could be reduced to half by using the following proposed equations:

\[
Y \times X = 2 \times (X \times (Y \div 2)) + (X \times (Y \mod 2))
\]  

(1)
According to equation 1, the number of repetitive additions for X×Y will be equal to (Y DIV 2) + (Y MOD 2). The value of (Y MOD 2) in equation 2 is equal to 1 for odd numbers and it is equal to 0 for even numbers, i.e. the value of (Y MOD 2) is equal to the value of Least Significant Bit (LSB) in Y. The proposed equations eliminate all shift operations required by conventional sequential multiplier to only one shift operation with the final accumulated result followed by one conditional addition based on the value of the LSB. To convert the idea behind the proposed equations (1 and 2) to a hardware implementation doing the multiplication of two binary numbers (X and Y) the following steps are used:

1. The process of finding (Y DIV 2) is done by taking the Most Significant Bits MSBs of Y without the LSB Y[0] and the process of finding (Y MOD 2) is done by taking the value of LSB Y[0]. The two processes for 8-bit numbers are shown in Figure 3 below.

   ![Figure 3](image)

   **Figure 3.** The process of finding (Y DIV 2) & (Y MOD 2).

2. The process of finding (X×(Y DIV 2)) is done by adding Y in an accumulator R a number of times equal to (Y DIV 2).

3. The accumulated value in R is multiplied by 2 by shifting it to left one time (add 0 to the LSB of R and shift bits to the left one time).

4. Final product equals to R + X when Y[0] = 1 and equals to R when Y[0] = 0.

The proposed multiplier is designed as a sequential multiplier according to the hardware architecture shown in Figure 4 below.
Figure 4. Hardware architecture of the proposed sequential multiplier.

The hardware shows the internal structure for 8-bit of the proposed sequential multiplier which consists of the following units:

1- Two 8-bit registers to hold the data input values for the multiplicand and multiplier (i.e. X & Y). The two registers have the control signals; ‘Load XY’ to store data in to the registers, and ‘Clear’ to clear the content of the registers. One more control signal is used with the multiplier register named ‘Dec’ to count down based on the content of the MSB (7-bit) of register.

2- 16-bit binary adder to do the repetitive addition operations.

3- 16-bit register R [15:0] to hold the accumulated value of repetitive additions. This register has the control signals; ‘load R’ to store accumulated data in register, ‘clear’ to clear the content of register, and ‘ShiftL’ to shift the content of register to the left.

4- Control signals including: ‘Zero’ signal is active when the value of the seven MSBs of Y equal to zero. ‘Odd’ signal is active when the value of LSB of Y equal to 1. A signal named ‘Done’ is used to indicate the end of multiplication process.

Figure 5 shows the ASM chart of the proposed multiplier. The basic steps needed to do the multiplication of two 8-bit numbers are summarized below. A start signal is used to decide the time of starting the multiplication. Three states of control units are working as follows:
Figure 5. ASM chart of the proposed sequential multiplier.

State 0 (S0): It is the state at which accumulator register R is cleared and external data for X, and Y is loaded to registers X and Y when start signal is zero. The control unit continues working at state 0 as long as start signal equal to zero and the movement to next state is done when start signal becomes one.

State 1 (S1): It is the state at which we are adding X to R, and decrementing Y [7:1] by one when ‘Zero’ signal is low. As long as ‘Zero’ signal is low, the process of addition and decrementing continue through setting of ‘LoadR’ and ‘Dec’ signals. When ‘Zero’ is high the content of R register is shift to left by setting ‘Shiftl’ signal. One more addition is done when ‘Odd” signal is 1 through setting of ‘LoadR’ and the movement to next state appears.

State 2 (S2): this is the last state at which the multiplication is done by generating a control signal ‘Done’ and no change in this state is recorded as long as start equal to 1. When start equal to zero the control unit moves back to reset state to process a new operation of multiplication.

4. Simulation Results
The designs of 8-bit size multipliers (including the design of both proposed and conventional sequential multiplier) are synthesized with Verilog implementation of Quartus II synthesis software
tool. The performance analysis, including time delay, power dissipation, and number of logic elements is recorded with the use of the Altera FPGA EP2C5T144C6 device (Cyclone II).

Structural design implementations of two components for conventional and proposed sequential multipliers were built using Quartus II Verilog HDL synthesis. A top-level component is built to instantiate all the lower level components of each multiplier. The designs were implemented and synthesized based on the structures explained previously in Figure 2 and Figure 4 (structure of conventional and proposed sequential multipliers respectively).

Table 1 shows the comparison between the proposed and conventional sequential multipliers in term of delay time (worst case), power consumption, and number of FPGA blocks (logic elements) respectively.

Table 1. Simulation results for the proposed and conventional sequential multipliers.

|                        | Worst case delay time (ns) | Total power (mW) | Total logic elements |
|------------------------|-----------------------------|------------------|---------------------|
| Conventional sequential multiplier | 7.959                       | 32.97            | 46                  |
| Proposed sequential multiplier | 6.594                       | 32.83            | 53                  |

Based on the simulation results, our proposed design shows better performance than the conventional sequential multiplier in terms of delay time and power consumption. Results of simulation show that the proposed design has a slightly larger number of logic elements, but the delay time, and power consumption are significantly reduced. Since the standard design of adder (i.e. Ripple Carry Adder RCA) is used in implementing the accumulator unit of the proposed multiplier, the area of the proposed multiplier is slightly bigger than the area of conventional multiplier. The classification of the proposed and conventional sequential multipliers in term of delay time, power consumption, and number of FPGA blocks is shown in Figure 6.

Figure 6. Comparison results of proposed and conventional sequential multiplier.

5. Conclusions
In this paper, a new approach is proposed for implementing a fast sequential multiplier. The new approach comes up with a new implementation for sequential multiplier that eliminates all shift operations required by conventional sequential multiplier to only one shift operation with the final accumulated result. Conventional and proposed designs of sequential multiplier are simulated in
Quartus II synthesis software tool using Verilog implementations. Results of simulation demonstrate that the proposed design of the sequential multiplier is faster than the conventional design. The proposed sequential multiplier is found to have a reduction in delay time of conventional design by 17.15%. As a future work reducing the total logic elements of the implementation could be achieved by applying more enhancements to the design of adder that occupied most of the area of the proposed multiplier.

References

[1] Jhamb M, Garima and Lohani H 2016 Design, implementation and performance comparison of multiplier topologies in power-delay space *Engineering Science and Technology, an International Journal* 19 355–63

[2] Sabeetha S, Ajayan J, Shriram S, Vivek K and Rajesh V 2015 A study of performance comparison of digital multipliers using 22nm strained silicon technology *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*

[3] Salman E H 2013 Fast convolution and correlation algorithm for multidimensional signals *Diyala Journal of Engineering Sciences* 6 75–89

[4] Mogheer H S 2018 A new technology for reducing power consumption in synchronous digital design using tri-state buffer *Diyala Journal of Engineering Sciences* 11 60–6

[5] Qian L, Wang C, Liu W, Lombardi F and Han J 2016 Design and evaluation of an approximate Wallace-Booth multiplier *2016 IEEE International Symposium on Circuits and Systems (ISCAS)* 1974–7

[6] Prabhu A S and Elakya V 2012 Design of modified low power booth multiplier *2012 International Conference on Computing, Communication and Applications*

[7] Kuang S-R, Wang J-P and Guo C-Y 2009 Modified Booth Multipliers With a Regular Partial Product Array *IEEE Transactions on Circuits and Systems II: Express Briefs* 56 404–8

[8] Govekar D and Amonkar A 2017 Design and implementation of high speed modified booth multiplier using hybrid adder *2017 International Conference on Computing Methodologies and Communication (ICCMC)*

[9] Sukowati A I, Putra H D and Wibowo E P 2016 Usage area and speed performance analysis of booth multiplier on its FPGA implementation *2016 International Conference on Informatics and Computing (ICIC)*

[10] Patterson D A and Hennessy J L 2018 Chapter 3 Arithmetic for Computers *Computer organization and design the hardware/software interface RISC-V* (Cambridge, MA, MA: Elsevier, Morgan Kaufmann Publishers) pp 177–81

[11] Navabi Z 2005 Sequential Multiplier *Digital Design and Implementation with Field Programmable Devices* ed Z Navabi (Boston, MA: Springer US) pp 229–45