FPGA design, simulation and prototyping of a high speed 32-bit pipeline multiplier based on Vedic mathematics

Shuja Ahmad Abbasi\textsuperscript{1a)}, Zulhelmi\textsuperscript{2}, and Abdul Rahman M. Alamoud\textsuperscript{1}

\textsuperscript{1} Department of Electrical Engineering, King Saud University, Riyadh 11421, Saudi Arabia
\textsuperscript{2} Department of Electrical Engineering, Syiah Kuala University, Darussalam-Banda Aceh 23111, Indonesia; formerly graduate student at King Saud University, Riyadh, Saudi Arabia

\textsuperscript{a)} abassi@ksu.edu.sa

Abstract: This research is about a new approach, which is used for optimizing multipliers designs, which are based on the concept of Vedic mathematics. The design has been targeted to to FPGAs (state-of-the art field-programmable gate arrays). It has been assessed that the multiplier produces partial products by utilizing Vedic mathematics concept by deploying basic $4 \times 4$ multipliers, which is designed by exploiting special features of multiplexers and 6-input look up tables (LUTs) on the same slices, resulting in considerable minimization in area. The multiplier has been realized on Xilinx\textsuperscript{®} Virtex-5 FPGAs. It is significant to notice that pipeline adders were used to obtain final products. Furthermore, the multiplier is developed and organized by using pipeline schemes, which contribute to the enhancement of operating frequency of the multiplier. The results show that the 32-bit pipeline multiplier can work up to a clock frequency of 450 MHz. It has utilized 514 slices and 1157 flip-flops and has much less dynamic power than the other reported work.

Keywords: pipeline multipliers, field programmable gate arrays, digital signal processing

Classification: Integrated circuits

References

[1] J. Valls and E. Boemo: IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. 50 (2003) 317. DOI:10.1109/TCSII.2003.811438
[2] G. Lakshminarayanan and B. Venkataramani: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 13 (2005) 783. DOI:10.1109/TVLSI.2005.850086
[3] F. Dinechin and B. Pasca: FPL (2009) 250.
[4] Assady: Eur. J. Sci. Res. 26 (2009) 362.
[5] S. R. Kuang, J. P. Wang and C. Y. Guo: IEEE Trans. Circuits Syst. II, Exp. Briefs 56 (2009) 404. DOI:10.1109/TCSII.2009.2019334
[6] M. Själander and P. Larsson-Edefors: IEEE Trans. Very Large Scale Integr.
1 Introduction

Multipliers are widely used in many applications and areas such as digital signal processing (DSP), image processing, matrix multiplication, etc. With regard to DSP applications, a large number of multipliers are employed for finite impulse response (FIR) filters as well as other DSP functions. Therefore, it can be stated that a multiplier, having less area utilization, high speed, and low power dissipation, is considered to be an effective solution for DSP and many other applications. Therefore, it is significant to consider these three important factors, while designing and implementing multipliers.

A number of attempts have been made to design multipliers. Javier Valls and Eduardo Boemo [1] have introduced FPGA implementation of serial/parallel multipliers (SPMs), in order to reduce the cost of the technologies in implementing parallel circuits. In addition to this, the serial stream of data matches better with the structure of many targeted FPGAs [1]. Although the scheme presented an efficient implementation of multipliers, the speed of SPMs is not fast enough. Consequently, the SPMs are not suitable for use in new communication applications. Moreover, the feedback loop of SPMs also limits the application of pipelining, which limits the operating frequency. Similar effort has been made by Lakshminarayanan and Venkataramani [2], who used optimization techniques for FPGA, which were based on wave-pipeline multiplier. They demonstrated that their multipliers have better performance in both speed and power dissipation. However, the performance of the multiplier is suitable only when operand sizes are small. In this regard, Florent de Dinechin and Bogdan Pasca [3] have also presented their work in which, they used fewer DSP blocks to realize large multipliers. They demonstrated better performance, in terms of saving precious DSP blocks and maintaining the operating frequency.

Assady [4] has introduced a new modified booth encoding (MBE) in order to reduce partial product generation. In addition to this, he also used hybrid adders to perform fast summation in getting final product. Similar attempt was made by
Shiann-Rong Kuang et al. [5], who redesigned MBE, in order to eliminate the generation of an irregular partial product array. Their modifications have given a significant improvement in realizing a multiplier with less area, delay, and power [4, 5]. Magnus Sjalander and Per Larsson-Edefors [6] have presented a twin precision algorithm to accelerate multiplication. The implemented multiplier has low power dissipation and enables an increased computational throughput. Vassil S. Dimitrov et al. [7] introduced a multiplier, which was based on multiple radix representations. The multiplier’s structure provides better performance, in terms of area utilization. Their multipliers were implemented by using standard cells, which were based on application specific integrated circuits (ASICs). This technology takes large turn-around time to manufacture; therefore, it is not suitable in such cases when an application requires fast solution. Another work was presented by Paul et al. [8]. They introduced ROM based logic (RBL) design to realize a 16-bit multiplier. The architecture demonstrated better performance, in terms of power dissipation and speed. However, the use of area is large, when the multiplier is realized on higher bit-lengths.

2 Vedic mathematic concepts

It has been assessed that Vedic mathematics is a concept of knowledge that was used by ancient Indian scientists. The word “Veda” or “Vedic” is meant by “knowledge”. In other words, Vedic mathematics is a store of knowledge, explaining a number of mathematical branches such as algebra, trigonometry, and geometry. One of the “sutras” or “formulas”, in Vedic mathematics is “Urdhva-tiryakhbyam”. Urdhva-tiryakbyam can be termed as vertically and crosswise [9]. This formula is mainly used to perform multiplication. It has been shown to be a much more efficient multiplication algorithm as compared to the conventional counterparts [9] for VLSI implementation.

The basic multiplication process involving concept of “vertical and cross wise” using Urdhva-Tiryak Sutra in vedic multiplication is illustrated with an example in Fig. 1. The numbers to be multiplied, 46821 (the multiplier) and 58379 (the multiplicand) are written horizontally and vertically, respectively on two consecutive sides of a box as shown in the figure. The box is divided into rows and columns where each row/column corresponds to one of the digits of either a multiplier or a multiplicand. Thus, each digit of the multiplier has a small box which is common to a digit of the multiplicand. Diagonal lines are then drawn in each square. Each digit of the multiplier is then independently multiplied with every digit of the multiplicand and the two-digit product is written in the common box. For example, 5 (most significant bit) of multiplicand is multiplied by the least significant bit of multiplier, 1. The result is 05 and it is written in the top right most square. The second number of the multiplicand, 8 is multiplied by the least significant bit of multiplier, 1 and the result, 08 is placed in the square adjacent to the number 8. Similarly the numbers 3, 7 and 9 are multiplied by 1 and the results, 03, 07 and 08, respectively are placed in the squares as shown in the figure. In the same way, each bit in the multiplicand 58379 is multiplied by 2, 8, 6 and 4 and the results are stored in appropriate boxes as shown. All the digits lying on a crosswise
dotted line are added to the previous carry. Carry for the first step (i.e., the dotted line on the extreme right side) is taken to be zero. The least significant digit of the obtained number acts as the result digit and the rest as the carry for the next step. For example, the lower dotted crosswise line in the bottom right most square contains number 9. This is taken as the right most (least significant) bit of the result, with zero carry. The next dotted crosswise line contains numbers 7, 0 and 8. The sum of these numbers is 15 of which 5 is written as the next bit in the result and 1 is taken forward as carry. The process is repeated for all the dotted crosswise lines to get the final result.

In a similar way, multiplication of binary numbers can also be done. For building a high order multiplier, Vedic concept offers a simple way to realize, for example, a development of 4-bit multiplier from 2-bit multiplier. Below mentioned diagram, i.e., Fig. 2 shows a block diagram of the 4-bit multiplier, which is built by using four 2-bit multipliers.

Similarly, higher order multipliers, for instance, up to 32-bit, can be developed by using this scheme [10].
3 Proposed multiplier

A multiplier scheme with the art of Vedic mathematics concept is presented, in this research study. In designing a 32-bit pipeline multiplier, a number of lower order multipliers are required. First, it needs a basic multiplier, which has the lowest possible bit-length. Although the lowest order multiplier is a 1-bit multiplier, however, in this design, a 4-bit multiplier is selected as a fundamental or a basic multiplier [11, 12]. Secondly, 16-bit pipeline multipliers, which are developed from basic ones, are also required to build the proposed 32-bit pipeline multiplier. In addition, carry save adders (CSAs), registers, and a pipeline adder are utilized to optimize the design. Below mentioned Fig. 3 shows block diagrams of the proposed 32-bit pipeline multiplier. It may be noted that basic or 4-bit multipliers are not shown in Fig. 3. It is due to the fact that those multipliers are parts of or inside the 16-bit pipeline multipliers.

3.1 Multiplier inputs

The number of inputs, which are involved in the proposed multiplier includes, operand A[31:0], operand B[31:0], chip enable (CE), and synchronous clear (SCLR). The input A[31:0] acts as multiplicand. On the other hand, input B[31:0] is a multiplier operand. In addition to this, CE, and SCLR are controlled signals, which are used to activate and reset the multiplier.

3.2 Basic multiplier

Since, in Vedic concept, the lowest-order multipliers affect the implementation of higher order multipliers, a compact design of a 4-bit multiplier is essential to reduce the area utilization and minimize delay during place and route (PAR) process.
Therefore, the 4-bit multiplier, which is based on 6-input look up tables (LUTs) is proposed. Following figure, i.e., Fig. 4, indicates a block diagram of the basic multiplier. F0 to F4 are logical functions, which are written in verilog codes. It is significant to notice that the inside values are given to adjust the function of LUTs. It is found that the maximum possible delay is approximately equal to the delay of one LUT plus the delays of two multiplexers, MUX7 and MUX8 [12]. Since multiplexers are fast devices in FPGA fabrication, the major component of the delay in this design can be assumed only from LUT’s.

Fig. 4 shows that the critical paths are found in producing P3 up to P6. To tackle this issue, a strategy of employing the same slice or CLB is applied. This feature is fully supported in hardware description language (HDL) programming, which directs the synthesis and PAR processes as desired.

![Block diagrams of the proposed 4-bit multiplier based on 6-input LUTs.](image)

### 3.3 16-bit pipeline multipliers

A number of steps are involved in building the 16-bit pipeline multiplier. Fig. 5, shows block diagrams of the multiplier and it shows that the multiplier is built by employing lower-order multipliers. Sixteen of 4-bit multipliers are required to produce partial product generation (PPG) rows. Three types of compressor blocks, 3:2, 5:3, and 7:3, are then used to reduce PPG rows to two rows. Final adders take tasks to produce the final product of the multiplication operation. Registers insertion is placed between one block to another. This method, which is known as a pipeline scheme, is commonly used to push up the operating frequency of multipliers.
Alphabetical symbols, A and B in Fig. 5 represent operands of the multiplication operation namely, multiplicand and multiplier, respectively [13]. On the other hand, numerical symbols, 0 up to 15 indicate the weight of the bits. For example, A12 is the bit of multiplicand with weight 2^{12} when the bit input A12 is ‘1’.

3.4 Carry save adders

A number of methods of reducing partial products have been introduced [11, 12]. They demonstrated simple and complex ways, in order to achieve the best results. Some provide an advantage either in saving area or decreasing delay, while others consider both, area and delay. A simple method is used in the proposed multiplier because the partial products have been formed in such a way that it gives an easy task for partial product reduction. This simple method employs carry save adders or compressors 3:2 Bhattacharjee et al. [14]. The compressors are to reduce three rows of PPG to two rows. Fig. 6 shows a block diagram of compressor 3:2. The three bit inputs which come into compressor are of the same weight, for example 2^n, and the output, S, is also 2^n, while the output, C has higher weight 2^{n+1}.

Fig. 5. Block diagram of the proposed 16-bit pipeline multiplier.

Fig. 6. Block diagram of compressor 3:2.
3.5 Registers and pipeline adders

In order to push up or increase the operating frequency of multipliers, a pipeline scheme is applied. The technique, which is used to realize this scheme, is basically inserting registers between parts which contribute critical delay. Multipliers, generally, have critical delays in several parts. Firstly, it is found in partial product generation process. Secondly, it occurs at partial product reduction step. Lastly, it is found at the last addition process.

In a pipeline multiplier, it is necessary to manage delays of each block so as to reduce those delays to as small value as possible. The compressor delay is found to be smallest and therefore, no further reduction is generally needed since the other delays occurring in partial product generation and final addition are much larger than delays in the compressor blocks. It is significant to notice that these delays are required to be reduced, in order to gain the maximum operating frequency. The delay of partial product generation is minimized by utilizing 16-bit pipeline multipliers, while the use of the pipeline adder for final addition guarantees that its delay is proportional to others.

4 Design and simulation

The proposed multiplier was implemented on Virtex 5 FPGAs, by using ISE design suite software. A number of steps are involved, starting with writing Verilog HDL codes to describe the design. The codes are then compiled and verified to obtain ready source codes for synthesizing an RTL (register transfer level). Several constraint files are prepared to optimize the RTL before implementing the place and route (PAR) process. Physical constraint file is one of the most important files, which plays a major role in making the best hardware configuration during PAR session [14].

Below mentioned Fig. 7 indicates the simulation of sub-multiplication, 16 bit pipeline multiplier. Although the simulation was done at selected frequency of 50 MHz, the maximum frequency that can be applied is 501 MHz, as mentioned in the PAR report.

![Fig. 7. Post route simulation of the proposed 16-bit pipeline multiplier](image)

The utilization summary of the hardware is tabulated in Table I. Based on the results of the 16-bit pipeline multiplier, it is expected that the proposed 32-bit
pipeline multiplier has a performance, which is close to the 16-bit one. This expectation is logical since the 32-bit pipeline multiplier is built from four 16-bit pipeline multipliers.

| Selected Device | Type of Reports | No. of LUTs | No. of Slices | No. of FFs | Max. Frequency (MHz) |
|-----------------|-----------------|-------------|--------------|------------|---------------------|
| Virtex-5        | Synthesis       | 474         | -            | 304        | 505.51              |
|                 | Post and Route  | 446         | 134          | 304        | 501.00              |

Table I. Report summary of the proposed 16-bit pipeline multiplier.

Fig. 8, depicts the post route simulation of the proposed 32-bit pipeline multiplier. Since, latency is commonly found in multipliers with pipelining scheme, the proposed multiplier requires six periods to complete a single multiplication operation.

Report summary of the proposed 32-bit pipeline multiplier is shown in Table II. It indicates that the maximum operating frequency of the multiplier is about 450 MHz, while the utilized area is around 24% of the available slices in device XC5VLX30. The Dynamic Power is 33 nW.

Table II. Report summary of the proposed 32-bit pipeline multiplier.

| Selected Device | Type of Reports | No. of LUTs | No. of Slices | No. of FFs | Max. Frequency (MHz) |
|-----------------|-----------------|-------------|--------------|------------|---------------------|
| Virtex-5        | Synthesis       | 1971        | -            | 1157       | 505.51              |
|                 | Post and Route  | 1823        | 514          | 1157       | 453.92              |

A closer look at the proposed multiplier reveals that the data reach the output pads after 7.3 ns as shown in Fig. 9, which is the maximum OFFSET OUT in the constraint file. Forcing this offset to a lower value decreases the operating
frequency. In fact, the OFFSET OUT is adapted to the application for which the multiplier is used. If the multiplier is used along with other functions on the same FPGA, the OFFSET OUT does not give any effect on the speed. However, the speed of the multiplier decreases when it is connected to other devices Bhattacharjee et al. [14].

5 Prototyping

The design was downloaded to the Xilinx XC5VLX30 FPGA belonging to Virtex-5 family using the Xilinx Virtex-5 prototype platform. The downloading involves a number of steps. First, the ISE design suite performs the physical design with a number of user constraints like defining the pin locations of various inputs and outputs and the timing constraints. The software then generates a bit stream file containing instructions for downloading the design into the chip. Finally the design is downloaded to the chip using JTAG–USB–cable. An elaborate testing scheme is developed involving design of test vectors. The testing is ultimately done using a 32-bit logic analyzer. The blocks of 4-bit, 8-bit, 16-bit and 32-bit multipliers were separately tested. It was observed that all the results were in agreement with the simulation results. This was expected also since the Xilinx ISE package is highly advanced and is widely used [10]. The ISE simulator is well validated and produces highly accurate simulation results. A comparison with some of the results reported in literature is presented.

Below mentioned Table III shows the comparison of the proposed 8-bit multiplier with the one, which was reported by Edirisuriya et al. [12]. Although the proposed multiplier was implemented with a slightly higher bit-length, the speed and area utilization are better compared to the reported one.

![Close examination of the proposed 32-bit pipeline multiplier](image)

**Table III.** Maximum achievable frequency and occupied slices (without pipelining)

| Multiplier width | Multiplier type       | Max. clock frequency (MHz) | No. of slices | Ref. |
|------------------|-----------------------|-----------------------------|---------------|------|
| 7-bit            | Karatsuba-MRM         | 120                         | 702           | [12] |
| 8-bit            | Proposed Multiplier   | 202                         | 70            |      |

Fig. 9. Close examination of the proposed 32-bit pipeline multiplier
A pipelined scheme has been applied to increase the operating frequency. The use of the scheme is required because multipliers usually have large delays because of complicated routing and the complexity of the circuit. For example, without pipelining, 32-bit multipliers, based on FPGA design, can be operated at around 100 to 150 MHz clock frequency depending on selected devices. On the other hand, multipliers with pipelined scheme can be pushed to operate at about 454 MHz clock frequencies as shown in Table IV.

Table IV. Comparison of maximum frequency and area in the 32-bit multiplier with and without pipelining

| Multiplier type       | Max. clock frequency (MHz) | No. of LUTs | No. of slices |
|-----------------------|----------------------------|-------------|--------------|
| Non-pipeline multiplier | 64                        | 1696        | 433          |
| Pipeline multiplier   | 454                       | 1823        | 514          |

Table V shows the comparison of 16-bit multiplier with the ones reported by Bulic et al. [13]. A comparison of results of various 32-bit multipliers, the basic multiplier (BM), the carry save multiplier (CSM), the carry ripple multiplier (CRM) and the booth sign multiplier (BSM) reported by Bhattacharjee et al. [14] are shown in below mentioned figure, i.e. Fig. 10.

Table V. Area usage and maximum delay of 16-bit pipeline multipliers

| Multiplier width | Multiplier type | Max. clock frequency (MHz) | No. of slices | FFs | Pipelining scheme |
|------------------|-----------------|-----------------------------|---------------|-----|-------------------|
| 16-bit           | BB Multiplier   | 153.33                      | 216           | 170 | Yes               |
|                  | BB-ECC          | 153.33                      | 427           | 306 | Yes               |
|                  | Proposed        | 501                         | 134           | 304 | Yes               |

Fig. 10. Area usage and maximum delay of 32-bit pipeline multipliers.

It has been observed that, when the bit-length of a multiplier is increased, for example, from 16-bit to 32-bit, the delay is generally doubled. The 32-bit multipliers reported in the studies of [14], for example, have the delays about twice those
of the 16-bit multipliers. In contrast, only a small increase in delay is found in the 32-bit proposed multiplier (PM). In the case of area occupation, PM occupies middle position among the multipliers. It is fair to sacrifice a number of small slices to achieve the best performance.

A comparison of dynamic power with the results, which were reported by Bhattacharjee et al. [14], has also been made. Fig. 11 shows that the proposed multiplier gives far better results.

![Comparison of dynamic power.](image)

**Fig. 11.** Comparison of dynamic power.

### 6 Conclusions

The present work addresses a new approach for design and hardware implementation of multipliers based upon the concept of Vedic mathematics. The technique is applied on sub-multiplication blocks, which are the fundamental blocks of the Vedic multiplier. The sub-multiplication blocks are optimized by employing 6-input LUTs and multiplexers within the same slices or CLBs. The pipeline scheme and the pipeline adders are employed. By setting placement and design goal strategies, better performance, in terms of speed compared to the multipliers reported in literature has been achieved. In terms of the device utilization also, better results are obtained in most of the cases.

### Acknowledgment

This project was funded by the National Plan for Science, Technology and Innovation (MAARIFA), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award Number (11.NAN-2118-02).