

HIGH SPEED SQUARER

Chandra Mohan Umapathy

Senior IC Design Engineer
Celstream Technologies Private Limited
Prestige Blue Chip, Block II
#9, Hosur Road, Bangalore: 29

chandra.mohan.umapathy@celstream.com

Abstract: This paper proposes a novel architecture for modular, scalable & reusable hybrid squaring circuit. Comparison is made between different implementations of squaring circuit. The implementation results show a significant improvement in performance in terms of area, power & timing.

The paper is organized as follows: the first section gives a brief introduction of the different aspects & the motivation for the High Speed Squarers. The second section introduces the proposed algorithm. The third section explains the proposed architecture. The fourth section deals with the comparison of the proposed architecture with the existing method & conclusion.

Key Words: *Squarer, Squaring Circuit, Multiplier, Low Power etc.*

SECTION 1: Introduction:

Squaring is one of the frequently performed functions in most of the DSP systems. Squaring is a special case of multiplication. Squaring circuit forms the heart of the different DSP operations like Image Compression, Decoding, Demodulation, Adaptive Filtering, Least Mean Squaring etc.

Traditionally, squaring was performed using multiplier itself. As the applications evolved & the demand for the high speed processing increased, special attention was given for squaring function & dedicated squarers were proposed & implemented.

Initially, squaring was performed using Look Up Table approach if delay was primary concern, while trading of the area constraint. Major drawback of the scheme was area penalty, which increases exponentially as the number of input bits increases. Due to the cost & the interconnect delay this approach was not most preferred implementation method till now. Recently, with the evolution of VLSI Process Technology the above method of implementation is becoming more popular.

Recently, lot of research has been conducted in order to develop different methodologies to implement squarers, giving more importance to improve delay & reducing area constraints. Due to which a new scheme was developed to compromise the above-mentioned trade-offs, which is called Hybrid Squarers. Greater emphasis is given on Hybrid Squarers, which comprises of Memory Elements & Computing Logic.

It implies from the above discussion that in order to achieve better optimization, challenge for the designers is to reduce area & power, while keeping the timing intact.

SECTION 2: Proposed Algorithm:

This section describes the proposed algorithm to implement Hybrid Squarer, based on the Vedic Sutra. Special emphasis is given on Modularity, Scalability & Reusability aspects of the design. The algorithm consists of following steps:

- The given input is partitioned into two parts, each part is treated as a separate unit processed individually by further units.
- Find the square of each part.
- Find twice the product of individual part.
- Add the above results suitably to get the final result.

If X is a two-digit number, whose square has to be computed.

$X = ab$.

Find square of $a = a^2$.

Find square of $b = b^2$.

Find twice the product of a & $b = 2ab$.

Find the sum of the above results to get the square of X.

Ex:

1. Let $X = 12$. $a = 1$, $b = 2$.

Find square of $a = a^2 = 1$.

Find square of $b = b^2 = 4$.

Find twice the product of a & $b = 2 * 1 * 2 = 4$.

Find the sum of the above results to get the square of $X = 144$.

2. Let $X = 14$. $a = 1$, $b = 4$.

Find square of $a = a^2 = 1$.

Find square of $b = b^2 = 16$.

Find twice the product of a & $b = 2 * 1 * 4 = 8$.

Find the sum of the above results to get the square of $X = 196$.

Important Note: Since, we are handling one digit at a time; if the resultant of any of the above operation is more than one digit, consider the higher order digit/s as carry to next stage.

2.1 Salient Features:

- It is Modular & Scalable architecture.
- Since we are handling one digit at a time:
 - Easy & simple to implement.
 - Low Power consumption.
 - Less Area.
 - Better Timing can be achieved.

The above concept can be extended to any number as explained below:

If X is a three-digit number, whose square has to be computed.

$X = abc$.

Find square of a = a^2 .

Find square of b = $(bc)^2$.

Find twice the product of a & bc = $2(a)(bc)$

Find the sum of the above results to get the square of X.

Ex:

1. Let X = 121. a = 1, b = 21.

Find square of a = $a^2 = 1$.

Find square of bc = $(bc)^2 = 441$.

Find twice the product of a & b = $2 * (1) * (21) = 42$.

Find the sum of the above results to get the square of X = 14641.

Important Note: Since, we are handling two digits at a time; if the resultant of any of the above operation is more than two digits, consider the higher order digit/s as carry to next stage. Hence the above operation is equal to adding $10000+4200+441 = 14641$.

Notice that the one shift is equivalent to multiplying by 100, since we are considering two digits at once.

If X is a four-digit number, whose square has to be computed.

$X = abcd$.

Find square of ab = $(ab)^2$.

Find square of cd = $(cd)^2$.

Find twice the product of ab & cd = $2(ab)(cd)$

Find the sum of the above results to get the square of X.

Ex:

1. Let X = 1234. a = 12, b = 34.

Find square of ab = $(ab)^2 = 144$.

Find square of cd = $(bc)^2 = 1156$.

Find twice the product of ab & cd = $2 * (12) * (34) = 816$.

Find the sum of the above results to get the square of X = 1522756.

Important Note: Since, we are handling two digits at a time; if the resultant of any of the above operation is more than two digits, consider the higher order digit/s as carry to next stage. Hence the above operation is equal to adding $1440000+81600+1156 = 1522756$.

Notice that the one shift is equivalent to multiplying by 100, since we are considering two digits at once.

From the above analysis, the proposed algorithm is proven if X is a 3-digit & 4-digit number. Consider an arbitrary number X which is a 5-digit number.

If X is a five-digit number, whose square has to be computed.

$X = abcde$.

Find square of abc = $(abc)^2$.
 Find square of de = $(de)^2$.
 Find twice the product of abc & de = $2(abc)(de)$
 Find the sum of the above results to get the square of X.

Ex:

1. Let X = 12345. a = 123, b = 45.
 Find square of abc = $(123)^2 = 15129$.
 Find square of de = $(45)^2 = 2025$.
 Find twice the product of abc & de = $2 * (123) * (45) = 11070$.
 Find the sum of the above results to get the square of X = 152399025.

The above theory can be extended for any given number X. Hence, by mathematical inspection; the proposed algorithm is proven for any arbitrary number X.

SECTION 3: Proposed Architecture:

The proposed architecture is depicted in figure 1.0. It consists of following blocks:

- Square.
- Multiply.
- Add.

3.1 Square: This block is used to compute the square of the partitioned input.

3.2 Multiply: This block is used to compute the twice the product of individual parts of the input.

3.3 Add: This block is used to compute the addition of the above results.

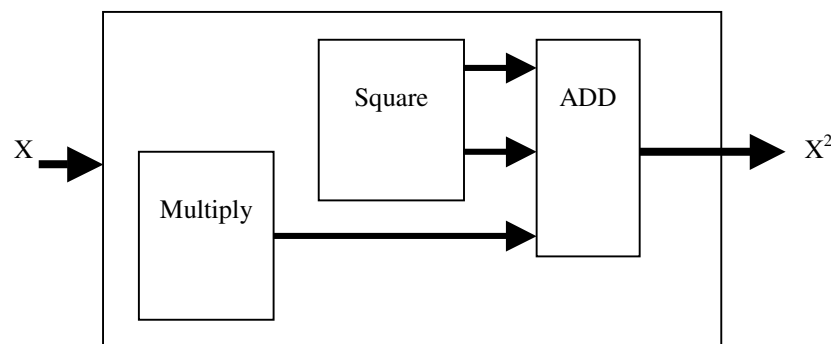


Figure 1.0: Block diagram of the Proposed Architecture.

SECTION 4: Comparison:

The proposed architecture is compared with the Synopsys Design Ware Component. Both existing & proposed schemes for implementing squarers was coded using VHDL & synthesized using Synopsys Design Analyzer (version

5.0). The results are tabulated as shown in tables 1.0 & 1.1:

Squaring Circuit results

Number Of Bits	Normal Method	Proposed Method	Improvement
4	Clk = 5.69 ns A = 192 T = 0.0 ns P = 23.4195 uw	Clk = 5.64 ns A = 183 T = 0.0 ns P = 18.8016 uw	A = 4.68 % T = 0.8 % P = 19.72 %
8	Clk = 11.55 ns A = 872 T = - 1.78 ns P = 57.6549 uw	Clk = 11.5 ns A = 518 T = - 0.32 ns P = 35.2419 uw	A = 40.56 % T = 10.95 % P = 38.87 %

Table 1.0: Comparison table.

Note:

- A = Area, T = Timing, P = Power.
- Design was mapped to lsi10K library, for slow corner.

Number Of Bits	Normal Method	Proposed Method	Improvement
8	Clk = 2 ns A = 7789 T = -0.45 ns DP = 3.0934 mw CLP = 8.6356 uw	Clk = 1.59 ns A = 3848 T = 0.0 ns DP = 2.0237 mw CLP = 3.6459 uw	A = 50.6 % T = 20.5 % DP = 34.6% CLP= 57.8 %
16	Clk = 4.97 ns A = 29018 T = 0 ns DP = 5.0361 mw CLP = 35.0561 uw	Clk = 3.97 ns A = 20491 T = 0.0 ns DP = 4.0874 mw CLP = 23.2080 uw	A = 29.39% T = 20.12% DP = 18.84% CLP= 33.80%
32	Clk = 9.65 ns A = 101175 T = 0 ns DP = 9.1776 mw CLP= 134.5426 uw	Clk = 6.74 ns A = 80972 T = 0.0 ns DP = 10.2906 mw CLP = 97.9276 uw	A = 19.97% T = 30.16% DP = -12.13% CLP= 27.21%

Table 1.1: Comparison table.

Note:

- A = Area, T = Timing, DP = Dynamic Power, CLP = Cell Leakage Power.
- Design was mapped to TSMC 0.13um library, for slow corner.

From tables 1.0 & 1.1, we can conclude that the proposed scheme is more efficient in terms of area, timing & power. The above results can be further improved by using the Look Up Table (LUT) approach to calculate the intermediate squaring values.

Ex: Table 2.0 compares the result for a 16-bit squarer:

Number Of Bits	Normal Method	Proposed Method	Improvement
16 WITHOUT LUT	Clk = 4.97 ns A = 29018 T = 0 ns DP = 5.0361 mw CLP = 35.0561 uw	Clk = 3.97 ns A = 20491 T = 0.0 ns DP = 4.0874 mw CLP = 23.2080 uw	A = 29.39% T = 20.12% DP = 18.84% CLP= 33.80%
16 WITH LUT	Clk = 4 ns A = 29054 T = - 0.91 ns DP = 6.1909 mw CLP = 35.1148 uw	Clk = 3.25 ns A = 16861 T = 0.0 ns DP = 2.8495 mw CLP = 13.5512 uw	A = 41.96% T = 33.8 % DP = 34.6% CLP= 53.97 %

Table 2.0: Comparison results for with & without LUT schemes.

Conclusion: This paper presented a novel approach to implement more area & power efficient Hybrid Squarer.

References:

1. Sri Bharathi Krishna Tirtha Maharaj, "Vedic Mathematics", Motilal Banarasi Das publications.
2. E. George Walters (2001), J. Schlessman, M. Schulte, "Combined Hybrid Squarers." Asilomar Conference on Signals, Systems and Computers. November.
3. www.vedicmaths.org
4. www.vedicganita.org