Efficient Large Integer Squarers on FPGA

Simin Xu[*], Suhaib A. Fahmy[‡], Ian V. McLoughlin[†]

[*]*Xilinx Asia Pacific, Singapore*
siminx@xilinx.com
[‡]*Nanyang Technological University, Singapore*
sfahmy@ntu.edu.sg
[†]*University of Science and Technology of China, Hefei, Anhui, China*
ivm@ustc.edu.cn

Abstract—**This paper presents an optimised high throughput architecture for integer squaring on FPGAs. The approach reduces the number of DSP blocks required compared to a standard multiplier. Previous work has proposed the tiling method for double precision squaring, using the least number of DSP blocks so far. However that approach incurs a large overhead in terms of look-up table (LUT) consumption and has a complex and irregular structure that is not suitable for higher word size. The architecture proposed in this paper can reduce DSP block usage by an equivalent amount to the tiling method while incurring a much lower LUT overhead: 21.8% fewer LUTs for a 53-bit squarer. The architecture is mapped to a Xilinx Virtex 6 FPGA and evaluated for a wide range of operand word sizes, demonstrating its scalability and efficiency.**

Keywords-**Field programmable gate arrays, Fixed-point arithmetic.**

I. INTRODUCTION

Specialized hardware for squaring is desirable for many high performance algorithms. A squarer can be significantly faster, consume less power, and be smaller than a general multiplier. Squaring is widely adopted in fixed point function evaluation [1], various floating point arithmetic [2], [3] and polynomial evaluation algorithms. Many techniques have been proposed to improve squaring in ASICs [4], [5].

Dedicated DSP Blocks which are able to implement operations such as multiplication or multiply-accumulate in a fast and efficient manner are provided in modern FPGA and they are used to build squarers in the literature [6]–[8]. To use them efficiently, designs must be customised to the datapath of the DSP block [9]. Since they are large and limited in number, reducing the number of DSP blocks required for elementary operations like squaring can facilitate the mapping of larger designs to an FPGA. However, simply trading DSPs with LUTs can lead to significant area cost.

In this paper, we investigate reducing squarer DSP blocks usage, by trading DSP blocks with limited amount of LUTs. We compare the proposed approach to a range of existing methods up to a wordlength of 64 bits.

II. BACKGROUND

The DSP48E1 block in the Virtex 6 and all 7-series FPGAs from Xilinx, contains a 25×18 bit signed multiplier with an optional pre-adder on input *A* and a programmable ALU after the multiplication, which can be used as a 48 bit post-adder [10]. The post-adder has a dedicated route through its *PCOUT* port to the *PCIN* port of the adjacent DSP block, with an optional 17-bit right shift. The input of the post-adder can come from input *C*, with an optional associated *CARRYIN*, but it is mutually exclusive with the *PCIN* signals. The DSP48E1 includes internal pipeline registers, allowing it to run at a maximum frequency of 450 MHz (for -1 speed grade Virtex 6 devices) [10].

The cascaded chain through dedicated routes allows users to build large wordlength multipliers. For a 35×35-bit multiplication, both operands would be split into two and the multiplication would be performed as four sub products in a chain comprising four DPSs. However, the number of DSP blocks required by the multiplier (for a given wordlength w where k splits are needed) grows quadratically as the wordlength increases, which could be derived from:

$$f(k) = \begin{cases} k^2 & \text{if } w - 17k > 1 \\ k^2 + 1 & \text{if } w - 17k \leq 1 \end{cases} \quad (1)$$

A dedicated squarer can be built in a similar way. For a 35×35-bit squarer, the equivalent equation:

$$x^2 = (a_0 + a_1 \cdot 2^{17})^2 = a_0^2 + 2 \cdot a_1 a_0 \cdot 2^{17} + a_1^2 \cdot 2^{34} \quad (2)$$

shows that only 3 DSPs are needed, a saving of one DSP. As the wordlength increases, more DSPs are saved compared with multiplication. Equations (3) and (4) show that only 6 and 10 DSPs are needed, compared with 9 and 16 for a multiplier, for three and four split operands respectively.

$$\begin{aligned} x \cdot x = {} & a_0^2 + 2a_1 a_0 \cdot 2^{17} + \left(a_1^2 + 2a_2 a_0\right) \cdot 2^{34} \\ & + 2a_2 a_1 \cdot 2^{51} + a_2^2 \cdot 2^{68} \end{aligned} \quad (3)$$

$$\begin{aligned} x \cdot x = {} & a_0^2 + 2a_1 a_0 \cdot 2^{17} + (a_1^2 + 2a_2 a_0) \cdot 2^{34} \\ & + (2a_2 a_1 + 2a_3 a_0) \cdot 2^{51} + (a_2^2 + 2a_3 a_1) \cdot 2^{68} \\ & + 2a_3 a_2 \cdot 2^{85} + a_3^2 \cdot 2^{102} \end{aligned} \quad (4)$$
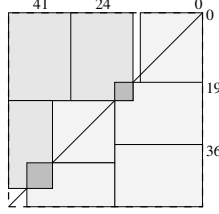
We can derive a similar equation to (1):

Figure 1. Tiling squarer method. Two LUT multipliers are shown in white in the middle top and bottom left, and two in dark grey [6].

$$f(k) = \begin{cases} (k^2 + k)/2 & \text{if } w - 17k > 1 \\ (k^2 + k)/2 + 1 & \text{if } w - 17k \leq 1 \end{cases} \quad (5)$$

A similar rationale was used in [6] (targeting the Xilinx Virtex 4) for the FloPoCo project [1]. We have mapped the auto-generated FloPoCo squarer to the Virtex 6 FPGA, however, we are able to produce expected performance with ISE v13.4 as the design is not optimally pipelined for the newer DSP architectures. Hence, we have also rewritten an optimized design, based on the same decomposition, that we refer to as the "cascaded method".

To further reduce DSP usage, the tiling method [6] was proposed, applied to 25×18 bit DSP blocks. The main approach is to achieve efficiency gains through maximizing the utilisation of the asymmetrically sized DSP inputs. This method can reduce the number of DSPs from 6 (cascaded method) to 5 for a 53-bit squarer, with the alignment of the DSPs shown in Figure 1. The main drawback is high LUT usage, as there are four small multiplications that must be mapped in LUTs. Although full implementation details are not provided, we have reproduced the design based on Figure 1, which applies to operand widths between 43 and 53 bits, to compare with the other methods evaluated.

III. PROPOSED DESIGN

In this section we propose an optimised squaring algorithm implementation on FPGA with lower DSP block usage. The algorithm is flexible with respect to operand size, though it targets higher wordlength, since (as will be shown later) it improves on existing methods when operand size exceeds 41 bit. The algorithm was inspired by the Karatsuba-Ofman algorithm [11], which is used to reduce the DSP cost for multiplication in [6] by trading multiplications with additions. However, the same reduction in [6] does not apply for squaring. According to equations in [12], there are no more advantages in using the reformation if both inputs are the same. We have thus modified the classic Karatsuba-Ofman algorithm and functions performed by the DSP blocks can be mapped to a small number of LUTs. As wordlength increases, more DSPs can be exchanged

[1]FloPoCo version 2.4.0, http://flopoco.gforge.inria.fr/

with LUTs, and the LUT count grows only gradually. We demonstrate the algorithm using two cases in the following.

We will take a 52×52 bit squarer to illustrate the algorithm, where equation (3) was the equivalent squaring equation using the cascaded method. We now define m and n as, $m = a_2 - a_1$ and $n = 2a_0 - a_1$ and their product as,

$$m \cdot n = 2a_2a_0 - 2a_1a_0 - a_2a_1 + a_1^2 \quad (6)$$

This can be rewritten to provide the term $a_1^2 + 2a_2a_0$ (which was in the middle of (3)) and now (3) becomes,

$$\begin{aligned} x \cdot x = a_0^2 + 2a_1a_0 \cdot 2^{17} + (mn + 2a_1a_0 + a_2a_1) \cdot 2^{34} \\ + 2a_2a_1 \cdot 2^{51} + a_2^2 \cdot 2^{68} \end{aligned} \quad (7)$$

In the formulation of (7), the DSPs are configured to compute the sub products a_0^2, $2a_1a_0$, a_2a_1, mn and a_2^2 (named DSP p_0 p_1 p_2 p_3 and p_4). Compared to the 6 DSP blocks needed in the cascaded method, only 5 are required, matching the DSP block savings of the tiling method.

The main benefit of this transformation is the reduced cost for the replacement logic: by carefully mapping the design, the LUT overhead is significantly smaller compared to the tiling method. Only one 17-bit LUT adder is required to obtain either m or n, since the other can be computed using the pre-adder in the DSP block p_3. Meanwhile, the rest has to be mapped to two cascaded chains, which are:

$$chain_0 = 2a_1a_0 + a_2a_1 \cdot 2^{17} + mn \cdot 2^{17} \quad (8)$$
$$chain_1 = 2a_1a_0 + 2a_2a_1 \cdot 2^{17} \quad (9)$$

$chain_0$ is the output directly from DSPs p_3, p_2 and p_1 with the cascaded addition chain and $chain_1$ can be derived from p_2 and p_1 in LUTs. The whole process requires one 34-bit adder to obtain $chain_2$ and one 51-bit adder to add the two chains. The final addition of the MSB can be implemented using the post adder in DSP p_4, by using its input bus C as well as the input CARRYIN. Figure 2 shows the overall implementation schematic of this architecture with pipeline registers. Due to the pipelined design, the architecture can be clocked at the maximum frequency of the DSP.

Note that when the wordlength extends to between 54 and 58, small changes to m and n are required to make their multiplication fit into one DSP block – these are not shown in detail here due to limited space.

When operand widths are above 59 bits, the inputs have to split into four parts. Take 64×64-bit as an example. Here, m and n are defined in a similar manner to in the case of three splits. Similarly, p and q can be defined as $p = 2a_3 - a_2$ and $q = a_1 - a_2$. Hence, the squaring equation (4) becomes,

$$\begin{aligned} x \cdot x = a_0^2 + 2a_1a_0 \cdot 2^{17} + (mn + a_1a_0 + 2a_2a_1) \cdot 2^{34} \\ + (2a_2a_1 + 2a_3a_0) \cdot 2^{51} + (pq + a_2a_1 + 2a_3a_2) \cdot 2^{68} \\ + 2a_3a_2 \cdot 2^{85} + a_3^2 \cdot 2^{102} \end{aligned} \quad (10)$$

Equation (10) reveals that only 8 DSPs are needed. We enumerate the DSPs from p_0 to p_7 for a_0^2, a_1a_0, a_2a_1, mn,
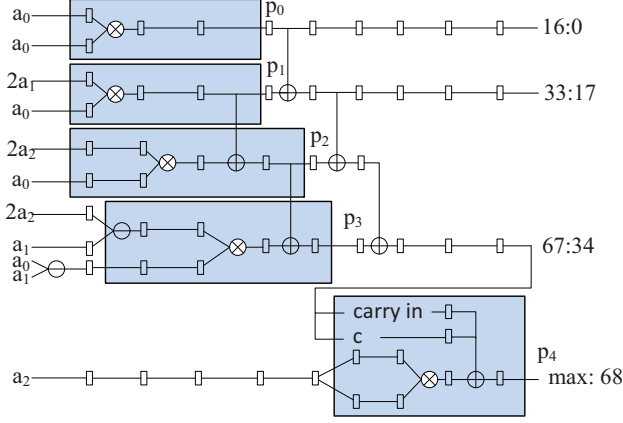
Figure 2. Pipeline schematic of squarer for three splits input.

$2a_3a_0$, pq, $2a_3a_2$ and a_3^2 respectively. This saves 2 DSPs over the cascaded method. By contrast, the tiling method doesn't propose a solution for operands this large.

Clearly, as operand size increases, the additional circuitry needed is more complex. To utilise the post-adder more efficiently, three chains of DSPs are defined, which are:

$$chain_0 = a_1a_0 + a_2a_1 \cdot 2^{17} \qquad (11)$$

$$chain_1 = chain0 + mn + 2a_3a_0 \cdot 2^{17} + pq \cdot 2^{34} \qquad (12)$$

$$chain_2 = a_2a_1 + 2a_3a_2 \cdot 2^{17} \qquad (13)$$

Equation (10) can be represented using these chains as,

$$x \cdot x = a_0^2 + 2chain_0 \cdot 2^{17} + chain_1 \cdot 2^{34}$$
$$+ chain_2 \cdot 2^{51} + chain_2 \cdot 2^{68} + a_3^2 \cdot 2^{102} \qquad (14)$$

We almost fully utilize the post-adder, either through a dedicated route or using the C input to map as much as possible into the DSPs, while the remainder are added with 3:1 compressors and normal adders in LUTs. Implementation details are not presented here for reasons of space.

The approach for even higher wordlengths is similar and the DSP count ($k > 2$) can be as low as:

$$f(k) = \begin{cases} (k^2 - k + 4)/2 & \text{if } w - 17k \geq 1 \\ (k^2 - k + 4)/2 + 1 & \text{if } w - 17k < 1 \end{cases} \qquad (15)$$

Note that the DSP block usage increases slower than for the standard cascaded method, resulting in higher gains with extremely large operands. With carefully aligned adders and formations of DSP chains, the extra LUT overhead can be constrained to be relatively small compared to the number of DSP blocks saved.

IV. RESULTS

The proposed algorithm has been synthesised, placed and routed using Xilinx ISE version 13.4, targeting the Xilinx Virtex 6 XC6VLX240T-1 FPGA. A design generator has

TABLE I
DSP BLOCK USAGE FOR ALL METHODS.

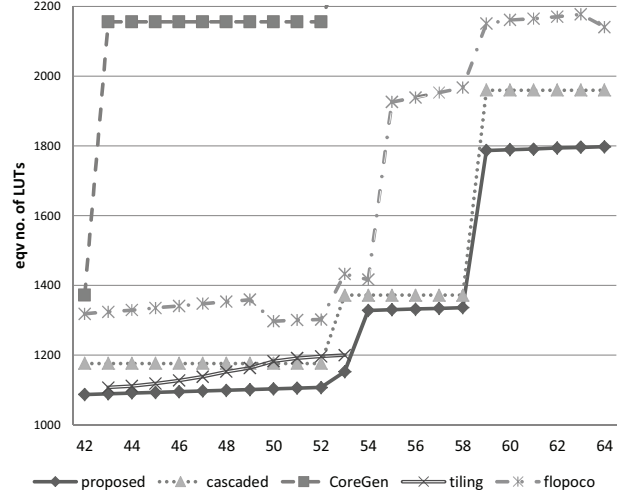| Size	CoreGen	FloPoCo	Cascaded	Tiling	Proposed
42–52	9	6	6	5	5
53	10	6	7	5	5
54	10	6	7	N/A	6
55–58	10	7	7	N/A	6
59	10	10	10	N/A	8
60–64	16	10	10	N/A	8

Figure 3. Equivalent LUT usage for all methods against wordlength.

been built to expand the algorithm across input operand wordlengths from 42 to 64 bits, inclusive. Both three splits (42 to 58 bits) and four splits (59 to 64 bits) are supported. The cascaded squarer designs as well as the general purpose multiplier designs provided by Xilinx CoreGen have been implemented for comparison across the same range of wordlengths. Similarly, squarers from FloPoCo have been compiled as another reference point. The tiling method has been built for wordlengths between 43 and 53 bits.

Table I shows the total number of DSPs used by each method. The proposed algorithm shows a gain across wordlengths and consumes up to 50% fewer DSP blocks compared to a general purpose multiplier. The tiling method has the same DSP count but only applies to sizes between 43 and 53 bits. Across most of the range, the proposed method uses 14.3 to 20% fewer DSP blocks than the cascaded method and FloPoCo squarers.

To quantify the total cost among different methods, we compute an equivalent number of LUTs to represent one DSP block. This is computed by taking the total number of LUTs in the device and dividing by the total number of DSP blocks. For the given FPGA, which has 150720 LUTs and 768 DSP blocks, this equals 196 (ranges from 160 to 240 for most general purpose Xilinx Virtex FPGAs).

Figure 3 shows the equivalent cost for all the implemented designs. Compared to the cascaded method and the design
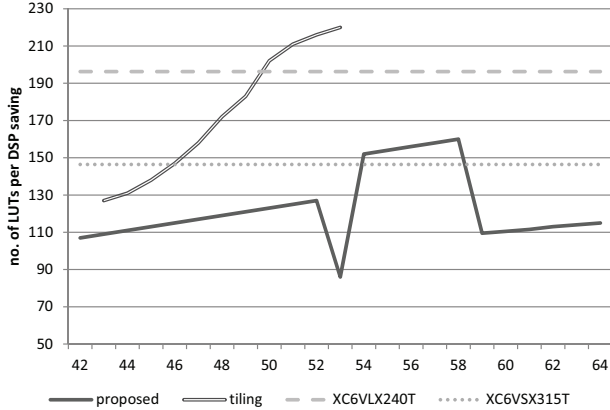
Figure 4. LUTs per DSP saved (from the cascaded method) ratio for tiling and proposed method against wordlength.

Table II
MAXIMUM FREQUENCY (MHZ) FOR ALL METHODS.

Size	CoreGen	FloPoCo	Cascaded	Tiling	Proposed
42–58	452.5	256.6	452.0	419.9	444.5
59–64	444.2	157.1	451.5	N/A	442.8

from FloPoCo, both the tiling method and the proposed method show an advantage in terms of total equivalent LUTs. For the 53-bit squarer, which is the optimal size for the tiling method, both methods save two DSP blocks over the cascaded method but the proposed method uses 21.8% fewer LUTs to achieve this. Between 43 and 52 bits, where both methods use 5 DSPs, the tiling method uses 127 to 216 LUTs to replace each DSP compared to 107 to 127 LUTs for the proposed method. This translates to up to a 41.2% improvement over the tiling method for the given operand word sizes. It is not possible to show the equivalent number of LUTs for the multiplier on the same axis, as for high wordlength the cost is as high as 3528 LUTs.

The proposed method is flexible in terms of operand size without significant efficiency penalties. Figure 4 shows the LUTs required per saved DSP compared to the cascaded method for the proposed method and the tiling method. Two horizontal lines show the LUTs:DSP block ratio for the target device and for the Xilinx XC6VSX315T DSP-rich FPGA. The tiling method reaches an architectural limit for replacing DSP with LUTs above a wordlength of 49 bits. In contrast, even for the DSP rich FPGAs, it is still worthwhile to use the proposed algorithm to reduce the DSP count.

Table II shows the maximum post place and route frequency of the evaluated methods. The proposed method, along with the cascaded method are able to sustain speed as operand wordlength increases.

V. CONCLUSION

This paper has proposed a new method for building efficient squarers on FPGAs , using up to 21.8% fewer LUTs

for a 53 bit squarer compared to the state-of-the-art tiling method and with less equivalent cost compared to other methods in the literature. The method is shown to be scalable across a wide range of operand wordlengths, maintaining its advantages over other methods, and supporting a sustained operating frequency.

Although the new approach is optimized for Xilinx Virtex 6 and newer devices, it is not restricted to architectures with asymmetric multipliers. It can be applied to older DSP blocks on Xilinx or Altera devices with minimum changes to the adder mapping. We aim to further research how this method can be applied to higher powers that would be useful for polynomial evaluation.

REFERENCES

[1] K. Chung and L.-S. Kim, "Area-efficient special function unit for mobile vertex processors," *Electronics Letters*, vol. 45, no. 16, pp. 826 –827, 30 2009.

[2] D. D. Donofrio and X. Li, "Enhanced floating-point unit for extended functions," U.S Patent 7 676 535, 2007.

[3] B. Pasca, "Correctly rounded floating-point division for DSP-enabled FPGAs," in *Field Programmable Logic and Applications*, Aug. 2012, pp. 249 –254.

[4] A. Strollo and D. De Caro, "Booth folding encoding for high performance squarer circuits," *IEEE Transactions on Circuits and Systems II*, vol. 50, no. 5, pp. 250 – 254, May 2003.

[5] K.-J. Cho and J.-G. Chung, "Parallel squarer design using pre-calculated sums of partial products," *Electronics Letters*, vol. 43, no. 25, pp. 1414 –1416, 6 2007.

[6] F. de Dinechin and B. Pasca, "Large multipliers with fewer DSP blocks," in *Field Programmable Logic and Applications*, Sep. 2009, pp. 250 –255.

[7] B. Lee and N. Burgess, "Improved small multiplier based multiplication, squaring and division," in *Field-programmable Custom Computing Machines*, Apr. 2003, pp. 91–97.

[8] S. Gao, N. Chabini, D. Al-Khalili, and P. Langlois, "FPGA-based efficient design approach for large-size two's complement squarers," in *Application-specific Systems, Architectures and Processors*, Jul. 2007, pp. 18 –23.

[9] B. Ronak and S. Fahmy, "Evaluating the efficiency of dsp block synthesis inference from flow graphs," in *Field Programmable Logic and Applications 2012*, Aug., pp. 727–730.

[10] "Virtex-6 user manual," Xilinx Inc.

[11] A. Karatsuba and Y. Ofman, "Multiplication of multi-digit numbers on automata," *Soviet Physics Doklady 7*, vol. 145, pp. pp. 595–596, 1963.

[12] P. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362 – 369, Mar. 2005.