This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2714259, IEEE Embedded Systems Letters

# Fast Energy Efficient Radix-16 Sequential Multiplier

Saba Amanollahi, Ghassem Jaberipur
Department of Computer Science and Engineering, Shahid Beheshti University, Iran
{s_amanollahi, jaberipur}@sbu.ac.ir

**ABSTRACT-** We propose a new sequential multiplier design that generates the radix-16 partial products (e.g., $P$) as two high ($H$) and low ($L$) components, such that $P = 4H + L$, $H, L \in \{0, 1, 2, 3\} \times X$, where $X$ denotes the multiplicand. The required hard $3X$ multiple is generated in a preliminary cycle to the advantage of reducing the cycle time of the main iteration. Two radix-16 carry-save adders are used to generate the radix-16 accumulated partial product. The synthesis results show improved latency, power dissipation, and energy consumption over the previous relevant designs at the cost of additional silicon area, while however, the energy-area product is also lowered.

**Keywords-** Radix-16 Sequential Multiplier, Radix-16 carry-save Adder, Power, Latency, Energy.

## I. INTRODUCTION

With the advent of internet of things, embedded low area/power digital processors play an essential role in the required system designs. An integral crucial part of the arithmetic unit of such processors is the frequently executed multiplier hardware. However, the sequential multiplier architectures can influentially contribute towards the desired low area/power properties.

The conventional design of an $n$-bit sequential binary multiplier [1] produces the product $X \times Y$ in $n$ cycles, during each the new partial product is added to the accumulated partial product (APP).

The longstanding practice of utilizing redundant-digit representation of APP contributes to additional speed for partial product reduction (PPR). It also reduces the power dissipations via avoiding ripple-carry addition. Carry-save and signed-digit number systems are commonly used for this purpose [1][2]. Also the corresponding high-radix alternatives (e.g., radix-16 multiplication) reduce the number of required cycles and better exploits the available cycle time (normally equal to the time required for an $n$-bit addition). For example, a comprehensive study on using several radix-2, -4, and -16 carry-save and signed-digit number systems to implement sequential multipliers is presented in [3], where the underlying general architecture follows that of [1]. Moreover, there is an instance of actual industrial multiplier realization [4] and a more recent design [5] that despite the required hard $\{3, 5, 7\} \times X$ multiples (even with Booth recoding) have opted for radix-16 partial product generation on the grounds of better power savings with respect to previous radix-4 practice with no hard multiples.

In the radix-16 realization of [3], as is illustrated in Fig. 1 (reproduced from [3]), the new partial products, among $[0, 15] \times X$, are generated as $8Y[3]X + 4Y[2]X + 2Y[1]X + Y[0]X$, where $Y[3:0]$ represents the current radix-16 digit of the multiplier, and $X$ is the multiplicand. The APP is also maintained in radix-16 carry-save format. The aforementioned four components of the new partial product along the sum (i.e., $S$) and carry (i.e., $C$) components of the APP are fed into a PPR architecture that is composed of four carry-save adders.

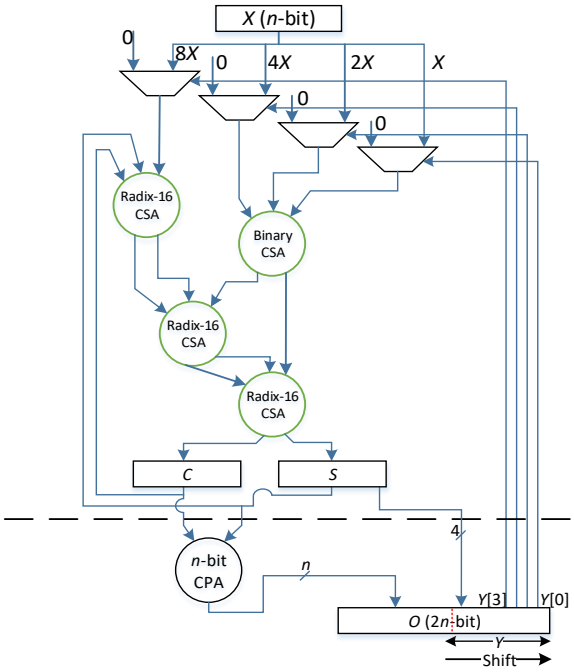



Fig. 1. The radix-16 sequential multiplier of [3].

An advantage of the latter scheme is that all the precomputed multiples (i.e., $\{2, 4, 8\} \times X$) are obtained via wired shifting. On the other hand, all the required hard multiples (i.e., $\{3, 5, 6, 7, 9, 11 \ldots 15\} \times X$) are computed carry-freely on the fly. In this paper, we provide a new PPR architecture that requires only two carry-save adders (CSA) in the main PPR body; hence speed up and lower power dissipation is expected. This is however, at the cost of a once used carry-propagate adder within a preprocessing cycle that produces the difficult $3X$ multiple. It is noteworthy that the conventional radix-8 Booth recoding multipliers also requires precomputation of the $3X$ multiple with carry propagation.

The rest of this work are organized as follows. Section II provides for some examples of low area/power processing environments that rely on sequential multipliers. Moreover, it refers to some selected relevant multiplier designs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2714259, IEEE Embedded Systems Letters

The proposed sequential multiplier architecture is discussed in Section III, while Section IV follows with evaluations of the previous relevant works and comparison of their figures of merit with those of the proposed one. The evaluations are both analytical and via synthesis with 45nm Nangate Synopsys design compiler. Finally we conclude in Section V.

## II. BACKGROUND

The history of sequential multiplier design dates back to the dawn of digital processors, when the high area consumption of parallel multipliers could not be afforded. However, while most of the arithmetic units of the today's general purpose digital processors are equipped with parallel multipliers, embedded systems mostly rely on sequential multiplier designs. For example, see Cortus APS23 and APS25 [6], ARM Cortex-M0, M0+, and M1 [7] digital processors; to name a few.

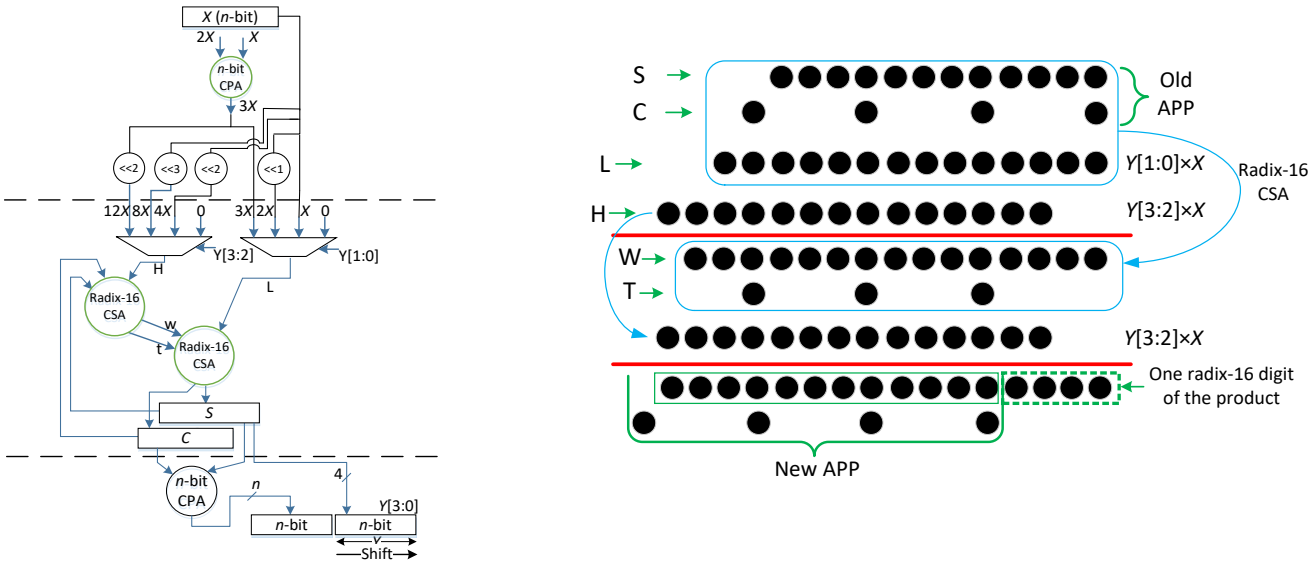
Note that the ARM Cortex-M series serve as the core of many system-on-chips such as the corresponding products of NXP [8], Actel [9], and ROHM [10].

Regarding the peculiarities of sequential multiplier design, several instances can be found in the textbooks for computer architecture and computer arithmetic [1][11]. Actual realizations mostly differ in the employed radix for partial product generation (PPG) and reduction. High radix PPG is commonly undertaken via selecting a pre-computed multiple of the multiplicand depending on the value of the next high radix digit of the multiplier [1][3].

Recoding of the multiplier's digits into redundant digits, in order to remove hard multiples (e.g., 3 in radix-4) is also common (e.g., Booth recoding [2]). However, negative multiples that are generated via such multiplier recoding, require carry-save adders that accommodate negative operands, whose considerable additional complexity is not desirable, let alone the overhead of the recoding logic itself. When it comes to PPR realization, the popular reduction cells include the carry-save adders [11][12], signed-digit adders [1], (4; 2) compressors [2], and the like.

## III. THE PROPOSED ARCHITECTURE

The internal structure of the proposed radix-16 sequential multiplier is depicted in Fig. 2, both in block diagram (a) and dot notation (b) representations, where each black circle represents a weighted bit [1], and component names (i.e., $S$, $C$, $W$, $T$, $H$, and $L$) establish the correspondence. The preprocessing cycle, produces the $3X$ multiple via an $n$-bit adder (not shown in the b part). The main iterative body is composed of two 4:1 multiplexers, whose inputs come from $X$, and $3X$ registers with the appropriate wiring shifts (neither shown in part b). Each partial product is produced as $P = 4H + L$, where $H, L \in \{0, 1, 2, 3\} \times X$. The bits of a multiplier digit such as $Y[3:0]$ are routed to the selector entries of the multiplexers. The two most significant bits $Y[3:2]$ select one of the multiples of the left one (i.e., the $H$ component), and $Y[1:0]$ is similarly used for the right multiplexer that selects the $L$ component. The two selected multiples along the carry (i.e., $C$) and sum (i.e., $S$) components of the APP constitute the four operands of the required 4-operand addition that is undertaken via two consecutive levels of radix-16 CSAs. The least significant radix-16 digit of such produced sum component (the dashed box in Fig. 2b) is directly sent to the least significant product register, while the rest of sum digits and the carry component comprise the new APP. After $n/4$ (i.e., the number of radix-16 digits of the multiplier $Y$) cycles of execution of the main body, a terminating cycle adds up the two components of the final APP to yield the most significant half of the product. Note that the least significant half has already been produced, one radix-16 digit per cycle. The final adder is an $n$-bit noneredundant adder.




a b

Fig. 2. Structure of the proposed sequential radix-16 multiplier in block diagram (a) and dot notation (b) representations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2714259, IEEE Embedded Systems Letters

## IV. EVALUATIONS AND COMPARISON

In this section, the figures of merit of the proposed radix-16 sequential multiplier are compared with those of a similar previous design [3], the latest proposed binary sequential multiplier [14] and the conventional parallel Wallace [15] tree multiplier. The latter is included as a reference just to exhibit the improvements due to sequential nature of the proposed design except for the speed. In the actual implementations, Kogge-Stone parallel prefix adder [13] is used for the required $n$-bit adders (i.e., the $3X$ generator, and the final adder for generating the most significant half of the product). For evaluation purposes, all the studied multipliers have been implemented by Verilog HDL (structural modeling), and synthesized by Synopsys Design Compiler under 45nm NanGate technology [16]. Consequently, the extracted measures from the reports of the synthesis tool are compiled in Table I, which provides delay, latency, power, energy, area, and energy-area product of these multipliers under different input operand widths. The ratio entries are assumed 1.00 in case of the corresponding least measures, and the rest of ratios are obtained by simple divisions. Also Fig. 3 provides five logarithmic curves for better illustration of the synthesis results.

A close look at the contents of Table I reveals that each cycle of the binary sequential design of [14] is fastest in cases of 32- and 64-bit multipliers. However, cycle time of the proposed 128-bit multiplier catches up with that of [14]. The reason is that for long word operands the cycle time is determined by the final adder, while delay of one PPR stage is considered for that of the shorter word operands. However, given the smaller number of cycles in the case of our high radix design, its corresponding latencies are 60%, 68%, and 73% less than that of [14] for 32-, 64-, and 128-bit cases, respectively.

On the other hand, the proposed designs show considerable improvement over our similar previous design. For example, the latencies of the three new designs are on the average 15% less than that of the previous one [3], as expected. Also, the latency ratios with respect to the parallel design are noteworthy (see also Fig. 3a). Regarding the evaluated power and energy figures, the lowest ones are due to the proposed radix-16 sequential multipliers. The energy ratios of the reference work [14] are almost 4 times more than the proposed designs. Similar ratios for the previous radix-16 design [3] is from 1.6 to 1.8 (see also Fig. 3c).

Among the studied designs, the binary sequential multiplier of [14] can be singled out as the least cost architecture (see the area ratios in Table I, and Fig. 3d). However, its area-energy product ratios are almost twice as those of the proposed architectures (see also Fig. 3e). On the other hand the area measures of the proposed designs are, on the average, about 7% less than those of our previous radix-16 sequential multiplier [3]. Finally, the very high ratios for the area and energy-area product of the parallel multiplier is notable, as of course expected.

## V. CONCLUSIONS

The $3X$ multiple is recognized as a hard multiple and thus undesirable in the practice of multiplier design, since it takes up one whole cycle to generate. The modified Booth multipliers remove such multiples for radix-4 multipliers. On the other hand, to reduce the total number of cycles, higher radix multiplication schemes are common, where higher radix Booth encoding cannot avoid the $3X$ multiples. However, we showed that allocation of a preliminary cycle for generation of $3X$ multiple can be advantageous.

TABLE I. FIGURES OF MERIT FOR THE PROPOSED AND THE RELEVANT REFERENCE WORKS

| Operand Width | Architecture | Cycles | Delay (ns) | ratio | Latency (ns) | ratio | Power (mW) | ratio | Energy (pJ) | ratio | Area ($\mu m^2$) | ratio | Energy ×Area | ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | Proposed | 10 | 0.71 | 1.29 | 7.1 | 4.77 | 1.62 | **1.00** | 11.50 | **1.00** | 2527 | 2.14 | 29066 | **1.00** |
| | [3] | 9 | 0.91 | 1.65 | 8.19 | 5.50 | 2.26 | 1.40 | 18.51 | 1.61 | 2606 | 2.21 | 48235 | 1.66 |
| | [14] | 32 | 0.55 | **1.00** | 17.6 | 11.81 | 2.72 | 1.68 | 47.87 | 4.16 | 1179 | **1.00** | 56441 | 1.94 |
| | Parallel | 1 | 1.49 | 2.71 | 1.49 | **1.00** | 10.49 | 6.48 | 15.63 | 1.36 | 16139 | 13.69 | 252254 | 8.68 |
| 64 | Proposed | 18 | 0.75 | 1.12 | 13.5 | 7.38 | 3.002 | **1.00** | 40.53 | **1.00** | 4829 | 2.03 | 195705 | **1.00** |
| | [3] | 17 | 0.94 | 1.40 | 15.98 | 8.73 | 4.42 | 1.47 | 70.63 | 1.74 | 5288 | 2.22 | 373500 | 1.91 |
| | [14] | 64 | 0.67 | **1.00** | 42.88 | 23.43 | 4.02 | 1.34 | 172.38 | 4.25 | 2380 | **1.00** | 410259 | 2.10 |
| | Parallel | 1 | 1.83 | 2.73 | 1.83 | **1.00** | 36.56 | 12.18 | 66.90 | 1.65 | 55715 | 23.41 | 3727601 | 19.05 |
| 128 | Proposed | 34 | 0.81 | **1.00** | 27.54 | 13.18 | 5.86 | **1.00** | 161.38 | **1.00** | 9653 | 1.97 | 1557844 | **1.00** |
| | [3] | 33 | 1 | 1.23 | 33 | 15.79 | 8.79 | 1.50 | 290.07 | 1.80 | 10562 | 2.16 | 3063719 | 1.97 |
| | [14] | 128 | 0.81 | **1.00** | 103.68 | 49.61 | 6.47 | 1.10 | 670.81 | 4.16 | 4894 | **1.00** | 3282942 | 2.11 |
| | Parallel | 1 | 2.09 | 2.58 | 2.09 | **1.00** | 96.61 | 16.49 | 201.91 | 1.25 | 224576 | 45.89 | 45345241 | 29.11 |

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2017.2714259, IEEE Embedded Systems Letters

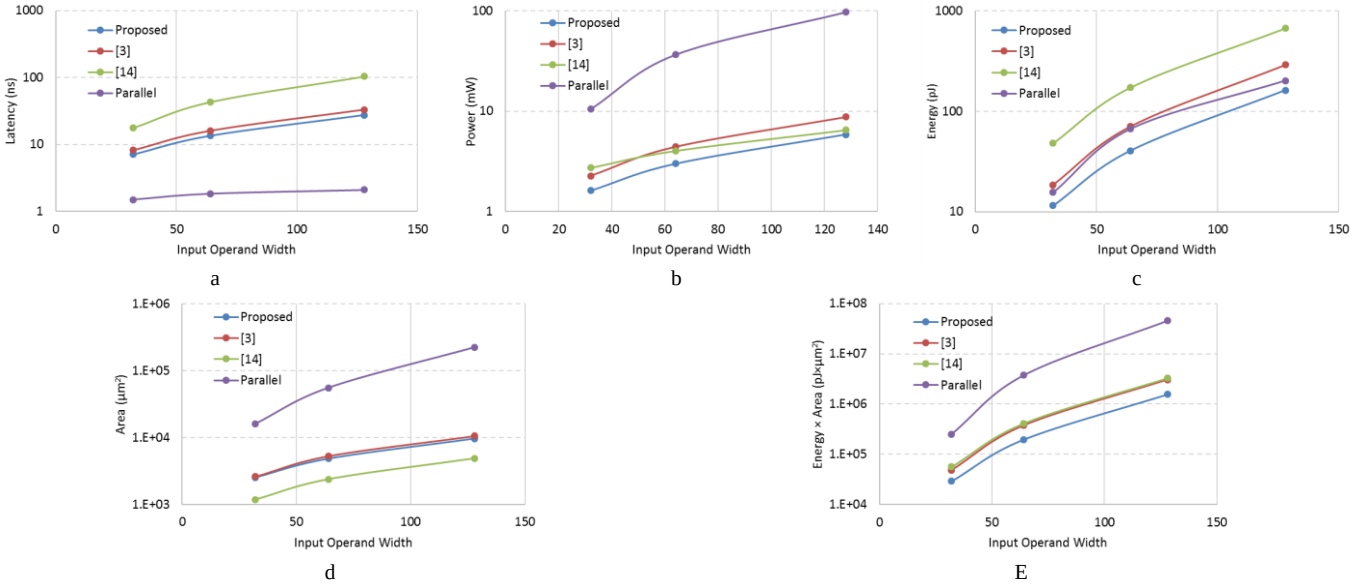



Fig 3. Logarithmic displays: a) Latency, b) Power, c) Energy, d)Area , and e) Energy-area product of the studied multipliers under different input operand width

The proposed radix-16 $n$-bit sequential $X \times Y$ multiplication scheme pre-computes $3X$ via an $n$-bit adder, and the other primary multiples (i.e., $\{2,4,8,12\} \times X$) are obtained via wired shifting, all in the preprocessing cycle. Each partial product is generated as two components $H = 4Y[3:2]X$ and $L = Y[1:0]X$ via two multiplexers. The generated partial product is added to the radix-16 carry save representation of the accumulated partial product via two levels of radix-16 CSAs that yields the new accumulated partial product and four bits of the product. Finally a terminating cycle converts the final carry-save accumulated partial product to the required most significant half of the binary product via an $n$-bit adder. This new architecture is shown, via synthesis of the proposed circuitry by 45nm Nangate Synopsys design compiler, to be considerably advantageous in latency, power dissipation, energy consumption and area-energy product over a previous sequential binary multiplier and a similar radix-16 design.

## REFERENCE


[1] Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, USA, 1999.

[2] M.D. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann, 2004.

[3] S. Amanollahi and G. Jaberipur, "Architecture-Level Design Space Exploration for Radix-16 Sequential Multipliers," CSI Journal of Computer Science and Engineering (JCSE), vol. 7, no.4 , pp. 24-30, 2016.

[4] R. Riedlinger et al., "A 32 nm, 3.1 billion transistor, 12 wide issue Itanium processor for mission-critical servers," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 177–193, Jan. 2012.

[5] E. Antelo, P. Montuschi, and A. Nannarelli, "Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction," IEEE Transactions on Circuits And Systems—I: Regular Papers, .vol 64, .no 2, pp. 409-418, 2017.

[6] Cortus.com, "Cortus Processors", 2016. [Online]. Available: http://www.cortus.com/index.php

[7] ARM.com, "ARM Cortex-M Series", 2016, [Online]. Available: http://www.arm.com/products/processors/cortex-m/index.php

[8] NXP.com, "NXP LPC 32-bit Microcontrollers", 2016. [Online]. Available: http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/lpc-cortex-m-mcus:LPC-ARM-CORTEX-M-MCUS.

[9] Actel.com, "Cortex-M1 with Actel", 2016. [Online]. Available: http://www.actel.com/eZone/ESC/p2.html.

[10] kionix.com, "ARM-Based Sensor Hub With Accelerometer", 2016. [Online]. Available: http://www.kionix.com/product/KX23H-1035.

[11] Mi Lu, Arithmetic and Logic in Computer Systems, Wiley, 2004.

[12] M. D. Ercegovac and T. Lang, "Fast Multiplication without Carry-Propagate Addition," IEEE Transactions on Computer, vol. 39, no. 11, 1385-1390, 1990.

[13] P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Transactions on Computers, vol.(22), no.(8), pp. 783-791, 1973.

[14] D. Baran, M. Aktan and V. G. Oklobdzija, "Multiplier Structures for Low Power Applications in Deep-CMOS," IEEE International Symposium on Circuits and Systems (ISCAS), 2011, pp. 1061-1064.

[15] C. S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp.14-17, 1964.

[16] Nangate.com, "NanGate - The Standard Cell Library Optimization Company", 2016. [On-line]. Available: http://www.nangate.com/.