HIGH-LEVEL OPTIMIZATION TECHNIQUES FOR LOW-POWER HIGH-PERFORMANCE BOOTH MULTIPLIERS

SHAIK ARSHIYA[1], DR.N.A.V.PRASAD [2]

[1] PG STUDENT, DEPARTMENT OF ECE, SRI VENKATESWARA INSTITUTE OF TECHNOLOGY, ANANTHAPURAMU.

[2] PROFESSOR, DEPARTMENT OF ECE, SRI VENKATESWARA INSTITUTE OF TECHNOLOGY, ANANTHAPURAMU.

Abstract: Multipliers are essential components of digital hardware, ranging from deeply embedded system on-chip (SoC) cores to GPU-based accelerators. As they are often critical for system performance, a great emphasis was placed on their performance improvement in the past few decades. The Radix-4 modified Booth encoding (MBE) scheme is often preferred in high-performance multipliers due to its minimized delay and silicon area. Booth encoding reduces the number of partial products required to be added by approximately twofold compared to non-Booth versions. Digital media and signal processing applications are usually dominated by integer addition and multiplication. Prior approaches have proposed several solutions supported the radix-4 Booth recoding. This technique makes it possible to diminish the peak of a multiplier by half, this being the foremost widespread option when designing multipliers, as only easy multiples are required. The proposed algorithm uses reconfigurable nature of VLSI systems. The proposed structure utilizes low power and that would be compared in terms of power, area utilization, logic utilization etc. In the proposed system, design of low power Radix-4 booth multiplier is being implemented using adjustable path selection routine that enable the Multiplier to save the carry path adjustable and finally combines both the result in the sum part.

1. INTRODUCTION

Multipliers are essential components of digital hardware, ranging from deeply embedded system-on-chip (SoC) cores to GPU-based accelerators. As they are often critical for system performance, a great emphasis was placed on their performance improvement in the past few decades [1]–[7]. While performance remains important, the high demand for battery-powered ubiquitous systems has promoted low-power operation to a primary design goal [8]. However, the majority of proposed high-performance multipliers suffers from increased capacitive loads and spurious activities due to their complex combinatorial modules and unbalanced reconvergent paths which could turn the multiplier to be the dominant source of power dissipation.

The Radix-4 modified Booth encoding (MBE) scheme is often preferred in high-performance multipliers due to its minimized delay and silicon area. Booth encoding reduces the number of partial products required to be added by approximately twofold compared to non-Booth versions. Moreover MBE is incorporated with various adder-tree-reduction schemes such as Wallace, optimized Wallace-tree (OWT), Dadda, Braun’s and three-dimensional minimization (TDM) to speed up the partial product addition. OWT scheme along with carry-save prop- agation is known for logarithmic delay reduction of the adder-tree which is composed of either full-adders or 4-to-2 compressors [18]–[20]. The latter is preferred for a regular adder-tree implementation.

Despite faster operation, the fitness of MBE for energy efficiency has been questioned due to its complex encoding–decoding circuity and higher spurious activities. This fact is especially prominent when the input operands are in 2's complement notation and have a smaller dynamic range. Therefore, alternative multiplier schemes such as Baugh-Wooley, sign magnitude (SM) , and gray coding (GC) have been proposed. The Baugh-Wooley implementation utilizes a 2-input AND array for partial product generation (PPG), which is simpler in logic and was shown to be ~25% more power efficient at a slightly higher delay when compared to Booth version. SM and GC, on the other hand, leverage the number representation to lower the signal transitions at the expense of a format conversion logic at both ends of the multiplier. SM implementations have reported up to 90% and 50% reduction in switching activity whereas GC reports 45% of power reduction compared to MBE. However, the applications where the input operands rapidly change across the entire word length scarcely benefit from these techniques. Besides, when the timing constraints are stringent, the conversion circuits in the critical path make these implementations slower and even more power-hungry due to the gate upsizing. The Booth multiplier has also been subjected to structural and gate-level optimizations in literature.

A more regular partial product array was proposed to minimize the extra adder rows for carry summation. The approach in has improved the multiplier performance by 25% when compared with the conventional implementations. Kang and Gaudiot presents a fast 2's complement generation circuit to reorganize the partial product array by removing the subsequent carry-in terms. The work in proposes a less hardware-intensive mechanism to achieve the same goal. These approaches have achieved up to 5%–9.1% improvements in performance while reporting 15%–33% of power savings for an 8-bit version, respectively. As alternatives to OWT, leap-frog (LFR, and left-to-right structures were proposed to alleviate the sum-carry imbalance. Despite their feasible layouts, the incurred area and delay overhead is not negligible. Alternatively, the optimized circuits presented demonstrate more balanced data paths and an efficient partial-product array structure that out- perform other higher level implementations. Row and column bypassing, dynamic operand interchanges were also considered to exploit the multiplier input asymmetry for low power. These techniques are questionable in general cases as the extra circuit overhead is a heavy burden. More recent approaches exploit the accuracy and the number representation for energy efficiency. Among them, only can be found relevant to the scope of this work, and it employs the same circuits presented in.

However, the fact remains that the area and speed are two conflicting performance constraints. Hence, innovating increased speed always results in larger area. The proposed architecture enhances the speed performance of the widely acknowledged Wallace tree multiplier when implemented on a FPGA. The structural optimization is performed on the conventional Wallace multiplier, in such a way that the latency of the total circuit reduces considerably. A truncated multiplier with constant correction has the maximum error if the partial products in the n-k least significant columns are all ones or all zeros. A variable correction truncated multiplier has been proposed. This method changes the correction term based on column n-k-1. If all partial products in column n-k-1 are one, then the correction term is increased. Similarly, if all partial products in this column are zero, the correction term is decreased. In a simplified 22 multiplier block is proposed for building larger multiplier arrays. In the design of a fast multiplier, compressors have been widely used to speed up the partial product reduction tree and decrease power dissipation. Kelly et al. and Ma et al. have also considered compression for approximate multiplication. An approximate signed multiplier has been proposed for use in arithmetic data value speculation (AVDS); multiplication is performed using the Baugh Wooley algorithm. However no new design is proposed for the compressors for the inexact computation.

2.LITERATURE SURVEY

A signed binary multiplication technique by A. D. Booth

The advances of digital arithmetic techniques permit computer designers to implement high speed application specific chips. The currently produced digital circuits have demonstrated high performance in terms of several criteria, such as, high clock rate, short input/output delay, small silicon area, and low power dissipation. In this paper, we implement several sinusoidal generation methods to optimize their performance and output using advanced digital arithmetic techniques. In this paper, the implementations of advanced digital oscillator structures with and without pipelining are proposed. The synthesis results of the implementation with pipelining have proven that it is superior to other sinusoidal generation methods in terms of the maximum frequency and signal resolution. Hence, this method is used in the design of the proposed digital oscillator chip.

A two’s complement parallel array multiplication algorithm by C. R. Baugh and B. A. Wooley

An algorithm for high-speed, two's complement, m-bit by n-bit parallel array multiplication is described. The two's complement multiplication is converted to an equivalent parallel array addition problem in which each partial product bit is the AND of a multiplier bit and a multiplicand bit, and the signs of all the partial product bits are positive.

High-performance low-power left-to-right array multiplier design by Z. Huang and M. D. Ercegovac

We present a high-performance low-power design of linear array multipliers based on a combination of the following techniques: signal flow optimization in [3:2] adder array for partial product reduction, left-to-right leapfrog (LRLF) signal flow, and splitting of the reduction array into upper/lower parts. The resulting upper/lower LRLF (ULLRLF) multiplier is compared with tree multipliers. From automatic layout experiments, we find that ULLRLF multipliers have similar power, delay, and area as tree multipliers for n/spl les/32. With more regularity and inherently shorter interconnects, the ULLRLF structure presents a competitive alternative to tree structures in the design of fast low-power multipliers implemented in deep submicron VLSI technology.

3. EXISTING SYSTEM

The algorithm of WALLACE multiplier is based on the below matrix form. The partial product matrix is formed in the first stage by AND stages

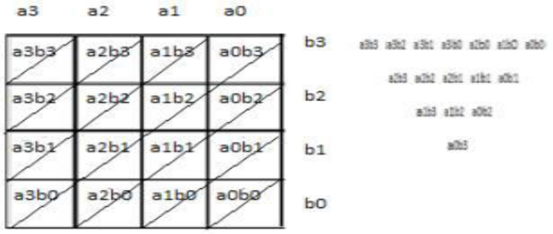

Fig.1-4x4 WALLACE Algorithm

Steps involved in WALLACE TREE multipliers Algorithm

Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding N results. Depending on position of the multiplied bits, the wires carry different weights.

Reduce the number of partial products to two layers of full adders. Group the wires in two numbers, and add them with a conventional adder.. Product terms generated by a collection of AND gates.

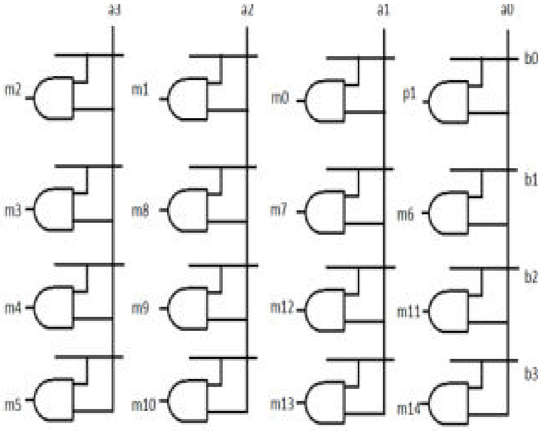

Fig 2 Product terms generated by a collection of AND gates.

Ripple Carry Adder is the method used to add more number of additions to be performed with the carry in sand carry outs that is to be chained. Thus multiple adders are used in ripple carry adder. It is possible to create a logical circuit using several full adders to add multiple-bit numbers. Each full adder inputs a Cin, which is the Cout of the previous adder. This kind of adder is a ripple carry adder, since each carry bit "ripples" to the next full adder. The proposed architecture of WALLACE multiplier algorithm using RCA is shown in Figs.9 to 11 Take any 3 values with the same weights and gives them as input into a full adder. The result will be an output wire of the same weight. Partial product obtained after multiplication is taken at the first stage. The data's are taken with 3 wires and added using adders and the carry of each stage is added with next two data‟s in the same stage. Partial products reduced to two layers of full adders with same procedure. At the final stage, same method of ripple carry adder method is performed and thus product terms p1 to p8 is obtained.

4. PROPOSED SYSTEM

Radix-4 Booth algorithm is the multiplier. This computes $x \times y$ where x and y are the n bit two's complement numbers (the multiplicand and multiplier respectively); producing a $2n$ two's complement value in the product p. The multiplication algorithm considers multiple digits of Y at a time and is computed in N partitions where

$$N = \left\lfloor \frac{n+2}{2} \right\rfloor . \qquad 1$$

An equation describing the computation is given by

$$p = (Y_1 + Y_0)x + \sum_{i=1}^{N} 2^{2i-1}(Y_{2i+1} + Y_{2i} - 2Y_{2i-1})x . \qquad 2$$

Y indicates the length-N digit vector of the multiplier y. The radix-4 Booth calculation considers three digits of the multiplier Y at a time to create an encoding e given by

$$e_i = Y_{2i+1} + Y_{2i} - 2Y_{2i-1} \qquad 3$$

Y_{i+2}	Y_{i+1}	Y_i	e_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	$\bar{2}$
1	0	1	$\bar{1}$
1	1	0	$\bar{1}$
1	1	1	0

TABLE I BOOTH ENCODING

where I indicates the I th digit. As outlined in Table I, separated from Yi+2Yi+1Yi = 000 and Yi+2Yi+1Yi = 111 which results in a 0, the multiplicand is scaled by one or the other 1, 2, −2, or −1 contingent upon the encoding.

This encoding ei is utilized to work out an incomplete item Partial Product i by working out

$$PartialProduct_i = e_i x = (Y_{2i+1} + Y_{2i} - 2Y_{2i-1})x. \qquad 4.$$

This Partial Product is adjusted utilizing a left shift (22i−1) and the summation is performed to compute the end-product p. Since the Y−1 digit is nonexistent, the 0th halfway item PartialProduct0 = (Y1 +Y0)x. A sequential (consecutive) rendition of the augmentation is performed by figuring every incomplete item in N cycles

$$p[0] = 2^{n-2}(Y_1 + Y_0)x$$
$$p[j+1] = 2^{-2}(p[j] + 2^n(Y_{2j+1} + Y_{2j} - 2Y_{2j-1})x),$$
$$j = 1, \ldots, N-1 \qquad 5$$

Two improvements are performed to take into account better equipment usage. In the first place, the item p is doled out the multiplier y (p = y), this eliminates the need to store y in a different register and uses the n LSBs of the p register. Subsequently, as the item p is moved right (p = sra(p, 2)), the following encoding ei can be determined from the three LSBs of p. The subsequent improvement eliminates the realignment left shift of the fractional item (2n) by aggregating the Partial Product to the n generally huge pieces of the item p (P[2_ B−1 : B]+ = Partial Product).

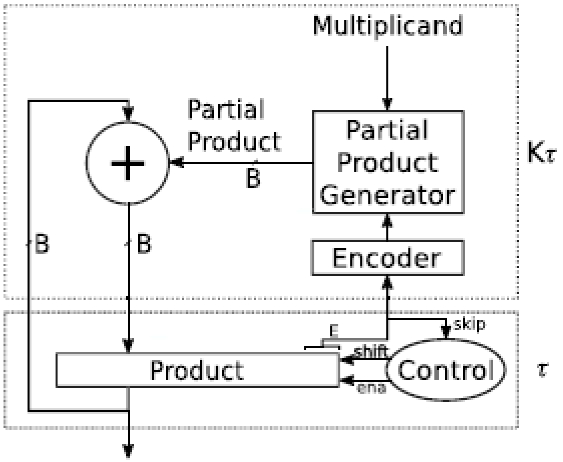

Fig. 3. n bit TSM. This contains an added control circuit for skipping and operating with two different delay paths

As observed in earlier research, a proper choice of inter-mediate signals in the interface between Booth encoding and decoding offers opportunities for logic optimization. Fig. 2(a)–(d) illustrates the traditional implementations of MBE circuits found in the literature. Note that only the full-swing circuit topologies were considered in this study. Fig. 2(a) (BED13) depicts a hybrid implementation of encoder–decoder circuits which require 36 and 10 transistors [46], respectively. This non-CMOS implementation reports the least number of transistors for the decoder block among the presented. However, there are a few issues that emanate from this implementation. First, the unbuffered selector circuit which is denoted by SEL (composed of four pass transistors), forms cascaded resistive paths from decoder inputs to the outputs as highlighted in Fig. 3(a). This results in an asymmetry in the driving loads to the SEL blocks for different input combinations and therefore different arrival times. Secondly, the routing congestion across the decoding blocks in Fig. 2(a) is relatively higher and increases the interconnects parasitic across the PPG.

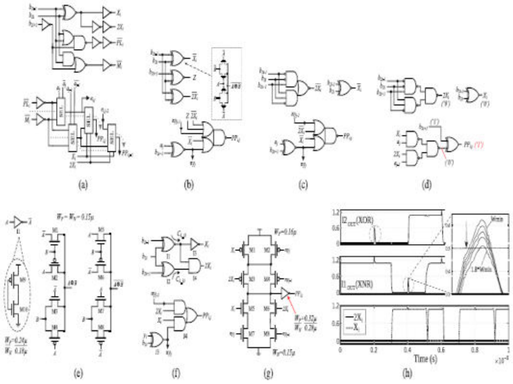

Fig.4. Various Booth encoder–decoder implementations. (a) BED13 [46]. (b) BED20 [27]. (c) BED22 [7], [16], [41]. (d) Erroneous Booth circuits in [17]. (e) 6T-XOR/XNR circuits of this work (*WM*1−*M*8 = 0.15*μ*). (f) Proposed encoder–decoder circuits (BED18). (g) AO22 (*J*3) of the decoder (*WM*1−*M*4 = 0.16 *μ*, *WM*5−*M*8 = 0.15 *μ*).

The circuits shown in Fig. 2(b) (BED20) [27] uses transmission gate pairs for encoders leading to a faster operation in PPG. However the unbuffered encoder outputs become transparent to the hazards induced by the circuit itself. The additional wiring and higher capacitive loading at the decoder leads to a higher power consumption in PPG at the same time. The arrangement in Fig. 2(c) (BED22) is the most optimized version in terms of transistor count and signal synchronization. The XORs which produce ny j −1–ny j are shared among the decoders and the AOI22 cell provides balanced loads to the encoder signals. Therefore, it was also preferred for the truncated multiplication in [41]. The unique Booth circuits presented are not considered for the evaluation due to functional failures when all the encoder inputs (b2i −1–b2i +1) are at logic "1".

The proposed MBE circuits in this work are shown in Fig. 2(e)–(g). The essential leaf cell of the proposed circuitry is depicted in Fig. 2(e). This XOR/XNR arrangement results in fewer number of gate capacitances when compared to any other full-swing implementations. Despite this merit, it suffers from the delay asymmetry between the signal paths. If, for example, in the circuit of Fig. 2(e), when both inputs change from 0 → 1, M1 of the XOR drives the output for a short period of time due to the inertial and propagation delays of the inverter and as a result, a glitch appears at the XOR output. The inversely proportional relationship between the inertial and propagation delays limits the liberty of device sizing. As such, the direct interfacing of these XOR/XNR outputs to high fan-out nets could only worsen the spurious activities in PPG.

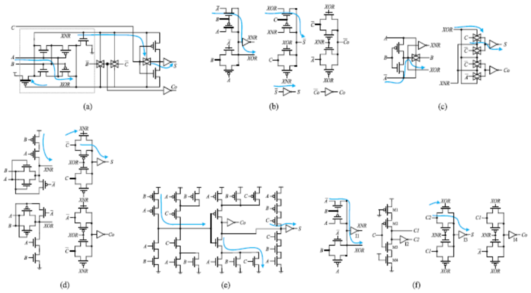

Fig. 5. Various low-power, full-swing full adders. (a) RFL22 [50]. (b) TFA22

[51]. (c) BFA22 [52]. (d) HFA26 [49]. (e) CMOS28 [44]. (f) Proposed (PBFA26).

Full adders are the basic building blocks of the multiplier adder-tree. The most prevalent, rail-to-rail static full adder implementations are shown in Fig. 3(a)–(e). For a fair comparison, the buffered versions of the original implementation are considered. The blue arrow line indicates the critical path of each full adder. Fig. 3(a)–(c) requires a minimum of 22 transistors (including the inverters for the input signals that have not been drawn). The numbers for Fig. 3(d)–(f) are respectively. Fig. 3(a) utilizes a simultaneous, six transistors XOR-XNR circuit which is delimited by a dashed line in 3(a). Despite its compactness, the regenerative feedback paths introduced by this circuit results in slower transitions. In addition, the cascaded transmission gates worsen the sum-carry generation (SCG), thereby making the outputs more susceptible to glitches. In Fig. 3(b) (TFA22), the Sum output (S) is produced faster when input C = “1,” compared to other input combinations. Besides, the late arrival of XOR-XNR signals to the SCG could introduce glitches at output S. By contrast, the control signals to the transmission gates in Fig. 3(c) (BFA22) are reasonably synchronized except its input signals, i.e. early arrival of input C when XOR0→1 is a potential scenario for glitch generation at output S. Similar to RFL22 and TFA22, HFA26 in Fig. 3(d) [49] suffers from asymmetric path delays despite its faster operation. Fig. 3(e) (CMOS28) [44] represents the traditional CMOS full adder which is reasonably immune to glitches. The proposed full adder (PBFA26) is illustrated in Fig. 3(f). This arrangement differs from the others in two aspects. First, the internal signals are capacitively terminated at the SCG stage and the gate capacitances of the transmission gate pairs in SCG absorb possible glitches similar to Booth circuits. Secondly, the synchronization of all signals to SCG is achieved by incorporating a low-overhead intracell delay element [44] depicted by M1–M4 of Fig. 3(f). M1 and M4 provide the required delay to the input C through their drain–source parasitic Cd /Cs which are smaller than C g . Since C g of both M1 and M4 are not switched, its parasitic contribution to the full adder dynamic power is significantly lower when compared to an inverter-based delay elements. Hence, the arrival of C can be independently controlled without a significant overhead.

5. SIMULATION RESULT OF MULTIPLIER:

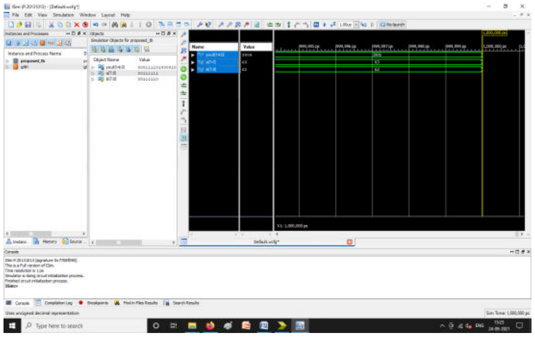

Fig 6 Simulation Result of Multiplier

Here we can give the inputs as A=63, B=62, then the final output is 3906.

SYSTEM	POWER(w)	Delay(ns)
Existing(Wallace tree multiplier)	0.299	32.681
Proposed method	0.215	24.305

Table 2 Comparison between power and delay

6.CONCLUSION

This article has proposed and investigated glitch-optimized circuit blocks for high-performance Booth multipliers aiming to reduce the dynamic power dissipation caused by the parasitics and spurious activities. The proposed strategy incorporates circuit-level techniques with a PASR to achieve this goal. Therefore, the proposed approach is an excellent choice for high-performance, energy-constrained multiplication at the expense of a slightly higher delay. Two versions of the multiplier structures (Prop-W, Prop-LFR) comprising these circuit blocks, have been compared to highly optimized array and tree versions of the multipliers comprised of the state-of-the-art building blocks in literature. From the postlayout simulations, it was concluded that the proposed versions are on average 10%–30% more power efficient compared to the baselines.

7. FUTURE SCOPE

This multipliers plays a very important role in our day to day life. In future the multipliers are going to play a major role. The speed of the multipliers are increased by using carry save adders, carry look ahead adder, and so on. Rounding patterns will be optimized based on required accuracy and different compression techniques. The area and delay can be reduced in future by using advanced technology.

BIBLIOGRAPHY

[1] A. D. Booth, “A signed binary multiplication technique,” Quart. J. Mech. Appl. Math., vol. 4, no. 2, pp. 236–240, 1951.

[2] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs. New York, NY, USA: Oxford Univ. Press, 2000.

[3] M. D. Ercegovac and T. Lang, Digital Arithmetic (Morgan Kaufmann Series in Computer Architecture and Design). San Mateo, CA, USA: Morgan Kaufmann, 2004.

[4] B. Dinesh, V. Venkateshwaran, P. Kavinmalar, and M. Kathirvelu, “Comparison of regular and tree based multiplier architectures with modified booth encoding for 4 bits on layout level using 45 nm technology,” in Proc. Int. Conf. Green Comput. Commun. Elect. Eng., Mar. 2014, pp. 1–6.

[5] O. L. MacSorley, “High-speed arithmetic in binary computers,” Proc. IRE, vol. PROC-49, no. 1, pp. 67–91, Jan. 1961.

[6] L. P. Rubinfield, “A proof of the modified Booth’s algorithm for multiplication,” IEEE Trans. Comput., vol. C-24, no. 10, pp. 1014–1015, Oct. 1975, doi: 10.1109/T-C.1975.224114.

[7] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, “Stripes: Bit-serial deep neural network computing,” in Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture, Oct. 2016, pp. 1–12.

[8] G. C. T. Chow, W. Luk, and P. H. W. Leong, “A mixed precision methodology for mathematical optimisation,” in Proc. IEEE 20th Int. Symp. Field-Program. Custom Comput. Mach., Apr./May 2012, pp. 33–36.

[9] G. C. T. Chow, A. H. T. Tse, Q. Jin, W. Luk, P. H. Leong, and D. B. Thomas, “A mixed precision Monte Carlo methodology for reconfigurable accelerator systems,” in Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays, 2012, pp. 57–66.

[10] S. J. Schmidt and D. Boland, “Dynamic bitwidth assignment for efficient dot products,” in Proc. Int. Conf. Field Program. Log. Appl., Sep. 2017, pp. 1–8.

[11] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, “Hardware for machine learning: Challenges and opportunities,” CoRR, vol. abs/1612.07625, Dec. 2016. [Online].

Available:
https://arxiv.org/abs/1612.07625

[12] B. Rashidi, S. M. Sayedi, and R. R. Farashahi, "Design of a low-power and low-cost Booth-shift/add multiplexer-based multiplier," in Proc. Iranian Conf. Elect. Eng. (ICEE), May 2014, pp. 14–19.

[13] P. Devi, G. P. Singh, and B. Singh, "Low power optimized array multiplier with reduced area," in High Performance Architecture and Grid Computing, A. Mantri, S. Nandi, G. Kumar, and S. Kumar, Eds. Berlin, Germany: Springer, 2011, pp. 224–232.