VLSI Design of High Speed and Area Efficient Radix-8 Serial Parallel Multiplier

Chandrakala[1], G.Nagendra[2]
[1]*Vidya Jyothi Institute of Technology, Hyderabad, India*
[2]*Vidya Jyothi Institute of Technology, Hyderabad, India*
chandrakalamedha@gmail.com
nag20209@gmail.com

Abstract— Multiplier is one of the hardware components that typically consumes a significant chip region which requires to be reduced and can be helpful to a wide range of applications whereby multiplication components are an essential device, such as digital signal processing (DSP) systems or computational methods. The proposed method it accelerates applicants like digital filter, machine learning and neural networks by developing a two-speed radix-8 serial parallel multiplier. Proposed multiplier is a variation of the adapted radix-8 Booth multiplier serial – parallel (SP), which only incorporates nonzero Booth encoding along with skips over zero functions, allowing latency based on the multiplier size. This study focused on DSP applications where in multiplier remains being used substantially and suggests strategy that helps to minimize the hardware as well as latency contributing to improved device output and thereby helps to improve the running frequency by a substantial amount. An 8-bit multiplier was designed using a booth multiplication of radix-4, which decreases number of partial products.

Keywords— Radix-8 Booth's multiplier; Radix-4, Booth's multiplier; Serial Parallel Multiplier; Partial products.

1. Introduction

Multipliers have been presented using add and shift procedure to execute multiplication operation of a arithmetic circuits. Simply put, the overall system production is defined by the multiplier's performance because the multiplier is regarded to become the slightly slower component in the entire system, and it consumes area. Digital signal processing also involves multiplication. Parallel multipliers offer a high-speed multiplication approach which need a wide area for VLSI implements.

Multiplication is clearly the most influential primary method for DSP and ML applicants, trying to dictate the parallel implementation area, delay, and overall production. There has been extensive work on optimizing multiplication circuits [1]-[2]. Furthermore, the combination of wallace or Dadda tree which is modified version of larger radixes of booth algorithm was widely considered as high performing development of overall issues [2]-[4]. Generally, there are three ways that were done in digital circuits multiplications which are 1) parallel-parallel; 2) serial-parallel; 3) serial-serial. Utilizing updated Booth algorithm [5]-[6], thus analyze a two-speed SP multiplier (TSM) which contingently attaches the non-zero encoding multiplying components and skips across these parts marked zero.

Reduction accuracy represents are sometimes utilized in DSP and ML architectures to boost the efficiency of a specification, aiming for the shortest bit width possible toward obtain desirable computational accuracy [7]. Accuracy was generally provided by the period of design, and therefore any changes to the requirements require that further modification involve a redesign of execution. In these situations, in which a shorter bit width would've been necessary, since unnecessary computation is undertaken, the design runs at a lower efficiency. To minimize this, mixed-precision algorithms aim to utilize any component of the time with a smaller bit width, .including a wide bit width if required. Normally these are executed with two data paths run at various accuracy.

This work presented a complex control system which can totally delete sections of the computing through initialization. It is achieved by utilizing a updated serial booth multiplier, that skips independently of position over encoded all-zero or all-one calculation. The multiplier holds all the parts from both operands through parallel and is structured to become an conveniently integrated fundamental unit into existing DSPs, CPUs, and GPUs. The multiplier makes major increases in numerical efficiency with some collections of inputs. A key feature of a multiplier is that these dimensionality within the collection of inputs and inner binary representation contribute to change in output.

In this study we are developing a revised Booth algorithm radix-8 which overcomes all these limitations of the algorithm Radix-4.

2. Multiplication

Multiplier is indeed a crucial fundamental which always determines the broad DSP implementations operate. Size et al. indicated most hardware improvements designed for ML aim to reduce multiplying costs and accumulating procedures. Consequently, careful design of a computing unit, with a emphasis of multiply, results in the greatest effect on efficiency. This segment introduces an algorithm for multiplying unsigned entities accompanied by their extended to signed entities [2],[3].

$n = 4$	$x = 13$ (X = 1101) $y = 9$ (X = 1001)
p[0]	0000
$2^4 x Y_0$	1101
p[1]	11101
$2^4 x Y_1$	0000
p[2]	111101
$2^4 x Y_2$	0000
p[3]	1111101
$-2^4 x Y_3$	0011
p[4]	01110101 = 117

Fig. 1. Multiplication p= x= y in which x is the multiplicand, y are the multiplying agent, and X and Y are the n =4-digit radius 2 conventional numbering vectors respectively.

There are two factors which are multiplicand and multiplier considered as X and Y which is represents as conventional radix-r numbering logic by n digit vector of X ,Y .The procedure of multiplying generates p = x x y when p is defined by vector P with 2n integer. Then multiplication is defined as

$$P= x \sum_{i=0}^{n-1} y_i r^i = \sum_{i=0}^{n-1} r^i \text{ x } y_i \qquad (1)$$

Equation (1) may be performed by first calculating the terms n x r^i y_i preceded by summation. The I th term computation includes an I -position left X shift as well as the multiplicands of a single Yi radix-r digit.

These equations will recursively be represented as

$$P[0] = 0$$
$$p[j + 1] = r{-}1(p[j] + r^n \text{ x } y_j)\ j = 0, 1, \ldots, n - 1$$
$$p = p[n]. \qquad (2)$$

In this way, it is articulated to guarantee that multiplying will start from its least significant digit of a multiplier y, thus preserving a certain location as regards the multiplicand x. An illustration of this is presented in Fig. 1.

BOOTH MULTIPLIER:

Booth algorithm is an optimal method for the partial product estimation and reduction. This procedure gives a short description of binary digits in the -2's complement representation signed. Below following methods were utilized to apply the booth algorithm. X and Y number of bits for two binary numbers which are A and B respectively.

1. Select multiplier (A) and multiplicand (B) of two digits.
2. New bits were obtained by encoding from 2's complement to the multiplicand (B).
3. Then those bits give partial products which is multiplied with multiplier (A).
4. It generates carry and sum by recoding adder which is obtained from above partial products.
5. In order to generate the outcome of an adder then carry and sum are added to the accurate adder.

3. Existing Work

RADIX-4 BOOTH MULTIPLICATION:

Each segment analyses the method [1] of the radix-4 booth, extending into serial parallel multiplier. It measures $x \times y$, when twos complement numbers were x along with y of n bit. Generating 2n two's complement in component p. Multiple digits of Y were computed with the N portion which is considered by the multiplication algorithm thus n is given as

$$N = \left\lceil \frac{n+2}{2} \right\rceil \qquad (3)$$

The expression defining the computation is shown through

$$p=(Y1 + Y0)\text{ x } +\sum_{i=1}^{N} 2^{2i-1}\,(Y2i{+}1 + Y2i - 2Y2i{-}1)x. \qquad (4)$$

TABLE I
RADIX-4 BOOTH RECODING

Y_{i+2}	Y_{i+1}	Y_i	Recoded value
0	0	0	0
0	0	1	+1
0	1	0	+1
0	1	1	+2
1	0	0	−2
1	0	1	−1
1	1	0	−1
1	1	1	0

According to Section II classification Y represents the multiplier y 's length-N digit function. The encoding e is determined by three digits of multiplier considering radix-4 booth algorithm

$$e_i = y_{2i+1} + y_{2i} - 2\,y_{2i-1} \qquad (5)$$

Algorithm 1 x, y Are n Bit Two's Complement Numbers, p Denotes the $2n$ Two's Complement Result, and the Shift Right Arithmetic (sra) Function. y Is Assigned to the n LSBs of p, Hence, the Encoding, E, Can Be Calculated Directly From P

```
Algorithm: Booth Radix-4 Multiplication
Data: $y$: Multiplier, $x$: Multiplicand
Result: $p$: Product
$p = y$;
$e = (P[0] - 2P[1])$;
for $count = 1$ to $N$ do
    $PartialProduct = e * x$;
    $p =$ sra($p$,2);
    $P[2 * B - 1 : B] \mathrel{+}= PartialProduct$;
    $e = (P[1] + P[0] - 2P[2])$;
end
```

From eq (5) where i indicates i th digit. As illuminated in Table I, apart from $y_{i+2}\ y_{i+1}\ y_i$ = 000 and $y_{i+2}\ y_{i+1}\ y_i$ = 111 gives outcomes in a 0, the multiplicand is scaled with either 1, 2, −2, or −1 dependent on encoding.

Encoding e_i is utilized to compute a partial product then $\mathrm{PartialProduct}_i$ by calculating

$$\mathrm{PartialProduct}_i = e_i\, x = (y_{2i+1} + y_{2i} - 2\,y_{2i-1})x \qquad (8)$$

The size of partial products of signed multiplication were reduced by the proposed various encoding scheme of booth algorithm. From those methods radix-4 booth algorithm were widely utilized for implementing signed multiply of hardware device.

Our suggested multiplier of radix 8 is therefore higher in comparison to the algorithms of radix 4.And since radix-8 recoding will provide time gaining advantage while summarizing the partial products as these partial products were mostly decreased to n/3 especially in comparison to n/2 in radix-4 for n bits of multiplier and multiplicand.

Some of the binary adders are summarized in this section. The most basic simple adder is the Ripple Carry Adder (RCA)[3][4] but it is the slowest with O(n) area and O(n) delay, where n is the operand size in bits. Carry Look-Ahead (CLA)[5][6] have O(n· log(n)) area and O(log(n)) delay, but typically suffer from irregular layout. An area efficient CLA is proposed by the authors in [7].

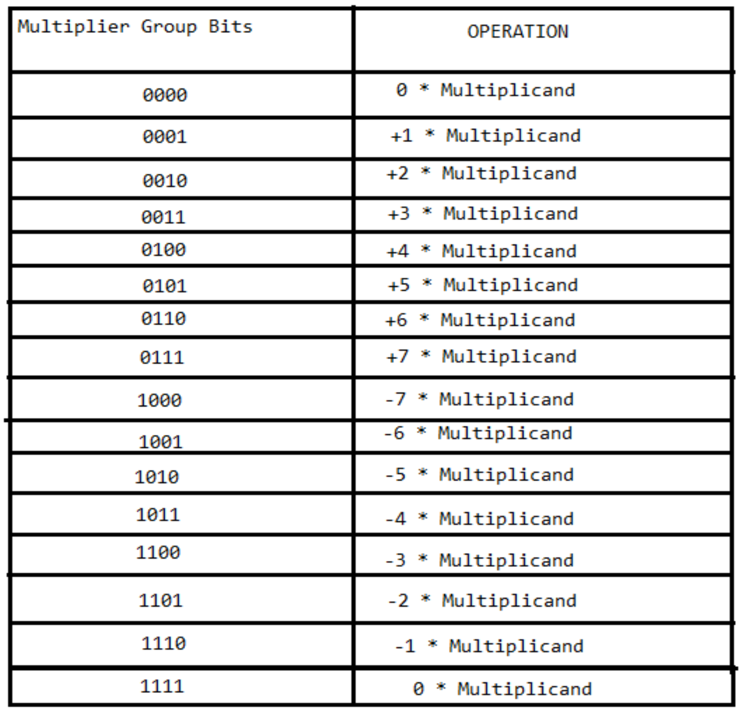
Some of the binary adders are summarized in this section. The most basic simple adder is the Ripple Carry Adder (RCA)[3][4] but it is the slowest with O(n) area and O(n) delay, where n is the operand size in bits. Carry Look-Ahead (CLA)[5][6] have O(n· log(n)) area and O(log(n)) delay, but typically suffer from irregular layout. An area efficient CLA is proposed by the authors in [7].

Some of the binary adders are summarized in this section. The most basic simple adder is the Ripple Carry Adder (RCA)[3][4] but it is the slowest with O(n) area and O(n) delay, where n is

the operand size in bits. Carry Look-Ahead (CLA)[5][6] have O(n· log(n)) area and O(log(n)) delay, but typically suffer from irregular layout. An area efficient CLA is proposed by the authors in [7].

4. Proposed Method

RADIX-8 BOOTH'S MULTIPLIER:
Radix multiplier efficiency can be improved by implementing parallelism where its outcomes in such a reduction in the amount of measurement steps.
Radix-8 means :8= 2^3= $(1000)_2$
Radix-8 uses 4-bit
Therefore, it requires a range of 4-bit binary numbers. The Signed Multiplier Digit is listed in table for a group of 4-bits. 2 explaining the recoding strategy of Radix-8 Booth for all feasible variations in the binary input where M is the multiplier.
Radix-8 recoding appears to apply a same technique as radix-4, but we are now taking quartets of bits rather than triplets. Signed digit codified by every quartet using the table below



Multiplier Group Bits	OPERATION
0000	0 * Multiplicand
0001	+1 * Multiplicand
0010	+2 * Multiplicand
0011	+3 * Multiplicand
0100	+4 * Multiplicand
0101	+5 * Multiplicand
0110	+6 * Multiplicand
0111	+7 * Multiplicand
1000	-7 * Multiplicand
1001	-6 * Multiplicand
1010	-5 * Multiplicand
1011	-4 * Multiplicand
1100	-3 * Multiplicand
1101	-2 * Multiplicand
1110	-1 * Multiplicand
1111	0 * Multiplicand

Table 2: Radix-8 Booth's recoding technique

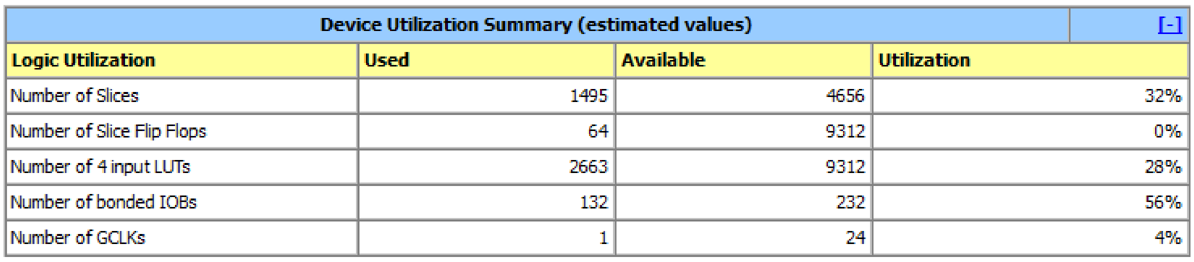
Multiplicator Radix-8 executes the 8 different categories of multiplicand operations that are + M, +2 M, +3 M, +4 M, – 4 M, – 3 M, – 2 M and – M where M specifies the multiplicand. All multiples excluding 3 M can be conveniently obtained by merely shifting and complementing to them.

The 3M series, which is refers as a hard multiple, will not be produced by easy shifting and complementing. It can either yield M+2 M, or 4M – M. This is generated here in this project by M+2M. For eg, the 8 partial product rows are created by a simple multiplier of 8 to 8 bit multiplication, but the proposed multiplier is reduced to 3. This indicates the radix-8 booth multiplier decreases partial component rows by N/3 where "N" is the multiplier bits.

This consists of four partial products. The Multiplier Sign Extension Trick for the Radix-8 Booth describes the partial products as given below:
[Partial Product 1]
[Partial Product 2]0 0 0
[Partial Product 3] 0 0 0 0 0 0
[Partial Product 4] 0 0 0 0 0 0 0 0 0

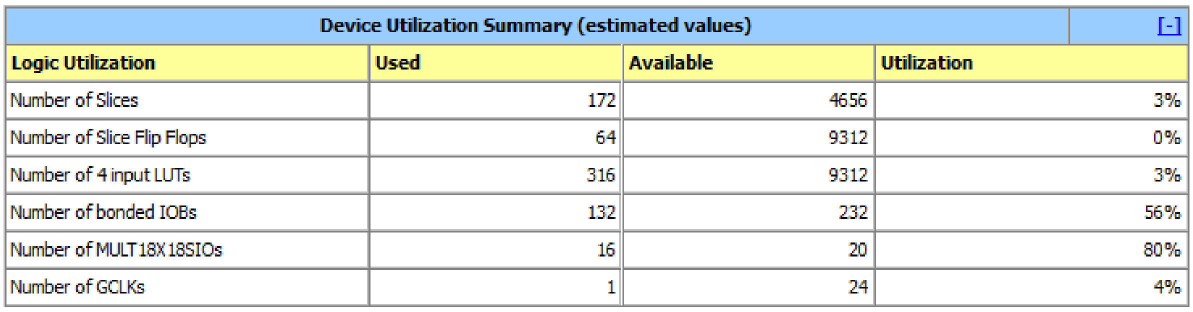
5. Results

The below two figures show the Device Utilization Summary of radix 4 and radix 8 two speed serial parallel multipliers
RADIX-4 DEVICE UTILIZATION



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1495	4656	32%
Number of Slice Flip Flops	64	9312	0%
Number of 4 input LUTs	2663	9312	28%
Number of bonded IOBs	132	232	56%
Number of GCLKs	1	24	4%

RADIX-8 DEVICE UTILIZATION



Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	172	4656	3%
Number of Slice Flip Flops	64	9312	0%
Number of 4 input LUTs	316	9312	3%
Number of bonded IOBs	132	232	56%
Number of MULT18X18SIOs	16	20	80%
Number of GCLKs	1	24	4%

COMPARISON

Design	**Area/Device Utilization**			**Delay**
	Slices	**LUTs**	**FFs**	
Radix-4	1495	2663	64	37.8ns
Radix-8	172	316	64	26.5ns

6. Conclusion

The Improved Radix- 8 Booth Multiplier in this paper found improved efficiency in terms of latency and region comparison with the existing estimated radix-4 Booth multiplier. It has been shown that it could be essential to develop a radix-8 architectural design for specific purposes in high-speed multipliers because of the time gaining. And while comparing with radix-4 the proposed radix-8 contains lesser amount of transistors and resulting in power dissipation and decreasing in area size. Which results by using higher radix-8 serial parallel multiplier there will be reduction in number of partial products.

7. References

[1] Duncan J. M. Moss , David Boland, and Philip H. W. Leong ”A Two-Speed, Radix-4, Serial–Parallel Multiplier” 2019.

[2] p. And D, J. Smidt. Boland in Proc, 'Efficient dot goods interactive bitwidth designation.' Int.: Int. Def.-Conf. Project Strategy. Training Application. Password. Password. Appl., September 2017, pp. 1–8.

[3] Ganesh Kumar, subhendu K Sahoo A High Performance and Low Power Transactions High Speed Multiplier Implementation, 2015.

[4] B. Dinesh, V. Kavinmalar, M. and V. Venkateshwaran Kathirvelu, "Comparison with updated 4bits booth encoding on the basis of standard and tree-based multiplier architectures utilizing 45 nm technology," in proc. Int.: Int. Conf.: Conf. Black machine. White device. Linked. Choose. Mar. 2014, pp. 1–6. architecture.

[5] S. Hetherington, T. M. Aamodt and A, J. Judd, J. Albericio, T. Moshovos, "Stripes: deep neural computation bit-serial," in Proc. Annu's 49th. ACM / IEEE Int. Sweet. Sweet. Oct. 2016, Microarchitecture,1-12 pp.

[6] G. C. T. Chow. C. T. Chow. A. H. T. Tse. Q. Jin. W. Luk. B. Thomas "Proc. A Precise Mixed Accelerator Process Monte Carlo Methodology." CHA / ISDR Int. Int. Sweet. Sweet. Project Strategy. Training Application. Line 47–57 of Gate Arrays, 2012.

[7] A. Suleiman, Z. [7] Sze, Y.-H. Chen, J. "Machine Learning Hardware: Problems and Chances,"CoRR, vol. abs/1612.07625, 2016 Dec. [Online]. [Online]. Disponible: Available: Available: https:/arxiv.org/aps, 017636.