An Encoder Based Radix -16 Booth Multiplier for Improving Speed and Area Efficiency

Nutakki Vijaya Kumari & K. Sampath Singh

2Assistant Professor, ECE Dept. Universal College of Engg & Tech. Perecherla, Guntur, A P, India 522438.

ABSTRACT: In this paper, we describe an optimization for binary radix-16 (modified) Booth recoded multipliers to reduce the maximum height of the partial product columns to \([n/4]\) for \(n = 64\)-bit unsigned operands. This is in contrast to the conventional maximum height of \([n + 1]/4\]. Therefore, a reduction of one unit in the maximum height is achieved. This reduction may add flexibility during the design of the pipelined multiplier to meet the design goals, it may allow further optimizations of the partial product array reduction stage in terms of area/delay/power and/or may allow additional addends to be included in the partial product array without increasing the delay. The method can be extended to Booth recoded radix-8 multipliers, signed multipliers, combined signed/unsigned multipliers, and other values of \(n\). The proposed architecture of this paper analysis the delay and area using Xilinx 14.3.

Index Terms—Binary multipliers, modified Booth recoding, radix-16.

1.INTRODUCTION

Binary multipliers are a widely used building block element in the design of microprocessors and embedded systems, and therefore, they are an important target for implementation optimization. Current implementations of binary multiplication follow the steps:

1) Recoding of the multiplier in digits in a certain number system;
2) Digit multiplication of each digit by the multiplicand, resulting in a certain number of partial products;
3) Reduction of the partial product array to two operands using multi-operand addition techniques; and
4) Carry-propagate addition of the two operands to obtain the final result.

The recoding type is a key issue, since it determines the number of partial products. The usual recoding process recodes a binary operand into a signed-digit operand with digits in a minimally redundant digit set. Specifically, for \(r\) \((r = 2m)\), the binary operand is composed of non-redundant \(r\)-digital by just making groups of \(m\) bits. Radix-4 modified Booth is a widely used recoding method that recodes a binary operand into a signed-digit operand with digits in a minimally redundant digit set. Specifically, for \(r\) \((r = 2m)\), the binary operand is composed of non-redundant \(r\)-digital by just making groups of \(m\) bits. Radix-4 modified Booth is a widely used recoding method that recodes a binary operand into a signed-digit operand with digits in the set \([-2, -1, 0, 1, 2]\). This is a popular recoding since the digit multiplication step to generate the partial products only requires simple shifts and complementation. The resulting number of partial products is about \(n/2\).

Higher radix signed recoding is less popular because the generation of the partial products requires odd multiples of the multiplicand which cannot be achieved by means of simple shifts, but require carry-propagate additions. However, the advantage of the high radix is that the number of partial products is further reduced. For instance,
for radix-16 and n-bit operands, about \( n/4 \) partial products are generated. Although less popular than radix-4, there exist industrial instances of radix-8 and radix-16 multipliers in microprocessors implementations.

### II. EXISTED SYSTEM

Fig. 1 shows a possible implementation of the partial product generation. Five bits of the multiplier \( Y \) are used to obtain the recoded digit (four bits of one digit and one bit of the previous digit to determine the transfer digit to be added).

The resultant digit is obtained as a one-hot code to directly drive a 8 to 1 multiplexer with an implicit zero output (output equal to zero when all the control signals of the multiplexer are zero).

The recoding requires the implementation of simple logic equations that are not in the critical path due to the generation in parallel of the odd multiples (carry-propagate addition). The XOR at the output of the multiplexer is for bit complementation (part of the computation of the two’s complement when the multiplier digit is negative).

In general, each partial product has \( n + 4 \) bits including the sign in two’s complement representation. The extra four bits are required to host a digit multiplication by up to 8 and a sign bit due to the possible multiplication by negative multiplier digits. Since the partial products are left-shifted four bit positions with respect to each other, a costly sign extension would be necessary. However, the sign extension is simplified by concatenation of some bits to each partial product.

After the generation of the partial product bit array, the reduction (multi-operand addition) from a maximum height of 17 (for \( n = 64 \)) to 2 is performed. The methods for multioperand addition are well known, with a common solution consisting of using 3 to 2 bit reduction with full adders (or 3:2 carry-save adders) or 4 to 2 bit reduction with 4:2 carry-save adders. The delay and design effort of this stage are highly dependent on the maximum height of the bit array.

### III. PROPOSED SYSTEM

The resultant digit is obtained as a one-hot code to directly drive a 8 to 1 multiplexer with an
**BOOTH ALGORITHM:** Booth multiplication algorithm or Booth algorithm was named after the inventor Andrew Donald Booth. It can be defined as an algorithm or method of multiplying binary numbers in two’s complement notation. It is a simple method to multiply binary numbers in which multiplication is performed with repeated addition operations by following the booth algorithm. Again this booth algorithm for multiplication operation is further modified and hence, named as modified booth algorithm.

A multiplier generator that creates a smaller number of partial products will allow the partial product summation to be efficient and use less hardware. The simple multiplication generator can be extended to reduce the number of partial products by grouping the bits of the multiplier into pairs, and selecting the partial products from the set of 0, M, 2M or their complements, where M is the multiplicand. This reduces the number of partial products, by a factor two but also generates some extra-bits for the sign extension and the 2’s complementation. All partial products set can be produced using simple shifting and complementing.

The multiplier is partitioned into overlapping groups of 3 bits, and each group is decoded to select a single partial product as per the selection table shown below. Each partial product is shifted 2 bit positions with respect to its neighbors. The number of partial products has been reduced to half of total number of multiplier bits. In general there will be n/2 products, where n is the operand length. The multiply by 2 can be obtained by a simple left shift of the multiplicand and negative of number obtained from its two’s complement form. Following table shows booth encoding table.

According to that partial products are generated and added to get final result.

**MODIFIED BOOTH ALGORITHM:** Booth multiplication algorithm consists of three major steps as shown in the structure of booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product.

**Modified Booth Algorithm Encoder:** This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm.

![Fig 2. Proposed system.](https://example.com/figure2.jpg)

This modified booth multiplier’s computation time and the logarithm of the word length of operands are proportional to each other. We can
reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or +1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding of every column of the booth multiplier. Thus, half of the partial product can be reduced using this booth algorithm. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4 booth encoder.

The overlapping is used for comparing three bits at a time. This grouping is started from least significant bit (LSB), in which only two bits of the booth multiplier are used by the first block and a zero is assumed as third bit.

IV. SIMULATION RESULTS

The below figures shows the simulation results of an encoder based radix-16booth multiplier for improving speed and area efficiency. The proposed is designed an encoder based radix-16 booth multiplier for improving speed and area efficiency in XILINX 14.7 Using VERILOG HDL code and simulated using Modelsim 6.5e To evaluate the efficiency of the proposed architecture.
V. CONCLUSION
Pipelined large wordlength digital multipliers are difficult to design under the constraints of core cycle time, pipeline depth, power and energy consumption and area. Low level optimizations might be required to meet these constraints. We have presented a method to reduce by one the maximum height of the partial product array for 64-bit radix-16 Booth recoded magnitude multipliers. This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier. We have shown that this reduction is achieved with no extra delay for $n \geq 32$ for a cell-based design. The method can be extended to Booth recoded radix-8 multipliers, signed multipliers and combined signed/unsigned multipliers.

VI. FUTURE SCOPE
As an attempt to develop arithmetic algorithm and architecture level optimization techniques for low-power multiplier design, the research presented in this dissertation has achieved good results and demonstrated the efficiency of high level optimization techniques. However, there are limitations in our work and several future research directions are possible. Higher-radix recoding further reduces the number of PPs and thus has the potential of power saving. Another possible direction can be representation of Arguments such as in sign-magnitude or 2’s compliment form which in any case would prove better according to situation and require less power and consume less time.

VII. REFERENCES