Design of Configurable Booth Multiplier Using Dynamic Range Detector

Deepak Garg¹ & Sandeep Arya²

Department of Electronics & Communication Engineering
Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India-125001
(E-mail: garg.deepu2018@gmail.com¹, arya1sandeep@gmail.com²)

Abstract: The main objective of this paper is to provide a solution for VLSI designers to design configurable Booth Multiplier that supports single 4-bit, single 8-bit, single 12-bit or single 16-bit multiplication. Multiplication is one of the basic functions used in digital signal processing for computation. Therefore, we need efficient algorithms to reduce the computations. Hence in order to minimize the number of cycles, a dynamic range detection technique is introduced to dynamically detect the effective dynamic ranges of two input operands so as to make the booth multiplier configurable. Using this technique, the unnecessary computation is truncated, so that the computation is done on the significant bits of both the operands.

Keywords: Configurable Booth Multiplier (CBM), Dynamic Range Detector (DRD).

1. INTRODUCTION

Multipliers have become a basic building block in computations especially in digital signal processing. Multipliers not only take a significant part of time delay, area cost but also cause high power consumption. To improve the speed and power dissipation of the multipliers, many techniques and design methodologies have been proposed. Most of the designs are targeted at a specific technology and require redesign for a new process technology. As a result, it is necessary to develop computation-efficient multipliers suitable for portable multimedia and digital processing systems, which require flexible processing ability, lesser switching activity and short design cycle.

The biggest challenge faced with use of simple and conventional multipliers for multimedia and DSP systems is that the multiplier coefficients are not constant. If it is constant, a general multiplier can be simplified to a network of shift, adders and subtractors to reduce power consumption [1]. However, this kind of simplified multiplier is inflexible which makes it to be unsuitable for multiplication operations with varying coefficients. To achieve improved processing ability, various techniques for reconfigurable multipliers that are capable of supporting multiple-precision multiplications have been developed. These multipliers are capable of decreasing the switching activity and minimizing the number of processing cycles. These techniques allow partitioned operands into multiple lower precision operands and performed several multiplications in parallel.

Zhou et al. [2] proposed reconfigurable multi-precision Radix-4 Booth Multiplier structure in which the number of bits can be extended by concatenating more stages together. On the other hand, Romain et al. [3] presented the error compensation method for truncated multiplication which helps to control the tradeoff between hardware cost and accuracy. However, these various multipliers do not take the power efficiency into consideration. Approaches termed as guarded evaluation [4] reduce the power consumption of multiplier to appreciable limit by eliminating spurious computation according to dynamic range of the input operands. The unnecessary computations in the sign extension part are truncated or removed, hence reducing the power consumption. Kim and Cho [5] described the pipeline architecture of high speed modified Booth Multiplier in which pipeline technique is used to accelerate the multiplication speed. The speed of multiplier is greatly improved by properly deciding the number of pipeline stages and the positions for the pipeline registers to be inserted. Various other multiplication algorithms such as Booth, modified Booth, Braun and Baugh Wooley have been proposed. The modified booth algorithm reduces the number of partial products to be generated.

In this paper, we try to attempt to combine configuration and dynamic range detection technique to design a configurable booth multiplier (CBM) that supports single 4-bit, single 8-bit, single 12-bit or single 16-bit multiplication. This CBM depends upon the output of proposed dynamic range detection technique with highly simplified circuit.

2. SIMPLE BOOTH MULTIPLIER

In 1951, Andrew Donald Booth devised a multiplication algorithm, which was named as Booth’s Algorithm. This algorithm multiplies two signed binary numbers in two’s complement notation. It makes repeated addition of one of
International Journal of Electronics Engineering

two predetermined values $A$ and $S$ to a product $P$ after which it performs a rightward arithmetic shift on $P$. Let $m$ and $r$ be the multiplicand and multiplier, respectively; and let $x$ and $y$ represent the number of bits in $m$ and $r$. It involves various steps as follows:

1. Determine the values of $A$ and $S$, and the initial value of $P$. All of these numbers should have a length equal to $(x + y + 1)$.

   - **A**: Fill the most significant (leftmost) bits with the value of $m$. Fill the remaining $(y + 1)$ bits with zeros.
   - **S**: Fill the most significant bits with the value of $(-m)$ in two’s complement notation. Fill the remaining $(y + 1)$ bits with zeros.
   - **P**: Fill the most significant $x$ bits with zeros. To the right of this, append the value of $r$. Fill the least significant bit with a zero.

2. Determine the two least significant (rightmost) bits of $P$.
   - If they are 01, find the value of $P + A$. Ignore any overflow.
   - If they are 10, find the value of $P + S$. Ignore any overflow.
   - If they are 00, do nothing. Use $P$ directly in the next step.
   - If they are 11, do nothing. Use $P$ directly in the next step.

3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let $P$ now equal this new value.

4. Repeat steps 2 and 3 until they have been done $y$ times.

5. Drop the least significant (rightmost) bit from $P$. This is the product of $m$ and $r$.

Hence, we can actually replace the multiplication by the string of ones in the original number by simpler operations, adding the multiplier, shifting the partial product thus formed by appropriate places, and then finally subtracting the multiplier. It is making use of the fact that we do not have to do anything but shift while we are dealing with 0s in a binary multiplier, and is similar to using the mathematical property that $99 = 100 - 1$ while multiplying by 99.

This scheme can be extended to any number of blocks of 1s in a multiplier (including the case of a single 1 in a block). Thus

$$M \times "001110110" = M \times (2^5 + 2^4 + 2^3 + 2^1) = M \times 62 \quad (4)$$

$$M \times "0100-1010" = M \times (2^6 - 2^3 + 2^1) = M \times 58 \quad (5)$$

Booth’s algorithm follows this scheme by performing an addition when it encounters the first digit of a block of one’s (01) and a subtraction when it encounters the end of the block (10). This works for a negative multiplier as well. When the ones in a multiplier are grouped into long blocks, Booth’s algorithm performs fewer additions and subtractions than the normal multiplication algorithm.

4. PROPOSED CONFIGURABLE BOOTH MULTIPLIER

4.1. Basic Architecture of Booth Multiplier

Here in this booth multiplier architecture there are various components that are performing certain task as follows:

1. A 16-bit register A that stores the multiplicand.
2. 16-bit parallel-load shift registers B that stores the multiplier initially, and the least significant 16 bits of the final multiplication product.
3. The 16-bit parallel-load shift registers ACCUMULATOR that is cleared initially and will store the most significant 16 bits of the final multiplication product. ACCUMULATOR and B are concatenated such that the bit that shifted out of ACCUMULATOR will be shifted into B. Also, the right shift operation is sign extended. For example, if ACCUMULATOR stores 1111111111111101 and B stores 010000000001111, then after concatenated right shift operation, ACCUMULATOR = 1111111111111110 and B = 1010000000001111. Note that the least significant bit in ACCUMULATOR (= 1) originally has been shifted to B as most significant bit.
4. The 16-bit ALU will pass zero or perform addition, subtraction, or shifting and finally pass its output to 16-bit ACCUMULATOR.
5. A CONTROL block Y that controls the operation of the multiplier.
(6) A 4-bit binary COUNTER K that give reference count to the CONTROL block.

4.2. Configurable Booth Algorithm

In this multiplication technique, firstly the size of both the operands A and B are detected by the dynamic range detector unit, it will detect whether the computation will be done on 4 bit, 8 bit, 12 bit or 16 bit. Thereafter the further computation will be done accordingly. A register PA is taken, that will store the concatenation of accumulator (4 bit or 8 bit or 12 bit or 16 bit, initially assigned with 0’s), multiplier A (4 bit or 8 bit or 12 bit or 16 bit), and an extra least significant bit, LSB (initially assigned with 0). PA [1:0] is checked whether it is 00, 01, 10, and 11. If PA [1:0] = 01, then PA + B operation will occur with single arithmetic right shift, else if PA [1:0] = 10, then PA – B will be calculated with single arithmetic right shift, otherwise only single arithmetic right shift will be done on PA. This whole procedure of checking of PA [1:0] bits will be done repeatedly and the number of iterations will depend upon the operand size. Finally, the final output will be saved in register P.

4.2.1. Dynamic Range Detector (DRD)

The dynamic range detector unit detects the effective dynamic range of input data and then generates the control signal. In this proposed multiplier, this control signal determines the flow of data. To simplify the implementation, the dynamic range detection can be realized by using the group of input bits. The functional block of dynamic-range determination unit includes neither logic gates like NOR, NAND and OR gates. The data detection starts from the most significant bits, examining each four bit group. Here in this detection technique, both the input operands A [16:0] and B [16:0] are divided into four parts that are A [15:12], A [11:8], A [7:4] and A [3:0], similarly B [15:12], B [11:8], B [7:4] and B [3:0]. The size of both operands is checked separately and simultaneously whether they come in the range of 4 bits, 8 bits, 12 bits or 16 bits. By doing this, the number of iterations or the number of partial product generation can be minimized to a certain limits. For example, if both the operands are in the range of 4 bits, then only 4x4 bit multiplication will take place, leaving behind the multiplication of rest of the bits. By doing so, the calculations will be cut shorter to much extent. With another advantage of simplicity, this circuit is preferred so as to suppress the most significant bits of the operands for multiplication if they are all zero.

5. SIMULATION RESULT BY XILINX ISE PROJECT NAVIGATOR

In the proposed design for n = 16 using Verilog HDL, there are two inputs, these are A [15:0] and B [15:0] and single output that is P [31:0]. If the input to A [15:0] is assigned as 1010 1100 1001 0101 (A \text{hex} = \text{ac95}) and B [15:0] as 0101 1100 1010 0101 (B \text{hex} = \text{5ca5}), that the output P [31:0] is expected to be 1110 0001 1100 1111 1100 1000 0000 1001 (P \text{hex} = \text{elefc809}). The result is verified by the simulation result. Here dynamic range is 16.
If the input to A [15:0] is assigned as 0000 0011 0101 0110 (A_{hex} = 075a) and B [15:0] as 0000 0010 1111 0011 (B_{hex} = 02f3), that the output P [31:0] is expected to be 0000 0000 0001 0101 0110 0110 1110 0110 (P_{hex} = 0015ae6e). The result is verified by the simulation result. Here dynamic range is 12.

If the input to A [15:0] is assigned as 0000 0000 0111 0111 (A_{hex} = 0077) and B [15:0] as 0000 0000 0101 0011 (B_{hex} = 0053), that the output P [31:0] is expected to be 0000 0000 0000 0000 0010 1101 0101 01001 (P_{hex} = 000002695). The result is verified by the simulation result. Here dynamic range is 8.

If the input to A [15:0] is assigned as 0000 0000 0000 0000 0010 0011 (A_{hex} = 0007) and B [15:0] as 0000 0000 0000 0101 0011 (B_{hex} = 0005), that the output P [31:0] is expected to be 0000 0000 0000 0000 0000 0000 0000 0000 0010 0011 (P_{hex} = 00000023). The result is verified by the simulation result. Here dynamic range is 4.

Here, Booth multiplier supports single 4-bit, single 8-bit, single 12-bit or single 16-bit multiplication depends upon the output of proposed dynamic range detection technique with highly simplified circuit.

6. CONCLUSION
This paper presents a solution to multiply single 4-bit, single 8-bit, single 12-bit or single 16-bits which depends upon the output of proposed dynamic range detection. The detection result is used to pick the operand with smaller dynamic range for Booth encoding to increase the probability of partial products becoming zero. It also deactivates the redundant switching activities in ineffective ranges as much as possible. Moreover, the output product of the proposed multiplier can be truncated to further decrease power consumption by sacrificing a bit of output precision. The proposed multiplier adopted due to its simple architecture outperforms over simple and conventional booth multiplier for these applications where truncation is performed. As a result, the proposed multiplier is very suitable for portable multimedia and DSP applications which require flexible processing ability, lesser switching activity and short design cycle.

References