DESIGN AND IMPLEMENTATION OF MAC UNIT FOR DSP APPLICATIONS USING VERILOG HDL

Amit kumar

amitjaiswalec162icfai@gmail.com

1PG Scholar, VLSI, Bhagwant University
Ajmer, Sikar Road Ajmer,Rajasthan,India.

Nidhi Verma

verma.nidhi17@gmail.com

2Assistant Professor, Bhagwant University
Ajmer, Sikar Road Ajmer,Rajasthan,India.

Abstract: In this paper, we proposed a new architecture of multiplier -and- accumulator (MAC) for high-speed arithmetic and low power. With the rapid advances in multimedia and communication system, high capacity signal processing are in demand, so High Speed MAC are essential to improve performance of signal processing System. Multiplication occurs frequently in finite impulse response filters, fast Fourier transforms, discrete cosine transforms, convolution, and other important DSP and multimedia kernels. The objective of a good multiplier and accumulator (MAC) is to provide a physically compact, good speed and low power consuming chip. The proposed SPST separates the target designs into two parts, i.e., the most significant part and least significant part (MSP and LSP), and turns off the MSP when it does not affect the computation results to save power. In this paper, we propose a high speed MAC adopting the new SPST implementing approach. This multiplier and accumulator is designed by equipping the Spurious Power Suppression Technique (SPST) on a modified Booth encoder which is controlled by a detection unit using an AND gate. The modified booth encoder will reduce the number of partial products generated by a factor of 2. The SPST adder will avoid the unwanted addition and thus minimize the switching power dissipation.

Keywords: Booth encoder, computer arithmetic, digital signal processing, spurious power suppression technique, low power.

I. INTRODUCTION

One of the accompanying challenges in designing ICs for portable electrical devices is lowering down the power consumption to prolong the operating time on the basis of given limited energy supply from batteries. Owing to the vigorous development of the wireless infrastructure and the personal electronic devices like video mobile phones, mobile TV sets, PDAs, etc., multimedia and DSP applications have been adopted in wireless environments. Increasing demands of high speed data signal processing motivated the researchers to seek fastest processors. The multiplier and multiplier-and-accumulator (MAC) [1] are the building blocks of the processor and has a great impact on the speed of the processor. MAC is the necessary element of the digital signal and image/audio processing system such as filtering, convolution and inner products hence high speed is crucial to develop for real processing applications. Many researchers have attempted in designing MAC for high computational performance and low power consumption. High throughput MAC is always a key factor to achieve high performance digital signal processing applications for real time signal processing applications. Since the multiplier requires the longest delay among the basic operation in digital system, the critical path is limited by the multiplier. Multiplier basically consists of three operational steps: Booth Encoder, Partial product reduction network (Wallace Tree) and final adder. For high speed multiplication, Modified Booth Algorithm (MBA) [4] is most commonly used, in which partial product is generated from Multiplicand (X) and Multiplier (Y) .Booth multiplication allows for the smaller ,faster multiplication circuits through encoding the signed bits to 2’s complement which is also the
standard technique in chip design and provide substantial improvement by reducing the partial products. Although the partial products are further reduced by using higher radix (4, 8, 16, 32) Booth Encoder which increases complexity and improves the performance.

II. OVER VIEW OF MAC

In this section, basic MAC operation is introduced. A multiplier can be divided into three operational steps. The first is radix-2 Booth encoding in which a partial product is generated from the multiplicand ($X$) and the multiplier ($Y$). The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry. The last is the final addition in which the final multiplication result is produced by adding the sum and the carry. If the process to accumulate the multiplied results is included, a MAC consists of four steps, as shown in Fig. 1, which shows the operational steps explicitly.

![Figure 1: Basic Arithmetic Steps of Multiplication and Accumulation](image)

In this paper, a new architecture for a high-speed MAC is proposed. In this MAC, the computations of multiplication and accumulation are combined and a hybrid-type CSA structure is proposed to reduce the critical path and improve the output rate. It uses MBA algorithm based on 1’s complement number system. A modified array structure for the sign bits is used to increase the density of the operands. A carry look-ahead adder (CLA) is inserted in the CSA tree to reduce the number of bits in the final adder. In addition, in order to increase the output rate by optimizing the pipeline efficiency, intermediate calculation results are accumulated in the form of sum and carry instead of the final adder outputs.

A general hardware architecture of this MAC is shown in Fig. 2. It executes the multiplication operation by multiplying the input multiplier and the multiplicand. This is added to the previous multiplication result as the accumulation step.

![Fig 2. Hardware architecture of general MAC.](image)

Modified Booth Encoder

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.
Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 3 shows the grouping of bits from the multiplier term for use in modified booth encoding.

![Fig.3.1 Grouping of bits from the multiplier term](image)

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1.

<table>
<thead>
<tr>
<th>Block</th>
<th>Recoded digit</th>
<th>Operation on X</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0 X</td>
</tr>
<tr>
<td>001</td>
<td>+1</td>
<td>+1 X</td>
</tr>
<tr>
<td>010</td>
<td>+1</td>
<td>+1 X</td>
</tr>
<tr>
<td>011</td>
<td>+2</td>
<td>+2 X</td>
</tr>
<tr>
<td>100</td>
<td>-2</td>
<td>-2 X</td>
</tr>
<tr>
<td>101</td>
<td>-1</td>
<td>-1 X</td>
</tr>
<tr>
<td>110</td>
<td>-1</td>
<td>-1 X</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>0 X</td>
</tr>
</tbody>
</table>

For the partial product generation, we adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2’s complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits. Figure 4, shows a computing example of Booth multiplying two numbers "2AC9" and “006A”. The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals. According to the analysis of the multiplication shown in figure 4, we propose the SPST-equipped modified-Booth encoder, which is controlled by a detection unit. The detection unit has one of the two operands as its input to decide whether the Booth encoder calculates redundant computations. As shown in figure 9. The latches can, respectively, freeze the inputs of MUX-4 to MUX-7 or only those of MUX-6 to MUX-7 when the PP4 to PP7 or the PP6 to PP7 are zero; to reduce the transition power dissipation. Figure 10, shows the booth partial product generation circuit. It includes AND/OR/EX-OR logic.

III. Partial product generator:

![Fig4.Booth partial product selector logic](image)
The multiplication first step generates from A and X a set of bits whose weights sum is the product P. For unsigned multiplication, P most significant bit weight is positive, while in 2’s complement it is negative.

The partial product is generated by doing AND between ‘a’ and ‘b’ which are a 4 bit vectors as shown in fig. If we take, four bit multiplier and 4-bit multiplicand we get sixteen partial products in which the first partial product is stored in ‘q’. Similarly, the second, third and fourth partial products are stored in 4-bit vector n, x, y.

The multiplication second step reduces the partial products from the preceding step into two numbers while preserving the weighted sum. The sought after product P is the sum of those two numbers. The two numbers will be added during the third step. The "Wallace trees" synthesis follows the Dadda’s algorithm, which assures of the minimum counter number. If on top of that we impose to reduce as late as (or as soon as) possible then the solution is unique. The two binary numbers to be added during the third step may also be seen as a one number in CSA notation (2 bits per digit).

III. Existing System

NR4SD Encoding Scheme

The following Boolean equations summarize the HA* operation:

\[ c_{3j+2} = b_{3j+1} \lor c_{3j+1}, \quad n_{3j+1} = b_{3j+1} \oplus c_{3j+1}. \]

Calculate the value of the 8th digit.
Table 2 shows how the NR4SD digits are formed. The NR4SD encoding signals $\textit{one}_{j}$, $\textit{two}_{j}$ and $\textit{two}_{j}$ of Table 2 are generated.

<table>
<thead>
<tr>
<th>$k_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

For the computation of the least and the most significant bits of the partial product we consider and respectively. Note that in case that the number of the resulting partial products is and the most significant MB digit is formed based on sign extension of the initial 2’s complement number.

After the partial products are generated, they are added, properly weighted, through a Carry-Save Adder (CSA) tree.

Finally, the carry-save output of the Wallace CSA tree is leaded to a fast Carry Look Ahead (CLA) adder to form the final result $Z = X \cdot Y$.

Fig. 8. Block Diagram of the NR4SD+ Encoding Scheme at the (a) Digit and (b) Word Level.

Calculate the value of the $b_{j}^{NR+}$ digit.

Table 3 shows how the NR4SD digits are formed. The NR4SD encoding signals $\textit{one}_{j}$, $\textit{two}_{j}$ and $\textit{two}_{j}$ of Table 3 are generated.

<table>
<thead>
<tr>
<th>$k_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
<th>$b_{j}$</th>
<th>$c_{j}$</th>
<th>$e_{j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 9. System Architecture of the NR4SD Multipliers

In the pre-encoded MB multiplier scheme, the coefficient $B$ is encoded off-line according to the conventional MB form (Table 1). The resulting encoding signals of $B$ are stored in a ROM. The circled part of Fig. 3, which contains the ROM with coefficients in 2’s complement form and the MB encoding circuit, is now totally replaced by the ROM. The MB encoding blocks of Fig. 3 are omitted. The new ROM is used to store the encoding signals of $B$ and feed them into the partial product generators (Pj Generators - PPG) on each clock cycle. Targeting to decrease switching activity, the value ‘1’ of $s_{j}$ in the last entry of Table 1 is replaced by ‘0’. The sign $s_{j}$ is now given by the relation:
However, the ROM width is increased. Each digit requests three encoding bits (i.e., s, two and one (Table 1)) to be stored in the ROM. Since the n-bit coefficient B needs three bits per digit when encoded in MB form, the ROM width requirement is $3n/2$ bits per coefficient. Thus, the width and the overall size of the ROM are increased by 50% compared to the ROM of the conventional scheme.

The system architecture for the pre-encoded NR4SD multipliers is presented in Fig. 6. Two bits are now stored in ROM: $n_{2j+1}$, $n_{2j+1}$ (Table 2) for the NR4SD or $n_{2j+1}$, $n_{2j}$ (Table 3) for the NR4SD+ form. In this way, we reduce the memory requirement to +1 bits per coefficient while the corresponding memory required for the pre-encoded MB scheme is $3n/2$ bits per coefficient. Thus, the amount of stored bits is equal to that of the conventional MB design, except for the most significant digit that needs an extra bit as it is MB encoded. Compared to the pre-encoded MB multiplier, where the MB encoding blocks are omitted, the pre-encoded NR4SD multipliers need extra hardware to generate the signals of (6) and (8) for the NR4SD and NR4SD+ form, respectively.

Each partial product of the pre-encoded NR4SD and NR4SD+ multipliers is implemented based on Fig. 4c and 4d, respectively, except for the $P_{k1}$ that corresponds to the most significant digit. As this digit is in MB form, we use the PPG of Fig. 4b applying the change mentioned in Section 4.2 for the $s_{j}$ bit. The partial products, properly weighted, and the correction term (COR) of (11) are fed into a CSA tree. The input carry $c_{in;j}$ of (11) is calculated as $c_{in;j} = twoj \cdot onej$ and $c_{in;j} = onej$ for the NR4SD and NR4SD+ form, respectively. The carry-save output of the CSA tree is finally summed using a fast CLA adder.

IV. PROPOSED SPST

Besides the explanations presented in our former studies, this paper provides further illustrations of the proposed SPST as described in the following sections.

The SPST uses a detection logic circuit to detect the effective data range of arithmetic units, e.g., adders or multipliers. When a portion of data does not affect the final computing results, the data controlling circuits of the SPST latch this portion to avoid useless data transitions occurring inside the arithmetic units. Besides, there is a data asserting control realized by using registers to further filter out the useless spurious signals of arithmetic unit every time when the latched portion is being turned on. This asserting control brings evident power reduction. Figure 5 shows the design of low power adder/subtractor with SPST.

![Fig 10. Spurious transition cases in multimedia/ DSP processing](image-url)

The adder/subtractor is divided into two parts, the most significant part (MSP) and the least significant part (LSP). The MSP of the original adder/subtractor is modified to include detection logic circuits, data controlling circuits, sign extension circuits, logics for calculating...
carry in and carry out signals. The most important part of this study is the design of the control signal asserting circuits, denoted as asserting circuits in Figure 5. Although this asserting circuit brings evident power reduction, it may induce additional delay. There are two implementing approaches for the control signal assertion circuits. The first implementing approach of control signal assertion circuit is using registers. This is illustrated in Figure 6. The three output signals of the detection logic are close, Carr_ctrl, sign. The three output signals the detection logic unit are given a certain amount of delay before they assert. The delay, used to assert the three output signals, must be set in a range of, denotes the data transient period the earliest required time of all the inputs. This will filter out the glitch signals as well as to keep the computation results correct. The restriction that must be greater than to guarantee the registers from latching the wrong values of control usually decreases the overall speed of the applied designs.

When the detection-logic unit remains its decision:

No matter whether the last decision is turning on or turning off the MSP, the delay of the detection logic is negligible because the path of the combinational circuitry (i.e., the 16-bit adder/subtractor in this design example) remains the same. From the analysis earlier, we can know that the total delay is affected only when the detection-logic unit turns on the MSP. However, the detection-logic unit should be a speed-oriented design. When the SPST is applied on combinational circuitries, we should first determine the longest transitions of the interested cross sections of each combinational circuitry, which is timing characteristic and is also related to the adopted technology. The longest transitions can be obtained from analyzing the timing differences between the earliest arrival and the latest arrival signals of the cross sections of a combinational circuitry. Then, a delay generator similar to the delay line used in the DLL.

V. Results

Simulation Results of MAC:
CONCLUSIONS

In this project, we propose a high speed low-power multiplier and accumulator (MAC) adopting the new SPST implementing approach. This MAC is designed by equipping the Spurious Power Suppression Technique (SPST) on a modified Booth encoder which is controlled by a detection unit using an AND gate. The modified Booth encoder will reduce the number of partial products generated by a factor of 2. The SPST adder will avoid the unwanted addition and thus minimize the switching power dissipation. The SPST MAC implementation with AND gates have an extremely high flexibility on adjusting the data asserting time. This facilitates the robustness of SPST can attain 30% speed improvement and 22% power reduction in the modified Booth encoder. This design can be verified using Modelsim and Xilinx using verilog.

Future Scope:

The proposed system can be done using Dadda multiplier, by using this delay will be reduced.

The process of Dadda multiplication is as follows: The entire $16 \times 16$ multiplication requires six stages. Always the first stage is partial products stage, which is obtained by simple multiplication of multiplicand with multiplier. The number of rows (height) present at this stage is 16. Now reduce the number of rows further in such a way that final stage contains only two rows. For this, Dadda introduces a sequence of intermediate matrix heights that provides the minimum number of reduction stages for a given size multiplier. This sequence determined by working back from the final two row matrix, limit the height of each intermediate matrix to the largest integer that is no more than 1.5 times the height of its successor.

REFERENCES


