Design of Modified Approximate Multiplier by Truncation-Rounding for Efficient Processing

Nisarga G1, Dr. Vijaya Prakash. A. M2
M. Tech Scholar1, Professor2
Department of Electronics and Communication Engineering
Bangalore Institute of Technology, Bangalore, India

Abstract:
A multiplier generally takes up a significant role in the signal processing systems. The main design requirements in any such digital system is to optimize the processing speed and area consumption of the digital design. The hardware resource necessities and speed of processing in any multiplier is crucial for proving its performance. So, this viewpoint leads to the incorporation of an approximate computing approach that elevates the multiplier performance in any processing system. The project proposed analyses an approximate computing multiplier of input bit widths of 8-bit, 16-bit and 32-bit with truncation and rounding technique that limits the amount of the partial products generated in a multiplication process thereby optimizing the area and speed design constraints with a small accuracy compromise. The proposed multiplier is scalable for n-bit multiplication. The proposed multiplier structure further incorporates a modified parallel prefix adder unit that speeds up the pace of operation. The multiplier proposed has better efficiency and results up to 90.8% improved speed and 81.99% hardware savings in the 32-bit multiplier design. The approximate computed multiplier can be made use of in error-resilient applications where the output quality changes are negligible to human perception. Furthermore, the proposed multiplier structure is employed and analyzed within an image enhancement application.

Keywords: Approximate Computing, Area-Efficient, High-Speed, Parallel Prefix Adder, Rounding, Truncation.

I. INTRODUCTION

The main design requirement in any digital system is to optimize the processing speed, area and energy consumption of the digital design. Attaining the optimization of design parameters in any system unit, has become a fundamental design goal for any compact and portable embedded computing device. To boost the efficiency of such computing devices, considerable efforts have been made at various levels, from the macro software level to the micro architecture level. These computing devices are often required to implement some key digital signal operations, classification and machine learning applications. In many such applications, the hardware resource requirement, the speed and the energy dissipation of a design is dominated by the arithmetic computing used. The multiplication operation with its frequent usage is a high delay causing, energy and hardware consuming operation that takes up a major share of energy usage than the other arithmetic functions in the ALU (Arithmetic and Logical Unit) processing block. Consequently, an improvement in speed, area requirements of multipliers plays an essential part in improvement of efficacy of the processors in any digital system. In the analysis of a suite of applications, it was seen that, 83% of runtime computations, on an average, can resist approximation to some extent [1]. These applications after employing approximate multipliers have negligible output quality changes and because of the restricted perceptual abilities of the user while perceiving an image or video, approximations are a great choice to improve the design parameters such area, speed and power. Generally, a multiplication operation begins with input operands being multiplied in the initial stage to generate the partial products. Later, they are reduced to two levels using reduction techniques. And lastly, the two levels are summated using a fast adder architecture. As multiplications form a major part of operations in processing blocks, improving its efficiency proves beneficial to the entire system. Hence, employing approximations at every stage enhances the overall multiplier performance. Initially, approximation can be employed to reduce the partial products generation and improve the area, power and speed design parameters. In the next stage that involves reducing of levels of partial products accumulated, approximate compressors can be utilized to attain optimizations in speed of the design. Lastly, while obtaining the final multiplication product by addition, where fast adder architectures are employed, approximations can be made use of to pace up the addition. In the proposed approximate computing multiplier, the method of extraction from truncating bits and rounding off values along with the utilization of a modified parallel prefix adder to optimize and obtain a high-speed performance and low area utilization multiplier is suggested. The ordering of the paper is done as follows. The design methodology of the approximation technique and modified parallel prefix adder is presented in Section II. The proposed design units of 8bit, 16bit and 32bit approximate multipliers is presented in Section III. The Section IV discusses the simulation and synthesis results of the approximate multiplier proposed and the image processing application employing the designed approximate multiplier. At last, the conclusions of the designed multiplier can be observed in section V.

II. DESIGN METHODOLOGY

A. Approximate Computation
In accordance with the mathematical modelling of equations [2], the approximate computation multiplier with the technique of truncation and rounding can be designed.
The exact multiplication of the input operand values is delay causing and consumes high energy. Hence, the computation of the approximate value of these terms is carried out. The approximate term is known by splitting the range into identical partitions. An illustration on how the approximation of a value is done is as shown below.

\[
\begin{align*}
\gamma_{APX}=1/8 & & \gamma_{APX}=3/8 & & \gamma_{APX}=5/8 & & \gamma_{APX}=7/8 \\
0/4 & & 1/4 & & 2/4 & & 3/4 & & 4/4
\end{align*}
\]

**Figure2.1. Approximation of a value for a four segment case**

In the above figure 2.1, the rounding off value, \( h \) is selected to be 2, the total number of segments, a value \( Y \) is divided into is 4 \((2^2)\). Depending on the \( Y \) value, the approximation is done. Any value of \( Y \) that lies between 0 to 1/4, the approximation done is 1/8. Similarly, the other values of \( Y \) ranging between 1/4 to 2/4 approximates to the value \( Y_{APX}=3/8 \); the \( Y \) range between 2/4 to 3/4 approximates to value 5/8 and the \( Y \) range between 3/4 to 1 approximates to 7/8.

The approximate value is procured merely by curtailing \( Y \) value to \( h \) number of bits and insetting “1” bit to the right of the curtailed \( Y \). The curtailing of input values to ‘1’ bits improves the performance speed and is indicated by \( (Y_{p}) \) and \( (Y_{q}) \). Therefore, the equation after employing truncation and rounding in multiplication of two numbers \( P \) and \( Q \) is expressed as,

\[
(P \times Q)_{APX} = 2^{p+4}x(1+(Y_{p})(Y_{q})) \quad (1)
\]

where the bit length of approximate terms of \( P \) and \( Q \) is \( h+1 \) bits and \( k_{P} \) and \( k_{Q} \) denotes the location of the first occurring one bit in \( P \) and \( Q \) values respectively. The multiplier provides better tradeoff between approximation results, speed and area constraints at \( t=h+4 \) values [2], which is considered in the further design. Using the above approximation methodology, the resulting partial products amount decreases that benefits the area requirement and operation speed of a multiplier.

**B. Modified Parallel Prefix Adder**

The adders used for arithmetic binary operations have a major impact on the performance parameters in any processing unit. The adder unit in a multiplier is responsible for the summation of partial products generated. So, employing an efficient fast adder, like parallel prefix adder (ppa) in this stage impacts the multiplier performance greatly.

**Figure2.2. Process flow OF THREE-stage parallel prefix adder**

The parallel adder structures have the below sequence of steps in their operation.

**Step1:** Pre-Computation of propagate and generate terms

\[ p_{i} = x_{i} \land y_{i} \quad \text{and} \quad g_{i} = x_{i} \lor y_{i} \quad (2) \]

**Step2:** Carry-Generation stage is unique to the parallel prefix adders that describes the carry formation structure. At this stage, Ling[3] proposed an alteration to the carry-lookahead adder for hardware savings of one logic element, where the technique of calculating a simpler pseudo carry \( (h_{i}) \) can be employed.

\[ h_{i} = g_{i} \lor g_{i-1} \land p_{i-1} \land g_{i-2} \land p_{i-2} \land g_{i-3} \ldots \ldots + p_{1} \land p_{2} \ldots p_{6} \]  

(3)

The actual carry \( (c_{i}) \) is generated by using the following equation,

\[ c_{i} = h_{i} \land p_{i} \quad (4) \]

But the sum calculation is complex and the calculation of carry here increases the power and hardware consumption of the adder structure. Hence, a modified Ling equation[3] was proposed where the carry calculation is done by utilizing the intermediate generate and propagate signals are given by (5) and (6)

\[
\begin{align*}
G_{i(j)} &= G_{i(j)} + G_{i(j) \lt P_{i(j)\lt P_{i(j+k)}} \quad (5) \\
P_{i(k)} &= P_{i(j)} - P_{i(j+k)} \quad (6)
\end{align*}
\]

The carry signal in terms of the intermediate signals (where \( i>k>j \)) is given by

\[ c_{i} = (G_{i(k)} + P_{i(j+1)} - G_{i(k+1)}) \land p_{i} \quad (7) \]

Therefore, the final sum calculations get easier and at every bit of sum calculation, one logic gate is saved. Hence, the overall area of the adder unit decreases.

**Step3:** Post-Computation involves the final sum calculation given by the equation,

\[ \text{sum} = x^{y} \times c_{i} \quad (8) \]

The modified parallel prefix adder has an ordered structure with lowered gate count that decreases the delay of propagation of the overall adder unit. Hence, utilizing this adder unit in the computation unit of the suggested approximate multiplier where the partial product terms are summed up, can further enhance the performance of the multiplier with its speed up operation. Overall, the multiplier performance leads to having a better and efficient processing unit.

**III. DESIGN OF APPROXIMATE MULTIPLIER**

The proposed approximate multiplier architecture utilizes curtailment and rounding method to improve delay and area requirement constraints, along with the usage of a modified parallel prefix adder in the computation unit for further speed improvement. The design of approximate multipliers of bit widths 8, 16 and 32 are proposed as shown in the following figures 3.1, 3.2 and 3.3 respectively.

**Figure3.1. 8X8 bit approximate multiplier (with h=1 and t=5) architecture using PPA**

The approximate multiplier architecture for the 8-bit inputs multiplication for \((1,5)\) truncation and rounding values along with the usage of a 5-bit parallel prefix adder is designed as observed in figure 3.1.
The approximate multiplier architecture for the 16-bit inputs multiplication for (3,7) truncation and rounding values using a 7-bit parallel prefix adder is designed as observed in figure 3.2. Similarly, as observed in figure 3.3, a 32-bit approximate multiplier architecture for (6,10) values employing a 10-bit and 14-bit parallel prefix adder is designed.

In the block diagram, the input signals, P and Q are fed to the Foremost One Detector block to find the locations of their foremost one-bit place denoted by signals kP and kQ are used for extraction of bits in the input signals in accordance with h and t values for further operation. In the Curtailment Unit, the input signals along with foremost one-bit locator positions are enforced, where with the application of truncation 't' value, the truncated values are generated. The results are moved to the computational block to compute the approximate expression as in (1). The core block here is the adder part that performs the summation of the above expression. The adder block proposed here is modified parallel prefix adder, that increases the swiftness of carry generation, which thus greatly improves the speed of the suggested multiplier. The shifting block expresses the resultant output of a n-bit input multiplication in the form of 2n-output bits result, in accordance to the summation of kP and kQ combined value. In the Zero operand locator block, if by leastways one of the input operands is zero, the output result is made zero.

IV. RESULTS AND DISCUSSION

C. Simulation and Synthesis of Approximate Multipliers
The simulation results of the proposed approximate multiplier with truncation and rounding technique along with the usage of the fast adder performed in Xilinx ISE 14.7, is obtained as observed in the following waveforms. The simulation outputs of the approximate 8-bit, 16-bit and 32-bit multipliers are as follows.

The above figure 4.1 shows the numeric example of a 8-bit approximate multiplier (1,5) with input values 75 and 89. The approximate output is observed to be 6,528 while the exact output equals to 6675, with a relative error of 2.20%. The average relative error observed is about 4.30% in the 8-bit multiplier.

The figure 4.2 shows the numeric example of a 16-bit approximate multiplier (3,7) with input values 16,571 and 4,281. The approximate output is observed to be 7,05,16,736 while the exact output equals to 7,09,40,451, with a relative error of 0.59%. The average relative error observed is about 3.47% in the 16-bit multiplier.

The figure 4.3 shows the numeric example of a 32-bit approximate multiplier (3,7) with input values 16,571 and 4,281. The approximate output is observed to be 7,05,16,736 while the exact output equals to 7,09,40,451, with a relative error of 0.59%. The average relative error observed is about 3.47% in the 16-bit multiplier.
The figure 4.3 shows the numeric example of a 32-bit approximate multiplier (6,10) with input values 3.58,920 and 13.85,698. The approximate output is observed to be 4,96,99,14,69,568 while the exact output equals to 4,97,35,47,26,160, with a relative error of 0.07%. The average error in accuracy is observed to be 1.06% in the 32-bit multiplier. The results of delay and area constraints of the designed 8-bit, 16-bit, 32-bit approximate multipliers in FPGA Spartan 3e are observed to be as follows as shown in the below table I.

Table 1. Timing and area Report OF the proposed approximate multipliers

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Approximate multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8X8 bit</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>5.921</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>168.89</td>
</tr>
<tr>
<td>4 I/P LUTs</td>
<td>149</td>
</tr>
</tbody>
</table>

Table 2. Timing and area comparison report of various multipliers

<table>
<thead>
<tr>
<th>Multiplier Type</th>
<th>Delay (ns)</th>
<th>Frequency (MHz)</th>
<th>LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Approximate Multiplier</td>
<td>8X8</td>
<td>5.921</td>
<td>168.89</td>
</tr>
<tr>
<td>Approximate Multiplier</td>
<td>16X16</td>
<td>7.516</td>
<td>133.056</td>
</tr>
<tr>
<td></td>
<td>32X32</td>
<td>8.061</td>
<td>124.060</td>
</tr>
<tr>
<td>Rounding Approximate Multiplier [6]</td>
<td>8X8</td>
<td>33.012</td>
<td>30.290</td>
</tr>
<tr>
<td></td>
<td>16X16</td>
<td>51.090</td>
<td>19.57</td>
</tr>
<tr>
<td></td>
<td>32X32</td>
<td>88.114</td>
<td>11.34</td>
</tr>
</tbody>
</table>

From the comparison and analysis of multipliers synthesized in FPGA Spartan 3e in Table II, the proposed approximate multiplier with truncation and rounding technique with the utilization of parallel prefix adder proves to be better performing both in terms of its processing speed and area consumption. The delay of the proposed multipliers has reduced greatly, almost ten times less in the case of 32-bit width multiplier compared to the existing 32-bit multiplier mentioned. The number of LUTs in the proposed multiplier has also greatly decreased in higher bit widths comparatively as observed in the table.

D. Approximate Multiplier utilization in an application

The designed approximate multiplier is employed in an image enhancement application that enhances each of the color component quality in the input image to output an enhanced image. The process flow of the application is as follows.

The pre-processing of input image is done initially and then the input pixels are read into the enhancement algorithm for processing where the proposed approximate multiplier is utilized. In the enhancement algorithm, the input image of RGB channel undergoes colour conversion to form separate channels. To increase the speed of operation in processing, the whole image is divided into small subgroups of size 3X3. The luminous operation on the resulting image is carried out. The resultant is applied for histogram equalization to almost level out the amplitudes of output signal. So, on applying all the above techniques, an enhanced image is obtained, for which the PSNR value is recorded and analysed.

![Figure 4.5](http://ijesc.org/) The Baboon (a) input image (b) enhancement images with exact multiplier with PSNR=18.0842 (c) enhancement with proposed approximate multiplier with PSNR=18.0626

From the above figure 4.5, the input image and output enhanced images after applying the image enhancement algorithm can be observed. The designed approximate multiplier is exploited in the image enhancement application and by observing the results, it can be analyzed that even though the proposed approximate multiplier with approximate result is used, it doesn’t affect the quality of the output image. There is negligible PSNR quality difference of 0.0216 dB observed in the above figure that is almost impossible to distinguish on human perception level.

On analyzing, the average PSNR differences between the exact and approximate computations is almost ≤1dB. Hence, employing these approximations in multiplications benefits the improvement in speed and area efficiency of the overall design with negligible quality trade-off. Therefore, utilizing approximate multipliers in any such image processing applications prove to be a great choice to improve the design parameters of the entire processing system.

V. CONCLUSION AND FUTURE SCOPE

A high speed and area-efficient 8-bit, 16-bit and 32-bit width approximate multipliers utilizing the technique of truncation and rounding that reduces the amount of partial products generation is achieved. The proposed multiplier structure further incorporates a modified parallel prefix adder unit that speeds up the pace of operation, enhancing the multiplier speed performance. With increase in the multiplier width, the speed and area improvements of the designed multiplier is achieved due to its simple calculation unit.

Also, the utilization of the designed approximate multiplier is done in an image enhancement application showing insignificant quality difference in the output. The proposed multiplier can be further scaled and be extended to higher order bit multiplication and the performance of the multiplier, including the power and area constraints can be improved.
VI. REFERENCES


