Volume 10 /Issue 1 / AUG 2018

An Efficient Design of Approximate Multipliers

[1]T.BHANU CHANDAR REDDY (M.Tech)[2]Mr.P.SANTHOSHKUMAR$_{\text{M.Tech}}$, Assistant.professor

[1,2]VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, GUNTUR, NAMBUR

1. **Email-id:** : [1]bhanuchandarreddy07@gmail.com[2]santhosh.peketi@gmail.com

Abstract: **In this paper, an approximate multiplier that is high speed yet energy efficient, for error resilient applications is proposed. In this new design approach for 16 bit approximation of multiplier, partial products of the multiplier are altered to introduce varying probability terms. Logic complexity of approximation is varied for the accumulation of altered partial products based on their probability. Synthesis results reveal that proposed multiplier achievesbetter performance, compared to an exact multiplier when done in XILINX 13.2. They have better precision when compared to existing approximate multipliers.**

Key Terms— Approximate computing, error analysis, low error, low power, multipliers.

I. INTRODUCTION

Energy minimization is one of the main requirements in almost any electronic systems, especially the portable ones such as smart phones, tablets, and different gadgets. It is highly desired to achieve this minimization with minimal performance (speed) penalty. Digital signal processing (DSP) blocks adders and multipliers are key components of these portable devices for realizing various multimedia applications. The computational core of these blocks is the arithmetic logic unit where multiplications have the greatest share among all arithmetic operations performed in these DSP systems. Therefore, improving the speed and power/energy-efficiency characteristics of multipliers plays a key role in improving the efficiency of processors. Many of the DSP cores implement image and video processing algorithms where final outputs are either images or videos prepared for human consumptions. This fact enables us touse approximations for improving the speed/energy efficiency. This originates from the limited perceptual abilities of human beings in observing an image or a video. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system. Being able to use the approximate computing provides the designer with the ability of making tradeoffs between the accuracy and the speed as well as power/energy consumption. Applying the approximation to the arithmetic units can be performed at different design abstraction levels including circuit, logic, and architecture levels, as well as algorithm and software layers. The approximation may be performed using different techniques such as allowing some timing violations (e.g., voltage overscaling or overclocking) and function approximation methods (e.g., modifying the Boolean function of a circuit) or a combination of them. In the category of function approximation methods, a number of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested.

To reduce hardware complexity of multipliers, truncation is widely employed in fixed-width multiplier designs. Then a constant or variable correction term is added to compensate for the quantization error introduced by the truncated part. Approximation techniques in multipliers focus on accumulation of partial products, which is crucial in terms of power consumption. Broken array multiplier is implemented, where the least significant bits of inputs are truncated, while forming partial products to reduce hardware complexity. The proposed multiplier in saves fewadder circuits in partial product accumulation.

II. LITERATURE REVIEW

Low-Power Digital Signal Processing Using Approximate Adders:

Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather useful information from slightly erroneous outputs. Therefore, we do not need to produce exactly correct numerical outputs. Previous research in this context exploits error resiliency primarily through voltage over scaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. In this paper, we propose logic complexity reduction at the transistor level as an alternative approach to take advantage of the relaxation of numerical accuracy. We demonstrate this concept by proposing various imprecise or approximate full adder cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders. In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. We design architectures for video and image compression algorithms using the proposed approximate arithmetic units and evaluate them to demonstrate the efficacy of our approach. We also derive simple mathematical models for error and power consumption of these approximate adders. Furthermore, we demonstrate the utility of these approximate adders in two digital signal processing architectures (discrete cosine transform and finite impulse response filter) with specific quality constraints. Simulation results indicate up to 69% power savings using the proposed approximate adders, when compared to existing implementations using accurate adders.

The variable correction truncated multiplier is introduced. This is a method for minimizing the error of a truncated multiplier. The error is reduced by using information from the partial product bits of the column adjacent to the truncated LSB. This results in a complexity savings while introducing minimum distortion to the result.

III. EXACT MULTIPLIER.

Booth Multiplier:

Multiplication consists of three steps:
1) The first step to generate the partial products;
2) The second step to add the generated partial products until the last two rows are remained;
3) The third step to compute the final multiplication results by adding the last two rows.
The modified Booth algorithm reduces the number of partial products by half in the first step. We used the modified Booth encoding (MBE) scheme proposed in. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0, 1, 2}. Table I shows the rules to generate the encoded signals by MBE scheme shows the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals as show.

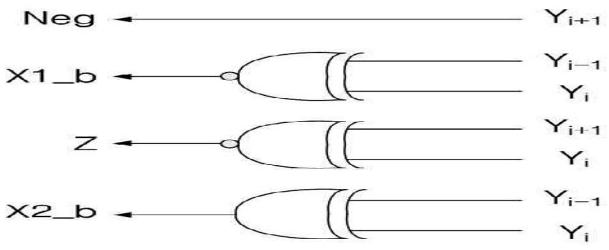


Fig.1 Booth Encoder

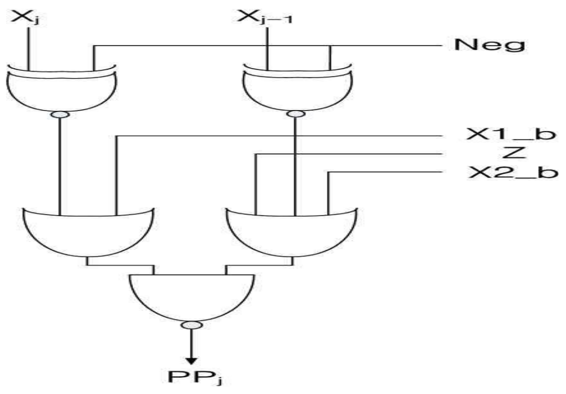


Fig.2 Booth Decoder

Fig 1.2 shows the generated partial products and sign extension scheme of the 8-bit modified Booth multiplier. The partial products generated by the modified Booth algorithm are added in parallel using the Wallace tree until the last two rows are remained. The final multiplication results are generated by adding the last two rows. The carry propagation adder is usually used in this step.

TABLE I

Truth table for MBE Scheme

Y_{i+1}	Y_i	Y_{i-1}	Value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

IV. PROPOSED SYSTEM

Implementation of multiplier comprises three steps: generation of partial products, partial products reduction tree, and finally, a vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. In this brief, approximation is applied in reduction tree stage.

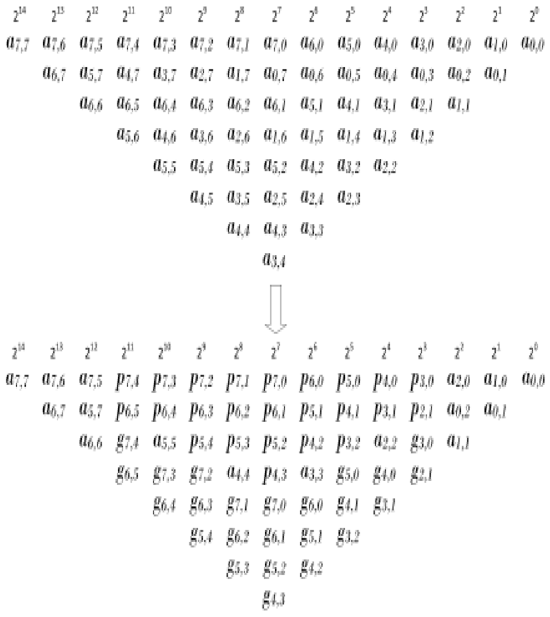

2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
$a_{7,7}$ $a_{7,6}$ $a_{7,5}$ $a_{7,4}$ $a_{7,3}$ $a_{7,2}$ $a_{7,1}$ $a_{7,0}$ $a_{6,0}$ $a_{5,0}$ $a_{4,0}$ $a_{3,0}$ $a_{2,0}$ $a_{1,0}$ $a_{0,0}$
$a_{6,7}$ $a_{5,7}$ $a_{4,7}$ $a_{3,7}$ $a_{2,7}$ $a_{1,7}$ $a_{0,7}$ $a_{0,6}$ $a_{0,5}$ $a_{0,4}$ $a_{0,3}$ $a_{0,2}$ $a_{0,1}$
$a_{6,6}$ $a_{6,5}$ $a_{6,4}$ $a_{6,3}$ $a_{6,2}$ $a_{6,1}$ $a_{5,1}$ $a_{4,1}$ $a_{3,1}$ $a_{2,1}$ $a_{1,1}$
$a_{5,6}$ $a_{4,6}$ $a_{3,6}$ $a_{2,6}$ $a_{1,6}$ $a_{1,5}$ $a_{1,4}$ $a_{1,3}$ $a_{1,2}$
$a_{5,5}$ $a_{5,4}$ $a_{5,3}$ $a_{5,2}$ $a_{4,2}$ $a_{3,2}$ $a_{2,2}$
$a_{4,5}$ $a_{3,5}$ $a_{2,5}$ $a_{2,4}$ $a_{2,3}$
$a_{4,4}$ $a_{4,3}$ $a_{3,3}$
$a_{3,4}$
2^{14} 2^{13} 2^{12} 2^{11} 2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
$a_{7,7}$ $a_{7,6}$ $a_{7,5}$ $p_{7,4}$ $p_{7,3}$ $p_{7,2}$ $p_{7,1}$ $p_{7,0}$ $p_{6,0}$ $p_{5,0}$ $p_{4,0}$ $p_{3,0}$ $a_{2,0}$ $a_{1,0}$ $a_{0,0}$
$a_{6,7}$ $a_{5,7}$ $p_{6,5}$ $p_{6,4}$ $p_{6,3}$ $p_{6,2}$ $p_{6,1}$ $p_{5,1}$ $p_{4,1}$ $p_{3,1}$ $p_{2,1}$ $a_{0,2}$ $a_{0,1}$
$a_{6,6}$ $g_{7,4}$ $a_{5,5}$ $p_{5,4}$ $p_{5,3}$ $p_{5,2}$ $p_{4,2}$ $p_{3,2}$ $a_{2,2}$ $g_{3,0}$ $a_{1,1}$
$g_{6,5}$ $g_{7,3}$ $g_{7,2}$ $a_{4,4}$ $p_{4,3}$ $a_{3,3}$ $g_{5,0}$ $g_{4,0}$ $g_{2,1}$
$g_{6,4}$ $g_{6,3}$ $g_{7,1}$ $g_{7,0}$ $g_{6,0}$ $g_{4,1}$ $g_{3,1}$
$g_{5,4}$ $g_{6,2}$ $g_{6,1}$ $g_{5,1}$ $g_{3,2}$
$g_{5,3}$ $g_{5,2}$ $g_{4,2}$
$g_{4,3}$

Fig. 3: Transformation of generated partial products into altered partial products

TABLE II

PROBABILITY STATISTICS OF *Generate* SIGNALS

m	Probability of the *generate* elements being				P_{err}
	all zero	one 1	two 1's	three 1's and more	
2	0.8789	0.1172	0.0039	-	0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

A8-bit unsigned multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsigned input operands α = $\sum_{m=0}^{7} \propto m2^m$ and β = $\sum_{n=0}^{7} \beta n\, 2^n$. The partial product a m , n = α m · β n in Fig. 3is the result of AND operation between the bits of α m and β n . From statistical point of view, the partial product a m , n has a probability of 1 / 4 of being 1. In the columns containing more than three partial products, the partial products a m , n and a n , m are combined to form propogate and generate signals as given in (1). The resulting propogate and generate signals form altered partial products p m , n and g m , n . From column 3 with weight 2^3 to column 11 with weight 2^{11} , the partial products a m , n and a n , m are replaced by altered partial products p m , n and g m , n . The original and transformed partial product matrices are shown in Fig 3

$$p_{m,n} = a_{m,n} + a_{n,m}$$
$$g_{m,n} = a_{m,n} \cdot a_{n,m}. \qquad (1)$$

The probability of the altered partial product g m , n being one is 1 / 16, which is significantly lower than 1 / 4of a m , n . The probability of altered partial product p m, n being one is 1 / 16 + 3 / 16 + 3 / 16 = 7 / 16, which is higher than g m , n . These factors are considered, while applying approximation to the altered partial product matrix.

A. Approximation of Altered Partial Products g m , n

The accumulation of generate signals is done columnwise. As each element has a probability of 1/ 16 of being one, two elements being 1 in the same column even decreases. For example, in a column with 4 generate signals, probability of all numbers being 0 is $(1 - pr)^4$, only one element being one is $4\,pr\,(1 - pr)^3$, the probability of two elements being one in the column is $6\,pr^2\,(1 - pr)^2$, three ones is $4\,pr^3\,(1 - pr)$ and probability of all elements being 1 is pr^4 ,where pr is 1 / 16. The probability statistics for a number of generate elements m in each column are given in Table II. BasedonTable II,using OR gate in the accumulation of column wise generate elements in the altered partial product matrix provides exact result in most of the cases. The probability of error (P err) while using OR gate for reduction of generate signals in each column is also listed in Table II. As canbe seen, the probability of misprediction is very low. As the number of generate signals increases, the error probability increases linearly. However, the value of error also rises. To prevent this, the maximum number of generate signals to be grouped by OR gate is kept at 4. For a column having m generate signals, OR gates are used.

The proposed approximate technique can be applied to signed multiplication including Booth multipliers as well, except it is not applied to sign extension bit

TABLE III

TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0 ✓	0 ✓	0
0	1	0	1	0 ✓	1 ✓	0
1	0	0	1	0 ✓	1 ✓	0
1	1	1	0	1 ✓	1 ✗	1

TABLE IV

TRUTH TABLE OF APPROXIMATE FULL ADDER

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0	0 ✓	0 ✓	0
0	0	1	0	1	0 ✓	1 ✓	0
0	1	0	0	1	0 ✓	1 ✓	0
0	1	1	1	0	1 ✓	0 ✓	0
1	0	0	0	1	0 ✓	1 ✓	0
1	0	1	1	0	1 ✓	0 ✓	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✓	0 ✗	1

B. Approximation of Other Partial Products:

The accumulation of other partial products with probability 1 / 4 for a m , n and 7 / 16 for p m , n uses approximate circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit, error in Carry bit will contribute more by producing error difference of two in the output. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence Carry outputs are approximated only for the cases, where Sum is approximated. In adders and compressors, XOR gates tend to contribute to high area and delay. Forapproximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table II. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch

$$\begin{aligned} Sum &= x1 + x2 \\ Carry &= x1 \cdot x2. \end{aligned} \qquad (2)$$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table III

$$\begin{aligned} W &= (x1 + x2) \\ Sum &= W \oplus x3 \\ Carry &= W \cdot x3. \end{aligned} \qquad (3)$$

Two approximate 4-2 compressors produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback. In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases.

TABLE V

TRUTH TABLE OF APPROXIMATE 4-2 COMPRESSOR

Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✓	0 ✓	0
0	0	0	1	0 ✓	1 ✓	0
0	0	1	0	0 ✓	1 ✓	0
0	0	1	1	1 ✓	0 ✓	0
0	1	0	0	0 ✓	1 ✓	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✓	1 ✓	0
1	0	0	0	0 ✓	1 ✓	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✓	1 ✓	0
1	1	0	0	1 ✓	0 ✓	0
1	1	0	1	1 ✓	1 ✓	0
1	1	1	0	1 ✓	1 ✓	0
1	1	1	1	1 ✗	1 ✗	1

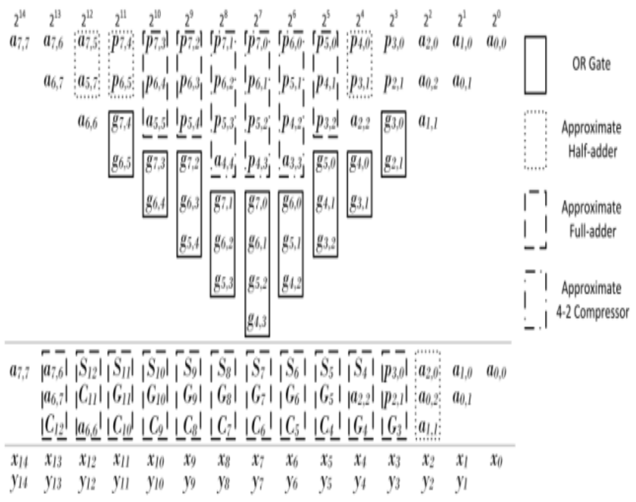


Fig. 4.Reduction of altered partial products.

This property is taken to eliminate one of the three output bits in 4-2 compressor. To maintain minimal error difference as one, the output “100" (the value of 4) for four inputs being one has to be replaced with outputs “11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate. Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit x 1 · x 2 · x 3 · x 4 is added to the Sum expression. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table V.

$$
\begin{aligned}
W1 &= x1 \cdot x2 \\
W2 &= x3 \cdot x4 \\
Sum &= (x1 \oplus x2) + (x3 \oplus x4) + W1 \cdot W2 \\
Carry &= W1 + W2.
\end{aligned} \tag{4}
$$

Fig. 4 shows the reduction of altered partial product matrix of 8 × 8 approximate multiplier. It requires two stages to produce sum and carry outputs for vector merge addition step. Four 2-input OR gates, four 3-input OR gates, and one 4-input OR gates are required for the reduction of generate signals from columns 3 to 11. The resultant signals of OR gates are labeled as G i corresponding to the column i with weight 2i . For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are required in the first stage to produce Sum and Carry signals, S i and C i corresponding to column i . The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders producing final two operands x i and y i to be fed to ripple carry adder for the final computation of the result.

C. Two Variants of Multipliers :

Two variants of multipliers are proposed. In the first case (Multi- plier1), approximation is applied in all columns of partial products of n -bit multiplier, whereas in Multiplier2, approximate circuits are used in n − 1 least significant columns.

TABLE VI
Synthesis Results Of Exact, Existing, and Proposed Approximate Multipliers

Multiplier Type	Area (μm^2)	Delay (ns)	Power (μW)	PDP (fJ)	APP ($\mu m^2 \cdot \mu W$
Exact	4859.28	0.68	1776.49	1208.01	86.32
Multiplier1	2158.56	0.47	503.15	236.48	10.86
Multiplier2	3319.20	0.66	1102.03	727.34	36.57
ACM1 [5]	2871.72	0.4	435.31	174.12	12.50
ACM2 [5]	3782.16	0.63	1250.70	787.94	47.30
SSM [6]	3953.88	0.69	1225.29	845.45	48.44
PPP [7]	4547.52	0.64	1570.79	1005.31	71.43
UDM [8]	3938.00	0.67	1318.51	883.40	51.92

TABLE VII
Error Metrics For 16-Bit Multiplier

Multiplier	Mean Relative Error	Normalized Error Distance
Multiplier1	7.63×10^{-2}	1.78×10^{-2}
Multiplier2	$2.44\ \times10^{-4}$	7.10×10^{-6}
ACM1 [7]	16.6	4.96×10^{-2}
ACM2 [7]	2.30×10^{-3}	6.36×10^{-6}
SSM [8]	6.34×10^{-4}	1.07×10^{-4}
PPP [9]	8.98×10^{-4}	4.58×10^{-5}
UDM [10]	3.32×10^{-2}	1.39×10^{-2}

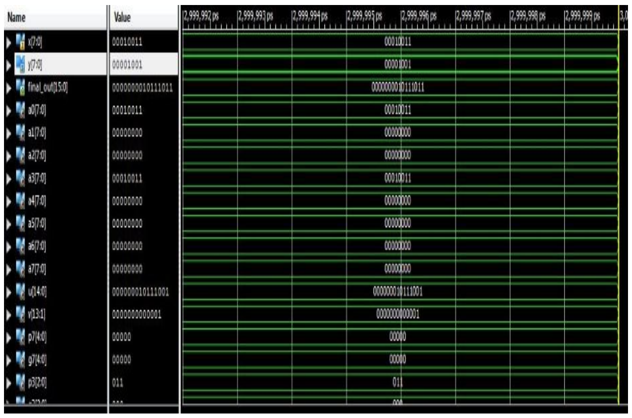
V. RESULTS

The Verilog HDL Modules have successfully simulate, verified and synthesized using Xilinxise13.2.

In this my project, to propose efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products.

Proposed Result:

Simulation:

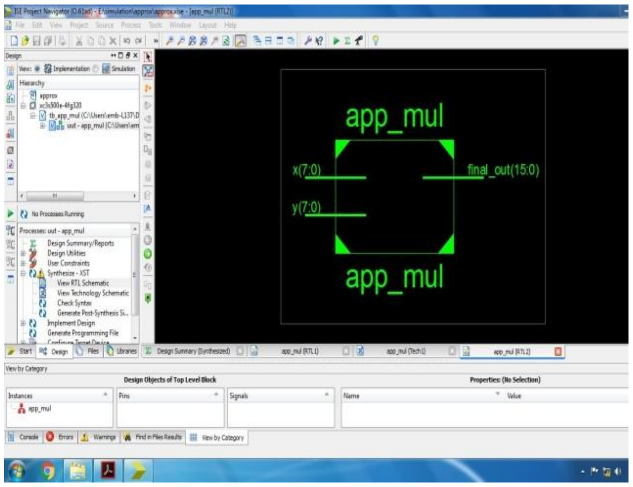


```
Name            Value
x[7:0]          00010011
y[7:0]          00001001
final_out[15:0] 0000000010111011
a0[7:0]         00010011
a1[7:0]         00000000
a2[7:0]         00000000
a3[7:0]         00010011
a4[7:0]         00000000
a5[7:0]         00000000
a6[7:0]         00000000
a7[7:0]         00000000
u[14:0]         000000010111001
v[13:1]         0000000000001
p7[4:0]         00000
g7[4:0]         00000
p3[2:0]         011
```

SYNTHESIS RESULTS:

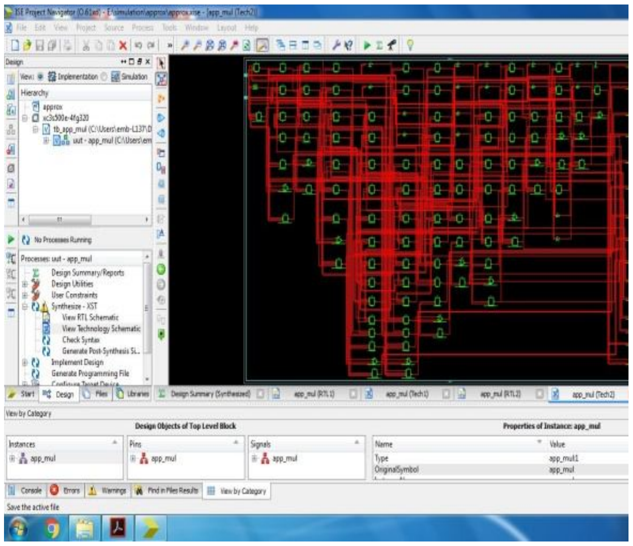
RTL SCHEMATIC:

This simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that the design is functioning as intended.



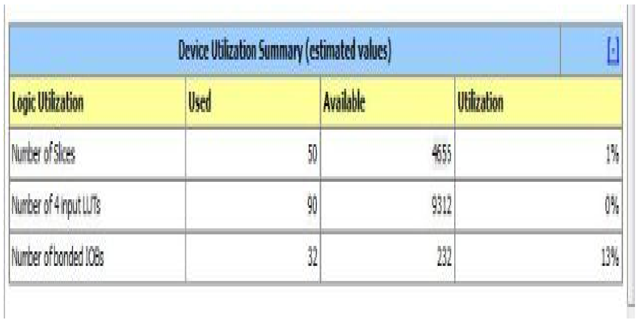
TECHNOLOGY SCHEMATIC:

Most manufacturing processes are fairly tightly coupled to the item they are manufacturing. An assembly line built to produce Buicks.



DESIGN SUMMARY:

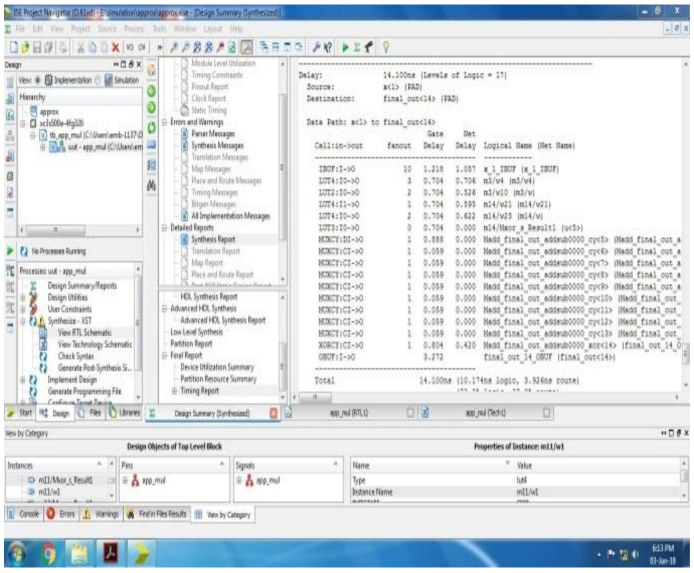
Now the design must be loaded on the FPGA. But the design must be converted to a format so that the FPGA can accept it. BITGEN program deals with the conversion. The routed NCD file is then given to the BITGEN program to generate a bit stream (a .BIT file) which can be used to configure the target FPGA device. This can be done using a cable. Selection of cable depends on the design.



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	50	4655	1%
Number of 4 input LUTs	90	9312	0%
Number of bonded IOBs	32	232	13%

TIMING SUMMARY:

This can be done after MAP or PAR processes Post MAP timing report lists signal path delays of the design derived from the design logic. Post Place and Route timing report.



```
Delay:               14.100ns (Levels of Logic = 17)
  Source:            x<1> (PAD)
  Destination:       final_out<14> (PAD)

  Data Path: x<1> to final_out<14>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            10   1.218   1.057  x_1_IBUF (x_1_IBUF)
     LUT4:I0->O            3   0.704   0.706  m3/w4 (m3/w4)
     LUT2:I0->O            2   0.704   0.526  m3/w10 (m3/w)
     LUT4:I1->O            1   0.704   0.595  m14/w21 (m14/w21)
     LUT4:I0->O            2   0.704   0.622  m14/w23 (m14/w)
     LUT3:I0->O            0   0.704   0.000  m14/Mxor_x_Result1 (w<5>)
     MUXCY:DI->O           1   0.888   0.000  Madd_final_out_addsub0000_cy<5> (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<6> (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<7> (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<8> (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<9> (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<10> (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<11> (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<12> (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<13> (Madd_final_out_
     XORCY:CI->O           1   0.804   0.420  Madd_final_out_addsub0000_xor<14> (final_out_14_O
     OBUF:I->O                 3.272          final_out_14_OBUF (final_out<14>)
    ----------------------------------------
    Total                     14.100ns (10.174ns logic, 3.926ns route)
```

VI. APPLICATION—IMAGE PROCESSING:

Geometric mean filter is widely used in image processing to reduce Gaussian noise. The geometric mean filter is better at preserving edge features than the arithmetic mean filter. Exact and approximate 16-bit multipliers are used to perform multiplication between 16-bit pixels.

VII. CONCLUSION

In this brief, to propose efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products. Designed approximate multiplier outperformsmore than existing designs. They are also found to have better precision when compared to existing approximate multiplier designs. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.

REFERENCES

[1] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[2] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., Nov. 1998, pp. 1178–1182.

[3] K.-J. Cho, K.-C.Lee, J.-G.Chung, and K. K. Parhi, "Design of low-error fixed-width modified booth multiplier," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[4] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[5] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[6] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[7] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[8] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," J. Low Power Electron., vol. 7, no. 4, pp. 490–501, 2011.

[9] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in Proc. IEEE 31st Int. Conf. Comput. Design, Sep. 2013, pp. 33–38.

[10] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.

[11] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Oct. 2011, pp. 667–673.

[12] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Comput., vol. 63, no. 9, pp. 1760–1771, Sep. 2013.

[13] S. Suman et al., "Image enhancement using geometric mean filter and gamma correction for WCE iamges," in Proc. 21st Int. Conf., Neural Inf. Process. (ICONIP), 2014, pp. 276–283.