International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

VHDL Implementation of Energy Efficient Multiplier using Bit Significance Driven Logic Compression

N.Naga Lakshmi [(1)]
[1]M.Tech, Embedded Systems ,
Gudlavalleru Engineering College,
Gudlavalleru, India.

V. Vittal Reddy[(2)]
[2]Associate Professor,
Dept of ECE , Gudlavalleru Engineering College,
Gudlavalleru, India

***Abstract*:- As a successful paradigm for many imprecision-liberal applications, approximate arithmetic is one of the newly emerging methods. By easing precise requirements, it can deliver significant decreases in loop complexity, delay and energy reduction. In this article, using a meaning-driven logic compression (SDLC) method, we propose a novel energy-efficient approximate multiplier design. Based on their progressive bit significance, an algorithmic and configurable loss compression of the partial product rows is fundamental to this strategy. This is preceded by switching the restore subsequent conditions of the item to decrease the amount of item lines. As such, the multiplier's efficiency is drastically decreased in excess of the number of logic cells and the duration of critical routes. The proposed estimated multiplier built in Verilog HDL and compounded in Xilinx ISE 14.5 using ISIM simulator.**

Index Terms- Approximate Arithmetic's, Significance Driven Logic Compression (SDLC), Logic Clustering

I.INTRODUCTION

For evolving apps, there is an ongoing requirement for greater computing efficiency at small energy costs. Improvements from production procedures it is unlikely that alone, such as technology nodes or multi-core system-on-chip, will be ready for this assignment. Therefore, in attempt to achieve reductions in conversion energy, there is a true need to develop disruptive growth techniques. Approximate development of computer applications is a successful strategy to this end [1]–[3].

Approximate computing's is the fundamental assumption is to substitute the traditional complicated and energy-wasteful information handling blocks with decreased logic numbers by low-complex blocks. As a consequence, the price of imprecision added to the stored information decreases the efficient processor region and energy consumption. Research has shown that the majority of modern applications such as digital signal processing, Computer vision, robotics, multimedia and data analytics are somewhat tolerant of this inaccuracy [4]. This can be used as a chance to develop energy-efficient devices for application-specific devices for present and future generations.

Approximate arithmetic,such as exact adders and multipliers, can be used in many of these apps as a way of lowering energy demands, increasing velocity, minimizing costs and enhancing efficiency. It was mainly active in computer applications using fixed-point procedures and flying level operations [5],[6].

For two main purposes, multipliers are critical arithmetic units in contemporary apps. First, they are defined by complicated logic design, being one of the most demanding systems in contemporary microprocessors for data processing. Second, compute-intensive apps typically perform a big amount of multiplication activities in order to calculate results. These variables have spurred near scrutiny in estimated multiplier design research, as changes in a multiplier's power and speed are anticipated to have a significant effect on general device performance trading [7].

Over the years, a number of calculation techniques have been proposed. These techniques are meant to reduce the trouble in computing and energy consumption for the systems and structures [8]. Approximations can be introduced at any development stage, from circuit [9] via logic [10] and architecture [11] to language programming [13] and Algorithms [15]. Approximations can also be applied at any architecture stage.

Popular design techniques include linear approximations, timing (VOS and over clocking) and systems lay outing methodology in projected system designs..N^2 item conditions are generated by a typical (NxN) precise multiplier, which are then collected as a final product of 2N. The precision of this item is mainly dependent on the importance of pieces; it is probable that maintaining parts of higher importance will produce an result similar to the precise item than that of pieces of lower importance.

This can be utilized to use bit-significance to gradually compress the conditions of the item. The goal is to accomplish significant energy gains at small precision failure. We create the following main donations, inspired by our prior work [16]:

1)We suggest a new energy-efficient estimated multiplier layout strategy using bit-significance-driven memory encoding.

2) at the heart of our strategy is a design-time configurable logic clustering of item condition properly selected for a specified energy-accuracy trade-off, accompanied by restoration using their commutative characteristics to decrease the subsequent amount of item conditions.

This is, to the finest of our understanding, the first example of an estimated multiplier layout strategy based on linear logic compression using a true implementation. The remainder of the document is as follows structured. Section II presents the suggested estimated layout of multipliers. The suggested multiplier is provided in Section III. Section IV describes the test outcomes and layout tradeoffs. Finally, Section V ends the document

II. RELATED WORK

The associated study attempts are discussed in this paper in the area of estimated multiplier layout. These attempts can be mainly classified as timing or functional behavior changes. First, timing conduct can be changed using extreme scaling methods for voltage supply [4],[5].

Operating below nominal voltage enables the expense of time-induced mistakes to reduce energy consumption. These errors cannot be strictly bound and therefore additional circuits for mistake reward need to be integrated. Because timing mistakes are triggered by lengthy transport lines, which are affecting the largest part of the finished item, a modification of the traditional multiplier to cater for smooth degradation must quantify the effect of timing breach.

Second, modifications to features deal with logical extraction techniques and can be made by facilitating accurate compatibility and execution of Boolean requirements. Speed calculations to achieve energy efficiency, minimize the silicone region or optimize other device parameters, this can be achieved.

The main characteristics and limits of study attempt in the field of estimated multipliers are summarized in Table I. Cutting multiplier item conditions, for instance, enables the removal of some of the less important parts[6],[7].

Further energy reduction is accomplished as more pillars are eliminated; however, mistakes are also increasing. Another efficient method is modular re-design with low-complex combination logic [8],[9]. Energy-efficient multipliers to be constructed using tiny estimated multipliers; however, the hierarchical structure of tiny estimated sets will ultimately spread mistakes which grow with the multiplier magnitude. A software-based perforation technique in[10] was suggested by acquiring the optimized number of partial product conditions oriented on trade-offs in power-area precision. By altering the functional behavior, a range of power-and area-efficient multiplier redevelopment methods were suggested. These modifications range from the transistor-level architecture [13], [14].

Table 1: Approximate multiplier summary design approaches

Approach	Methodology	Features and Limitations
[4][5]	Aggressive voltage scaling: Lowering the supply voltage below its nominal value	Unexpected time-induced errors, which normally impact the most significant bits.
[6][7]	Truncation: eliminating partial products from the least significant columns.	As more columns are eliminated, the resulting errors are maximized.
[8][9]	Modular re-design: Large efficient multipliers using in accurate small multiplier blocks	Scalability is not simple and this method may not significantly reduce the critical path.
[10]	S/W-based perforation: approximation of the generation of the partial products.	Decreasing the depth of the accumulation tree by utilizing a tool, and also real-time needs.
[11][12]	Automated re-design: Systematically reducing the complexity of circuits.	Greedy approach depending on circuit activity profile and output significance.
[13][14]	Manual re-design: manual alteration of the functional behavior of the structure.	Disparate ideas of redesigning the multiplier extend architecture to transistor level.

Automated architecture approaches[11],[12], current layout streams to synthesize estimated circuits using models of system operation and performance boundaries. For instance, the descriptive methodology of (SALSA) in [11] starts with an precise loop RT-level characterization and mistake constraint.The strategy presents Q-function when combining estimated and initial results to determine whether the performance limitations are met. A single Boolean value is output by the Q-function. Iteratively changing the cognitive depiction of an approx. loop is the primary concept of the SALSA technique. To decrease the difficulty of the estimated loop in SALSA technique maintaining untouched Q-function production. Evolutionary circuit designs are recently demonstrates that some developed destination grid applications can be regarded as creative. However, in generating helpful applications for estimated complex circuits, the developmental architecture strategy suffers. Within the ABC instrument, a successful developmental development method has been introduced and widely assessed on linear multiplier alignment.

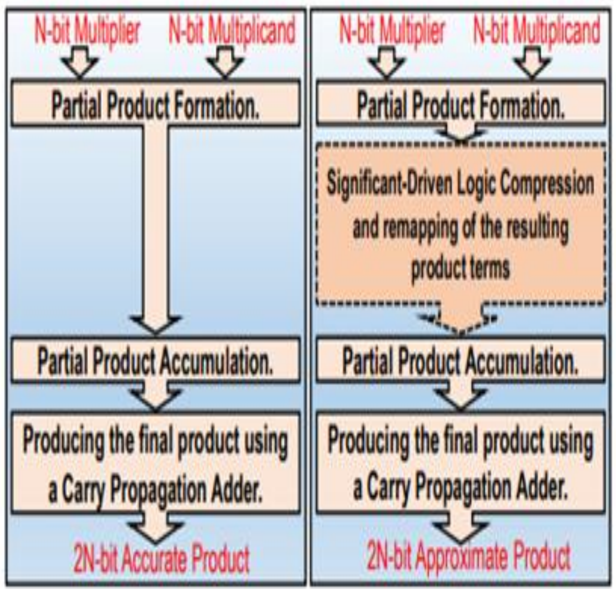
The key concept of the above research is to attain a decreased nature of logic, which is also the primary objective of our job. In this paper, we aim (for the first moment) to use meaning-driven logic compression to develop energy-efficient multipliers. Unlike associated functions, the suggested technique can be readily implemented in any architecture of multipliers without the need for a unique model. In addition, estimated mistake / energy / output depends on the configuration parameters such as magnitude of multiplier and logic compression range. Consequently, knowing the PEQ compromise enables logical encoding to be chosen so as to enhance power costs and effectiveness.

III. APPROXIMATE MULTIPLIER DESIGN USING VARIABLE LOGIC CLUSTER SIZE

Allow us to construct R two N bits binary multiplier input (NxN) multiplier without failure of generality. (A = N-12 N-1 + + a0) and multipliers (B=b N-12 N-1 + + b0). Product P can be put as follows:

$P=A.B=P_{2N-1}2^{2N-1}+-+P_0=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_i b_j 2^{i+j}$ (1)

P computation is usually split into three successive phases in simultaneous math layout: I full item creation, ii) full item retention and iii) perform propagation adder. Figure1 Displays the distinction in accuracy between the layout phases and the multiplication suggested. First, N^2 AND switches are used simultaneously to produce full value bit-matrix (PPM) N^2 item conditions. The matrix is then synthesized in a column by multiple aggregation methods, for example, carrying-save range, wallace and dada-tree, together with a carrying-propagated adder, to create the latest 2N-bit product. The effectiveness, software complexity and energy usage of multiplier building are based primarily on the greatest tree height collected. The suggested strategy reduced the amount of vertical components of the item in the PPM in order to reduce the height of the critical pillar in the tree of concentration. Following two significant measures outlined in Figure 1(b). In the first, circuit clustering is used to perform loss compression. The subsequent condensed conditions are then reworked using their commutative characteristics. In combination with the varying storage technology, these aspects of the proposed design approach are described below.


N-bit Multiplier
N-bit Multiplicand
Partial Product Formation.
Partial Product Accumulation.
Producing the final product using a Carry Propagation Adder.
2N-bit Accurate Product
N-bit Multiplier
N-bit Multiplicand
Partial Product Formation.
Significant-Driven Logic Compression and remapping of the resulting product terms
Partial Product Accumulation.
Producing the final product using a Carry Propagation Adder.
2N-bit Approximate Product

(a) (b)

Figure 1: Process diagram which shows the distinction in: (a) conventional math, and
(b) a suggested multiplication strategy.

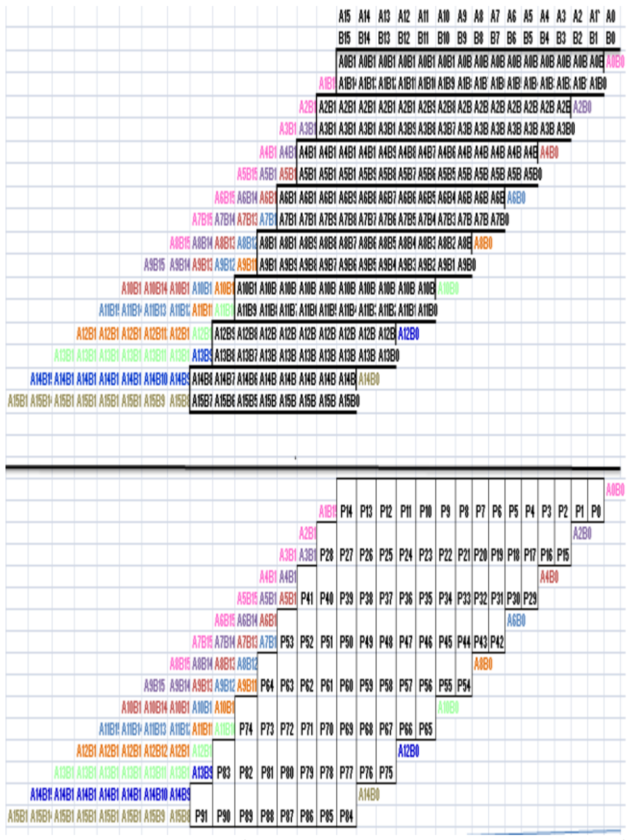


Figure 2 Stylized display of SDLC strategy Eight distinct dimensions of logic arrays are used to compress the temporary products in simultaneous multiplier architecture depending on their linear bit-significance (16x16).

A. Significance-Driven Logic Compression (SDLC):

As shown in Figure 2, this phase, comparable to standard multiplication, starts to produce all complete products using N^2 AND doors. The amount of pieces in the partial product matrix is decreased by performing loss logic compression before continuing to the storage phase. The goal is to minimize the number of rows in the PPM, resulting in low-complex hardware before accumulation proceeds. We pursue two main values as follows in order to attain loss compression.

1) Logic Clustering:

The suggested multiplier divides the conditions of the partial product using various dimensions of logic cores with significant drive. Each logic array is intended for a set of Columns comprising two parts in consecutive temporary products beginning with the least important pieces. The use of2-input OR gate can be used to compress two adjoining partial product conditions corresponding to the same row in a given word as shown in figure 2. Consider two vertically aligned bits within two successive a_ib_j partial products and a_{i-1} b_{j+1} column (i+j) in the PPM. O_{i+j}^{2-j} is a logic node signal that is displayed as:

$O_{i+j}^{2-bit}= =a_ib_j \vee a_{i-1}b_{j+1}$ (2)

For illustration purposes, the volume logic arrays (2x L) are a set of subsequent part product conditions of a length of columns in 2 rows. There are two tasks for each (2xL) storage set: I can produce 2L partial product components within two adjoining rows using 2L AND gates i.e., L pairs of vertically spaced pieces. Then, ii) use L OR doors to minimize these 2L parts by quarter. Fig. 2 Illustrate the use of four logic cluster dimensions in simultaneous multiplier (16x16). By using 30 AND logic gates, the first (2 x15) logic cluster creates 30 temporary products and derives 15-bit importance using an set of 15 OR logic gates. The second logic array (2x14) minimizes in 14 parts 28 total products. The third and fourth logic centers similarly use (2x13), (2x12), (2x11), (2x10), (2x9) and (2x8) to minimize 24,22,20,18 and 16 total products in 13,12,11,10,9 and 8 pieces respectively. By doing so, each logic array compresses within two consecutive full products a group of vertically oriented pieces depending on their progressive part meaning.

2)Logic Compression:

The temporary conditions of each item in each logic unit shall be compressed quarterly with the application of the OR door range (see Fig. 2). A reduced sample of pre-processed partial product matrix is therefore ready to be gathered by applying a helpful multiplication scheme. A double-input door is sufficient theoretical for summarizing two components,i.e.' 0'+1'='1'+'0'='0'V'1'='1'V'0'=' 1and'0'+'0'='0'V'0'='0' However, if the two outputs **are' one,' i.e.'** amount, the gap score is' 1' as the adder yields' 10' and OR produces' 1.' The arithmetic sum of two consecutive total products assigned to the row (i+j) can therefore be approximated as follows:

$$a_i b_j + a_{i-1} b_{j+1} \cong O_{i+j}^{2\text{-bit}} \quad (3)$$

If the two provisional products conditions [$a_i b_j$ and $a_{i+1} b_{j+1}$] are large, that is, an error occurs when $a_i b_j + a_{i-1} b_{j+1}$ equals $O_{i+j}^{2\text{-Bit}}$, the set of item within the PPM is reduced at the fault cost by using alternate OR compressions via logical arrays. Assuming the yield values ai and bj are equally and independently, this error is probably caused by (1/16). However, OR distortion does not affect the bigger components of the completed product.

3) Progressive Cluster Sizing

Because the main goal is to design an efficient power multiplier with a minor precision loss, the length of the L logic clusters is reduced by the PPM. The more important parts are more precisely processed, while tiny parts are compressed using the SDLC method. This allows the most important item conditions to be collected on a carry-propagation principle as in conventional multiplier. Therefore, the precision of the final product's important pieces is less influenced. The duration of the 2-bit compression logic cluster used to generate L-bit range in the simulated PPM rhythm is generally provided by:

$$L_{2\text{-bit}}(r) = N - r \quad (4)$$

While AND Gates are the exact multi fastening agent, this approach determines how much hardware is required to accumulate the parts of the product, for example, the number of compressor cells required for the column multiplication of the Wallace or Dadda column as well as the number of half and full adders gets reduced.

B. Commutative Remapping

In the message encoding stage, the number of complete object circumstances is reduced. This can be used to minimize lines in the PMM before the procurement stage. To this end, the logical compression part item conditions are changed using the Bit commuter estate, which means that parts with the same weight are gathered in the same row. For example, the height of the critical column is reduced by a half when the (2xL) logic cluster is compared to the exact accumulation tree. The condensed and rewritten bit-matrix will be shown in Figure3 after a commutational reconstruction of the bit series. Due to the reduced number of rows, the critical path distance is drastically reduced. The height of the critical pillar is further reduced as shown below.

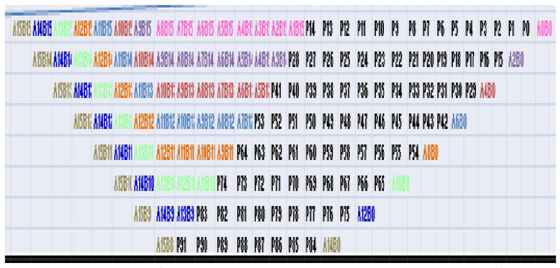


Figure3 After applying commutative restoration of the bit sequence resulting from the SDLC approach, ordered the matrix. The rectangles show the critical pillar height that is lowered by quarter relative to the precise tree of concentration.

As shown in Algorithm, the suggested strategy is scalable for any (NXN) multiplier. This algorithm creates a decreased and organized bit-matrix of the partial product, which can then be handled by any multiplication system as an acquisition tree. Line (9) shows how to compress partial product parts using logic nodes. As shown in Figure 3The principal loop (lines 6 to 17) shall be responsible for restoring a ordered bit matrix to the terms of the product.

Algorithm Generating a reduced partial product matrix M using the significant-Driven logic clusters (SDLC) approach for (NxN) multiplier,{$\forall$ N $\in\{2\sigma\epsilon$ N$^*\}$}

```
1: procedure SDLC (M,A,B)
2:Output:M[1,2,..$\frac{N}{2}$][1,2,..,2N-1] Reduced matrix
3:Inputs:A[1,2,……,N]   Multiplicand bits
4:B[1,2,……,N]   Multiplier bits
5: Initialize:$\rho \leftarrow 1$    Bit position
6:fori←1 to $\frac{N}{2}$ do     Forming rows of M
7: M[i][$\rho$]← A(0)ΛB(2i-2)   First bit in each row of M
 8: for j←1 to N-ido Forming Outputs of each logic cluster
9:M[i][j+p]←(A(j)ΛB(2i-2))V(A(j-1)ΛB(2i- 1))
10: end for
11:initialize$\delta \leftarrow 1$   Bit position
12: for k←2i-1 to N-1 do          Forming unaffected MSBs
13: M[i][$\rho$+(N-i)+$\delta$]←A(N-i)ΛB(k)
14: $\delta \leftarrow \delta + 1$
15: end for
16: $\rho \leftarrow \rho + 2$        Shift left by 2 (next row)
17: end for
18: return M    M is then treated as a reduced accumulation tree
19: end procedure
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

Approximate multiplier (16x16) Design using Fixed Logic Cluster size:

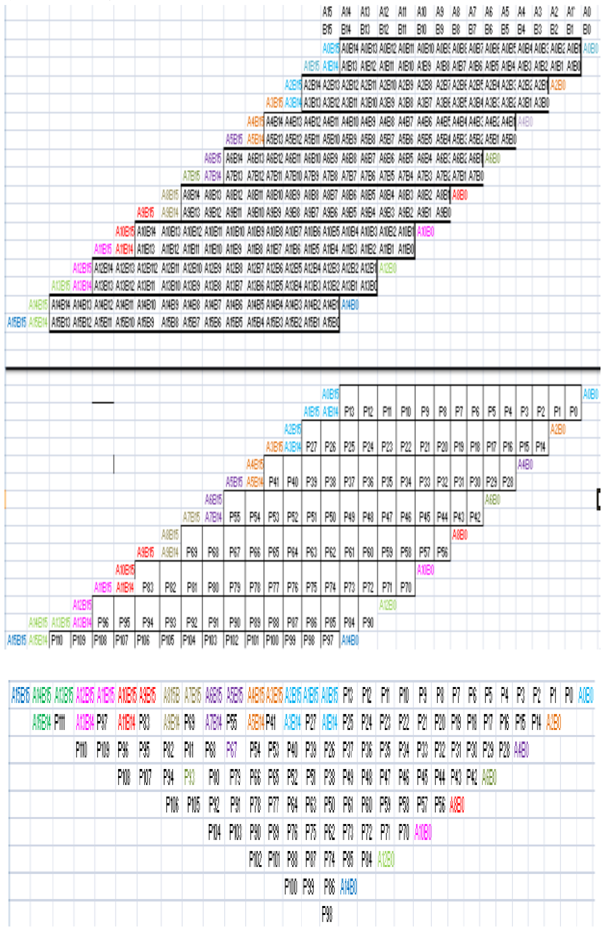
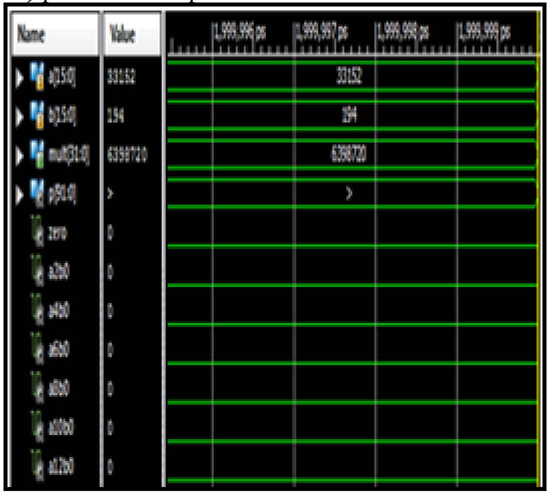


Figure 4 Stylized display of SDLC strategy Eight same dimensions of logic arrays used to compress temporary products in simultaneous multiplier architecture depending on their linear bit-significance (16x16).

The number of logic arrays is the same when moving down in the partial product matrix in set Logic cluster size strategy. By maintaining the constant size of the logic cluster the partial products are reduced more compared to the size of the variable logic cluster. As the temporary products are decreased, the cost of the hardware and the consumption of energy are decreased. The more important parts are handled with increasingly greater accuracy, while the SDLC method compresses parts with reduced importance. This makes it possible to accumulate the most important consumer conditions on a carry-propagation principle as in the case of The normal multiplier. The precision of the important components of the finished item is therefore less compromised. This approach determines The hardware complexity of partial product accumulation such as the number of compressor cells needed in Wallace and Dadda, as well as the amount of half and complete carry-saver additions in t when the number of bits is decreasing, despite using the same number of AND gates as the precise number of bits multiplier.

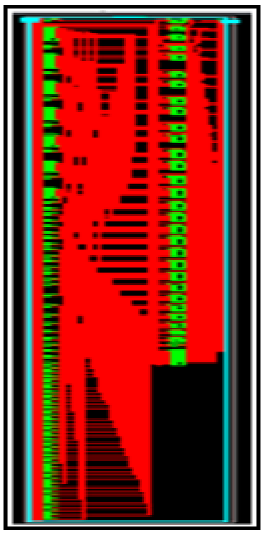
IV RESULTS

Simulation Result of SDLC approach of Eight different sizes of logic clusters used to compress partial products in (16x16) parallel multiplier architecture

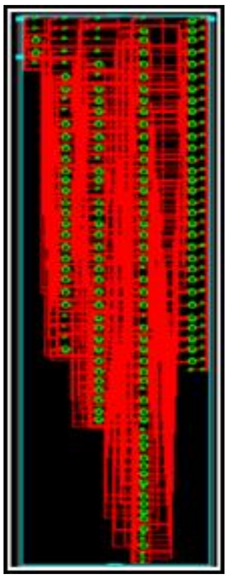


```
Name        Value     1,999,996 ps  1,999,997 ps  1,999,998 ps  1,999,999 ps
a[15:0]     33152     33152
b[15:0]     194       194
mult[31:0]  6398720   6398720
p[9:0]      >         >
zero        0
a2b0        0
a4b0        0
a6b0        0
a8b0        0
a10b0       0
a12b0       0
```

RTL SCHEMATIC



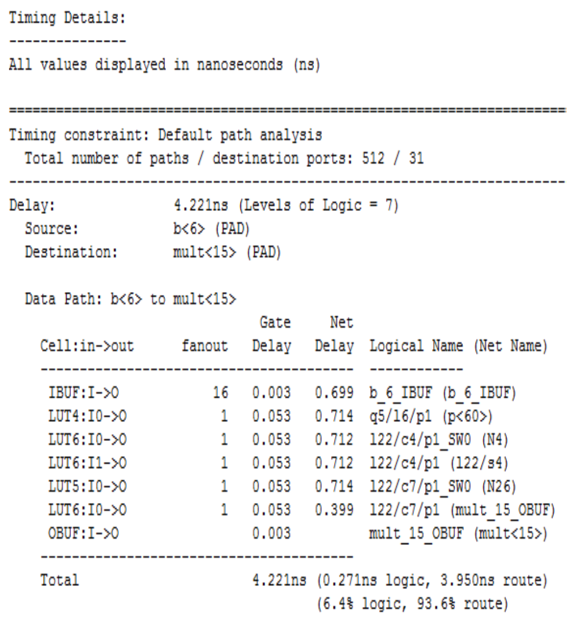
TECHNOLOGY SCHEMATIC:



DESIGN SUMMARY

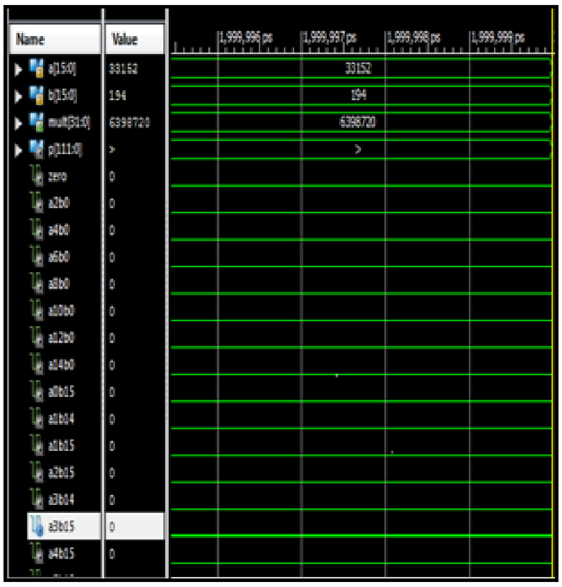
Logic utilization	Used	Available	Utilization
Number of 4 input LUTS	179	1920	9%
Number of occupied slices	91	960	9%
Number of slices containing only related logic	91	91	100%
Number of slices containing unrelated logic	0	91	0%
Total number of 4 input LUTs	179	1920	9%
Number of bonded IOBs	64	66	96%
Average Fanout of Non-clock Nets	3.72		

TIMING REPORT



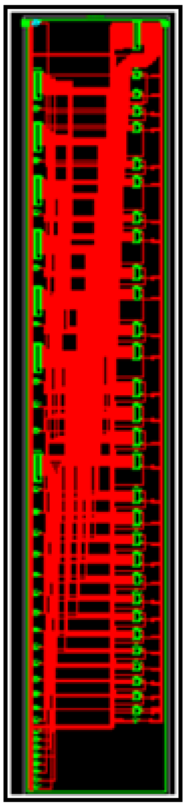
```
Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 512 / 31
-------------------------------------------------------------------------
Delay:               4.221ns (Levels of Logic = 7)
  Source:            b<6> (PAD)
  Destination:       mult<15> (PAD)

  Data Path: b<6> to mult<15>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            16   0.003   0.699  b_6_IBUF (b_6_IBUF)
     LUT4:I0->O            1   0.053   0.714  q5/16/p1 (p<60>)
     LUT6:I0->O            1   0.053   0.712  122/c4/p1_SW0 (N4)
     LUT6:I1->O            1   0.053   0.712  122/c4/p1 (122/s4)
     LUT5:I0->O            1   0.053   0.714  122/c7/p1_SW0 (N26)
     LUT6:I0->O            1   0.053   0.399  122/c7/p1 (mult_15_OBUF)
     OBUF:I->O                 0.003          mult_15_OBUF (mult<15>)
    ----------------------------------------
    Total                      4.221ns (0.271ns logic, 3.950ns route)
                                       (6.4% logic, 93.6% route)
```

Simulation Result of SDLC approach of Eight same sizes of logic clusters used to compress partial products in (16x16) parallel multiplier architecture.

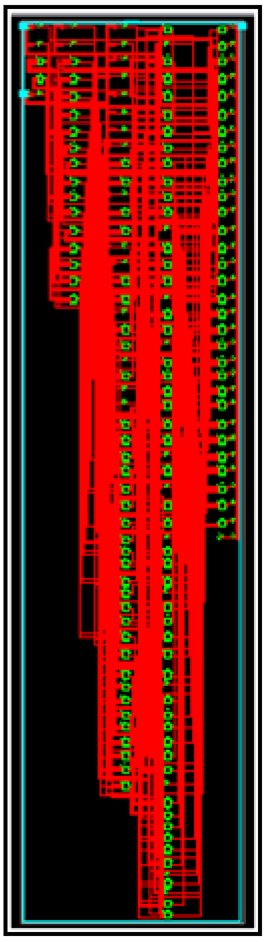


Name	Value
a[15:0]	33152
b[15:0]	194
mult[31:0]	6398720
p[111:0]	>
zero	0
a2b0	0
a4b0	0
a6b0	0
a8b0	0
a10b0	0
a12b0	0
a14b0	0
a0b15	0
a1b14	0
a1b15	0
a2b15	0
a3b14	0
a3b15	0
a4b15	0

RTLSCHEMATIC



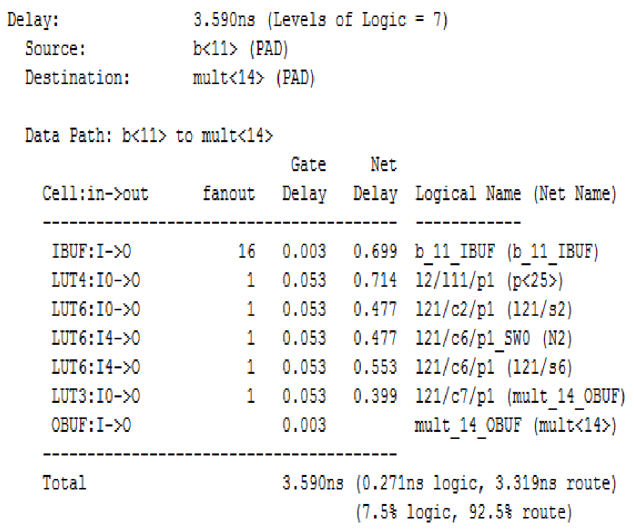
TECHNOLOGY SCHEMATIC



DESIGN SUMMARY

Logic utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTS	171	1920	8%
Number of occupied slices	86	960	8%
Number of slices containing only related logic	86	86	100%
Number of slices containing unrelated logic	0	86	0%
Total number of 4 input LUTs	171	1920	8%
Number of bonded IOBs	64	66	96%

TIMING REPORT



```
Delay:              3.590ns (Levels of Logic = 7)
  Source:           b<11> (PAD)
  Destination:      mult<14> (PAD)

  Data Path: b<11> to mult<14>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            16   0.003   0.699  b_11_IBUF (b_11_IBUF)
     LUT4:I0->O            1   0.053   0.714  12/111/p1 (p<25>)
     LUT6:I0->O            1   0.053   0.477  121/c2/p1 (121/s2)
     LUT6:I4->O            1   0.053   0.477  121/c6/p1_SW0 (N2)
     LUT6:I4->O            1   0.053   0.553  121/c6/p1 (121/s6)
     LUT3:I0->O            1   0.053   0.399  121/c7/p1 (mult_14_OBUF)
     OBUF:I->O                 0.003          mult_14_OBUF (mult<14>)
    ----------------------------------------
    Total                      3.590ns (0.271ns logic, 3.319ns route)
                                       (7.5% logic, 92.5% route)
```

COMPARISON TABLE

Parameters	Delay(ns)	Area(No.of Slices)
Multiplier with Variable Logic cluster size	4.221ns	91
Multiplier with fixed logic cluster size	3.590ns	86

V. CONCLUSION

Using Significance driven logic cluster approach (SDLC), a new estimated multiplier is intended. This layout strategy uses an algorithmic and configurable compression of losses centered on part meaning to create a decreased number of temporary conditions of item. This is then rebuilt and collected using different simultaneous computing systems. For most outputs, the 16x16 multiplier using SDLC provides near proximity to precise products. Experimental results on Xilinx show that multipliers with a fixed logic cluster size can achieve a delay of 3.590ns and an area of 86 slices compared to the 4.221ns and 91 slices of variable logic cluster size delay. The findings achieved after synthesis showed a significant reduction in run-time and silicon region. We think that with current low-power computing systems, the suggested method can be used to obtain multiple advantages with minimal error in performance.

REFERENCES

[1] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in 2013 18th IEEE European Test Symposium (ETS), pp. 1–6, May 2013.
[2] L. Sekanina, “Introduction to approximate computing: Embedded tutorial,” in 2016 IEEE 19th DDECS, pp. 1–6, April 2016.
[3] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, “A comparative evaluation of approximate multipliers,” in 2016 IEEE/ACM International Symposium on NANOARCH, pp. 191–196, July 2016.
[4] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, “Design of voltage-scalable meta-functions for approximate computing,” in 2011 DATE, pp. 1–6, March 2011.
[5] Y. Liu, T.Zhang, and K. K.Parhi, “Computation error analysis in digital signal processing systems with overscaled supply voltage,” IEEE Transactions on VLSI Systems, vol. 18, pp. 517–526, April 2010.
[6] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, “Truncated binary multipliers with variable correction and minimum mean square error,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, pp. 1312–1325, June 2010.
[7] J. E. Stine and O. M. Duverne, “Variations on truncated multiplication,” in DSD, 2003, pp. 112–119, Sept 2003.
[8] P. Kulkarni, P. Gupta, and M. Ercegovac, “Trading accuracy for power with an under designed multiplier architecture,” in 2011 24th Internatioal Conference on VLSI Design, pp. 346–351, Jan 2011.
[9] C. H. Lin and I. C. Lin, “High accuracy approximate multiplier with error correction,” in 2013 IEEE 31st ICCD, pp. 33–38, Oct 2013.
[10] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, “Design-efficient approximate multiplication circuits through partial product perforation,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, pp. 3105–3117, Oct 2016.
[11] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, “Salsa: Systematic logic synthesis of approximate circuits,” in DAC, 2012 49th ACM/EDAC/IEEE, pp. 796–801, June 2012.
[12] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “Macaco: Modeling and analysis of circuits for approximate computing,” in 2011 IEEE/ACM ICCAD, pp. 667–673, Nov 2011.
[13] K. Bhardwaj, P. S. Mane, and J. Henkel, “Power- and area-efficient approximate wallace tree multiplier for error-resilient systems,” in Fifteenth ISQED, pp. 263–269, March 2014.
[14] V. Gupta, D.Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 32, pp. 124–137, 2013.
[15] Z. Vasicek and L.Sekanina, “Evolutionary approach to approximate digital circuits design,” IEEE Transactions on Evolutionary Computation, vol. 19, pp. 432–444, June 2015.
[16] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” in 2014 DATE, pp. 1–4, March 2014.