IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 65-72
www.iosrjournals.org

High Speed Speculative Multiplier Using 3 Step Speculative Carry Save Reduction Tree

Alfiya V M, Meera Thampy
Student, Dept. of ECE, Sree Narayana Gurukulam College of Engineering, Kadayiruppu, Ernakulam, India
alfiyavm@gmail.com
Professor, Dept. of ECE, Sree Narayana Gurukulam College of Engineering, Kadayiruppu, Ernakulam, India
meera9584@gmail.com

Abstract: *This paper proposes a novel technique for integer multiplication based on speculative approach, a technique which is faster but occasionally resulting in wrong operation provided with a error correction circuit only in the rare case of error. Speculative multiplier uses speculative carry save reduction tree based on three steps: recoding partial product, partitioning partial product, speculative compression. Speculative tree uses speculative counters (m:2),with m>3 that are faster than conventional counters. Speculative adder and a correction block are also included in the circuit. In this paper we synthesis both speculative multiplier and non speculative multiplier. Comparison with conventional multiplier shows that speculative multiplier is effective when high speed is required. It is also quite effective in terms of power dissipation. With Speculative adder a high speed is achieved compared to traditional adder.*
Keywords: *multiplication,speculative multiplier*

I. INTRODUCTION

In conventional digital VLSI design, one usually assumes that a usable circuit/system should always provide definite and accurate results. The world accepts computation which generates good enough results in very short time rather than totally accurate results which takes more time. Speculative computing is an emerging design paradigm to improve the performance and energy efficiency of digital computer systems. By speculation it means that we are making some prediction or guess work. Faster circuits can be obtained by adopting a speculative approach. Speculative circuits are based on the idea of performing a faster-but occasionally wrong-operation, resorting to a multi-cycle error correction circuit only in the rare case of error. In applications such as signal processing where inexact results are tolerable, speculative can improve both speed and energy efficiency. On the other hand, when exact results are desired, appropriate error detection and recovery mechanisms are necessary to correct the errors.

Integer multiplication is a fundamental building block of digital designs, deeply affecting DSP and microprocessor performance. We are applying Speculative approach to multipliers, as well as to adders to make its operation much faster.

In this paper we propose a novel efficient architecture to build a speculative multiplier. The proposed speculative multiplier uses a novel speculative carry-save reduction tree using three steps: recoding partial products, partitioning partial products, speculative compression. We use a speculative counters and speculative adder in the architecture of speculative multiplier. The speculative tree uses (m:2) speculative counters, with m>3, that are faster than conventional counters based on full-adders and half-adders. The speculative tree is completed with a fast speculative carry-propagate adder. For exact result we use a correction circuit in speculative multiplier.

We have synthesized speculative multipliers for 16bit input, using Xilinx and implemented in Spartan 3E. We also compare speculative multiplier with non speculative multiplier. Our analysis show that speculative multipliers allow reaching a higher speed compared with standard counterparts and are also quite effective in terms of power dissipation, when a high speed operation is required. With speculative adder we obtain a high speed operation compared to traditional adder. We also see the advantages of speculative counter with traditional counter.

II. SPECULATIVE MULTIPLIER ARCHITECTURE

Let us consider two n bit binary inputs $A=a_{n-1}2^{n-1}+.....+a_0$, $B=b_{n-1}2^{n-1}+.....+b_0$. The product between A and B is

$$Y=A.B= y_{n-1}2^{n-1}+.....+y_0 = \sum_{n-1}^{i=0} \sum_{n-1}^{j=0} a_i b_j 2^{i+j} \quad (1)$$

The computation of Y requires the summation of the partial products a_ib_j according to their weights 2^{i+j}.

IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 65-72
www.iosrjournals.org

The best approach to add the partial products is using a carry save compression tree, since the delay of the carry save reduction tree is the predominant delay component of fast multipliers; we realize a speculative multiplier by using a speculative carry-save reduction tree.

Fig. 1(a) shows the partial products matrix (PPM) for a 16 x 16 multiplier. The rightmost and the leftmost columns of the PPM comprise a small number of partial products, whereas the inner columns are higher. The delay of the circuit is obviously related to the height of the PPM: the higher the matrix, the higher the multiplier delay. A speculative carry-save reduction tree could be obtained by deleting some partial products from the inner columns of the PPM. In that case the overall error probability of such approach would be unacceptably large.

Thus, we propose a novel technique to generate a speculative carry-save reduction tree based on three steps: recoding partial products, partitioning partial products, speculative compression.

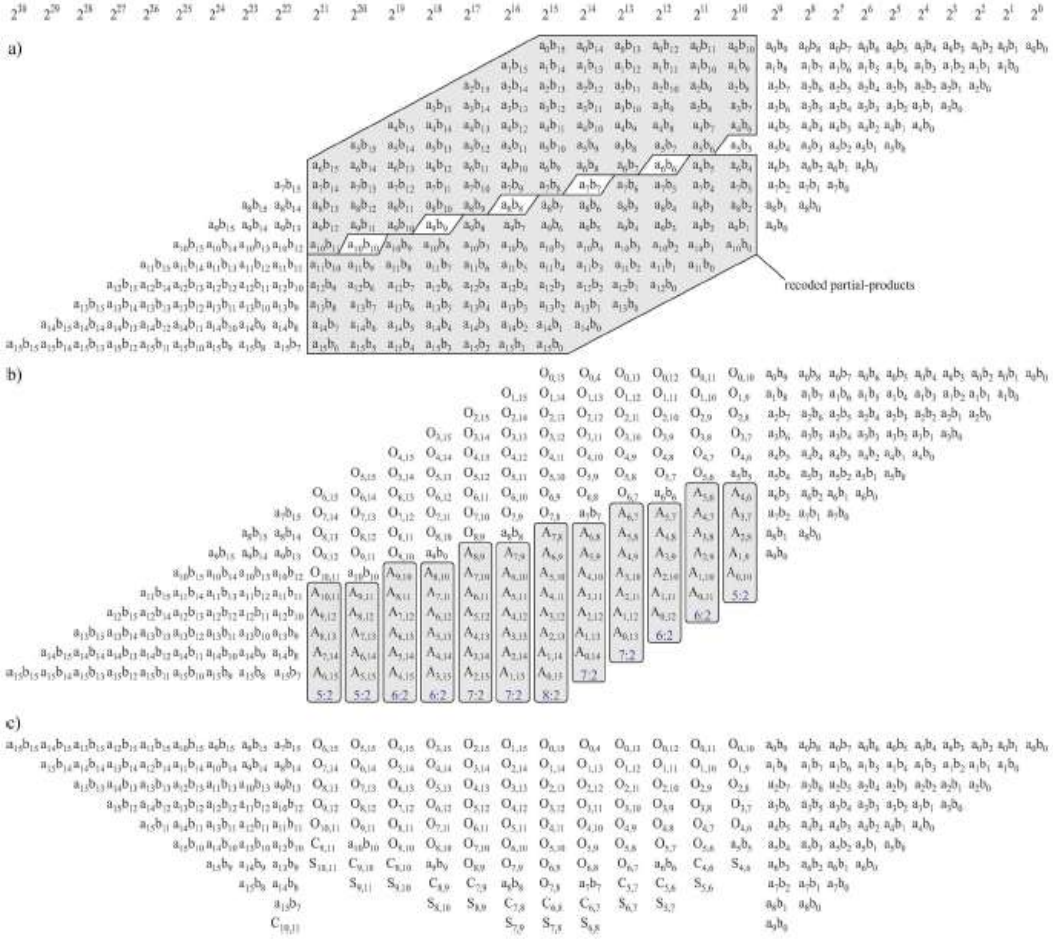

Fig. 1. 16 x 16 bit Multiplier partial products matrixes: (a) initial partial products matrix; (b) partial products matrix after recoding; (c) partial products matrix after speculative compression

A. partial products Recoding

Let us consider two partial products a_ib_j and a_jb_i of the i+j-th column of the PPM and let us introduce the following two modified partial products

$$A_{i,j} = a_ib_j \text{ AND } a_jb_i$$
$$O_{i,j} = a_ib_j \text{ OR } a_jb_i \quad (2)$$

It can easily be observed that: $A_{i,j} + O_{i,j} = a_ib_j + a_jb_i$ Thus, in the i+j-th column of the PPM, we can replace the couple of partial products a_ib_j and a_jb_i with the modified partial products $A_{i,j}$ and $O_{i,j}$. The advantage of this recoding is the introduction of low-probability terms in the PPM.

The probability of is in fact, much lower than probability of original partial products.

As it will be shown in the following, the speculative carry-save tree uses only low-probability terms, to minimize the probability of misprediction. It is worth noting that the recoding (1) does not modify the total number of partial products, but introduces an additional delay for the recoded partial products.

B. partial products Partitioning

Only low-probability terms are included in the speculative carry-save tree. We recode only the partial products belonging to the largest columns of the PPM. An example is shown in Fig. 1(b), where only the partial products of the columns 11 to 22 are recoded.

C. Speculative compression

Although the probability of $A_{i,j}$ is reduced with respect to original partial products, simple deletion $A_{i,j}$ terms would introduce a very large misprediction probability. Thus, instead of deleting $A_{i,j}$ terms, we sum them in an approximate way, by using speculative counters

An (m:2) speculative counter has $m(x_{0.....}x_{m-1})$ inputs and only two outputs: sum and Carry . The speculative counter counts the number of input bits that are "1" and encode the result on C and S, assuming that no more than three inputs are high. Similarly to full-adders and half-adders, the output C has a weight doubled with respect to S so that:

$$2C+S= x_0 +x_{1.....+} x_{m-1} \text{ for } x_0 +x_{1.....+} x_{m-1} \leq 3 \quad (3)$$

For m=2 and m=3 the speculative counter yields the correct count of high input bit and hence corresponds either to a half-adder (m=2) or to a full-adder (m=3). For m>3 it is impossible to represent the sum $x_0 +x_{1.....+} x_{m-1}$ by using only C and S signals for all the possible input configurations. The speculative counter nevertheless calculates its outputs assuming that no more than three inputs are high: if this condition is not verified an error (misprediction) occurs, the multiplication result is wrong and it must be corrected in the next cycle. By using three (5:2),four (6:2), four (7:2) speculative counters and one (8:2) speculative counter applied to the recoded $A_{i,j}$ terms, we obtain the reduced PPM of Fig. 1(c) .Compared to conventional counters, speculative counters are simpler since they produces a lower number of output bits and are faster. This explains the possible improvements in multiplier performance.

D. Multiplier Architecture

The complete architecture of the proposed speculative multiplier is outlined in Fig. 2. The multiplier inputs are firstly processed by the partial products generation and recoding block. This block computes all the partial products a_ib_j and recodes those belonging to the inner columns of the PPM, generating $A_{i,j}$ and $O_{i,j}$ recoded partial products.

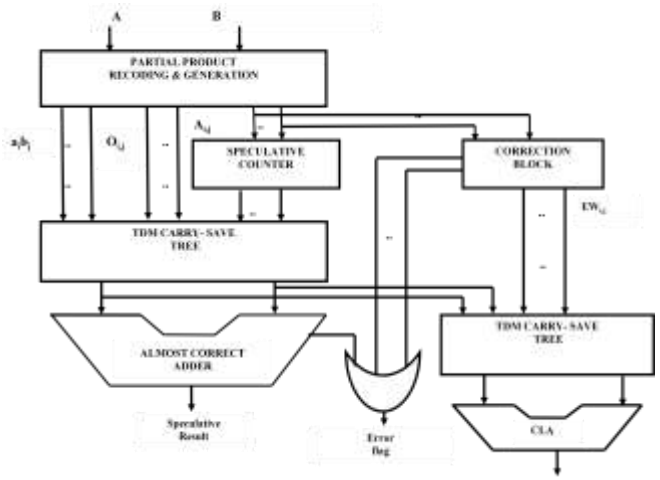

Fig. 2. Architecture of speculative multiplier

The $A_{i,j}$ terms are processed by the speculative counters obtaining the reduced $S_{i,j}$ and $C_{i,j}$ terms .The obtained $S_{i,j}$ and $C_{i,j}$, terms, the un-recoded a_ib_j and the recoded $O_{i,j}$ partial products are summed together by using a TDM carry-save tree[6]-[7]. The TDM considers the different arrival times of its inputs and tries to make proper connections to full adders so that the delay throughout each path is approximately the same. Thus, the late arriving outputs of the speculative counters are connected to the shortest delay path in the TDM. Globally, at the TDM outputs we obtain a delay profile similar to the one of a conventional multiplier. The maximum delay, however, is reduced compared to conventional multipliers, owing to the use of speculative counters.

The two outputs of the TDM tree are added by using a speculative adder [2], obtaining the speculative result Y_s.

As in any speculative functional unit, no information loss is allowed at the output of the proposed multiplier. Since each speculative compressor can introduce an error, a correction block is needed for each speculative compressor. This correction block receives the same inputs of the related speculative compressor and computes two outputs: an error flag and a suitable correction word. The error flag is high if four or more inputs of the speculative compressor are "1." The correction word is constructed so that by adding to the speculative compressor output we obtain the correct result.

As it can be observed in the right-hand part of Fig. 2. all error flags produced by each correction block are OR-ed together and OR-ed with the error flag of the speculative adder, to obtain the error flag of the speculative multiplier. In case of error, the non speculative multiplication result (Y) is computed by summing the correction words $EW_{i,j}$ with the outputs of the TDM carry-save tree included in the speculative part of the multiplier.

It is interesting to observe that in this architecture the error correction part of the speculative adder is not needed—only the error flag is required. In case of misprediction, in fact, the multiplier output Y is, which is computed independently of the speculative adder.

III. TDM

By examining the entire partial product array at once, one can construct trees for each column that sum all of the partial products in the shortest possible time. This approach is called the three-dimensional method (TDM) because it considers the arrival time as a third dimension along with rows and columns.

In the three-dimensional method, each column is summed with a vertical compressor slice (VCS).In this modified TDM, in each VCS we give inputs with 0 arrival time at its first level. While input with arrival time 1 is given in the second level and input with arrival time 2 on third level and so on. Carry generated on each level is then passed to next VCS.

Fig 3 and 4 shows VCS 9 and VCS 10 that will add 9 and 10 partial products. Wire is labelled with its arrival time. Input with 0 arrival time indicates partial products from the matrix.

In VCS 9 inputs from the partial products are given to full adders located on first level, generating carry c_0,c_1 and c_2 . Carry c_0,c_1 and c_2 are then passed to next VCS say VCS 10 which is then placed at its second level. Sum output from first level full adders of VCS 9 is then passed to second level full adders as shown in Fig 4. Generating c_3 and c_4 which is then passed to VCS10 at its second level and this process continues. By providing such a connection sum of partial product can be obtained in much short time. Since carry generated on each level of VCS is passed to next VCS delay can be greatly reduced.

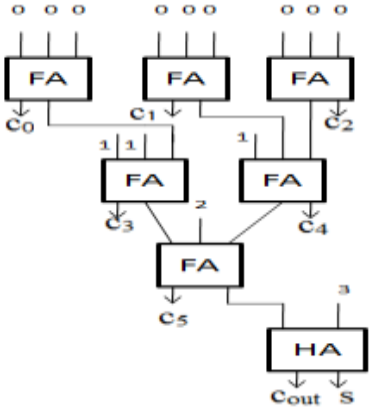

Fig 3. VCS9

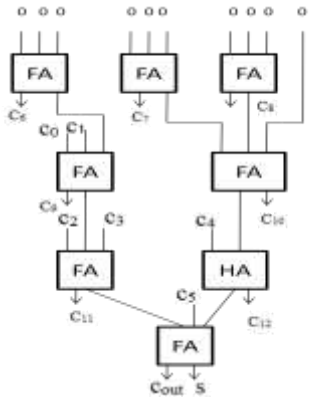

Fig. 4.VCS10

IV. SPECULATIVE COUNTER

One of the central matters in the design of speculative multiplier is allocation of speculative counter. With more speculative counters used, the compression of their partial products matrix can be sped-up. We mainly use speculative counter to reduce the height of multiplier. Recoding and speculative counters are applied only on few columns of the PPM mainly on the middle of the column. The leftmost and rightmost columns of the PPM are left unchanged.

From Fig. 1. It is clear that we need to design speculative counter of (5:2),(6:2),(7:2),(8:2). Sum output of speculative counter is computed as tree of XOR gates. Calculation of carry C posses few difficulties. Consider a binary function $f\geq_2(x_0, x_1, \ldots x_{m-1})$ that is high if at least two input signal x_i are high. The function $f\geq_2(x_0, x_1, \ldots x_{m-1})$ corresponds to C (carry) output of speculative counter.

In the case of two, three, and four inputs, the function can be easily and efficiently computed. In the case of two inputs we readily have

$$f\geq 2(x_0, x_1) = x_0 \cdot x_1 \quad (10)$$

$$f\geq_2(x_0, x_1, x_2) = x_0 \cdot x_1 + x_0 \cdot x_2 + x_1 \cdot x_2$$

Function $f\geq 2$ applied to three input mainly corresponds to carry output of a full adder.

Function $f\geq_2$ applied to four input can be computed as

$$f\geq_2(x_0, x_1, x_2, x_3, x_4) = f\geq_2(x_0, x_1, x_2) + f\geq_2(x_3, x_4) + f\geq_2(x_0 + x_1 + x_2, x_3 + x_4) \quad (11)$$

Last equation (11) can be implemented using Fig. 5.

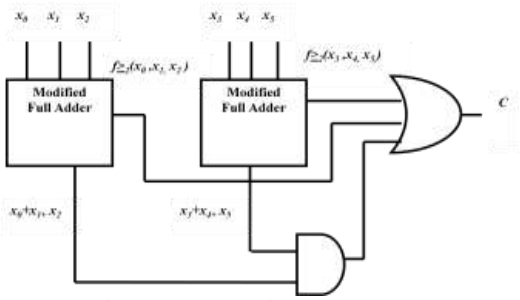

Fig. 5.Implementation of carry(c) output of (5:2) speculative compressor

Modified full adder compute carry output as $f\geq_2(x_0, x_1, x_2)$ and sum output as or between x_0, x_1 and x_2. Similarly modified half adder in Fig. 5. computes compute carry output as $f\geq_2(x_0, x_1)$ and sum output as or between x_0 and x_1.

In general, to compute $f\geq_2(x_0, x_{1,\ldots} x_{m-1})$ we subdivide the inputs signal (x_i) set in a partition of k disjoined subsets as follows:

$$(x_0, x_{1,\ldots} x_{m-1}) = (x_0, x_{1,\ldots} x_{m1-1}) \cup (x_{m1}, x_{1,\ldots} x_{m2-1}) \cup \ldots\ldots\ldots (x_{mk-1}, x_{,\ldots} x_{m-1}) \quad (12)$$

It can be shown that the function can be $f\geq_2(x_0, x_{1,\ldots} x_{m-1})$ calculated as follows

$$f\geq_2(x_0, x_{1,\ldots} x_{m-1}) = f\geq_2(x_0, x_{1,\ldots} x_{m1-1}) \cup f\geq_2(x_{m1}, x_{1,\ldots} x_{m2-1}) \cup \ldots\ldots\ldots f\geq_2(x_{mk-1}, x_{,\ldots} x_{m-1}) \quad (13)$$

For (8:2) speculative counter $f\geq_2$ is given by

$$f\geq_2(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = f\geq_2(x_0, x_1, x_2) + f\geq_2(x_3, x_4, x_5) + f\geq_2(x_6, x_7) + f\geq_2(x_0 + x_1 + x_2, x_3 + x_4 + x_5, x_6 + x_7) \quad (14)$$

Gate level implementation of $f\geq_2(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ is given in Fig. 6.

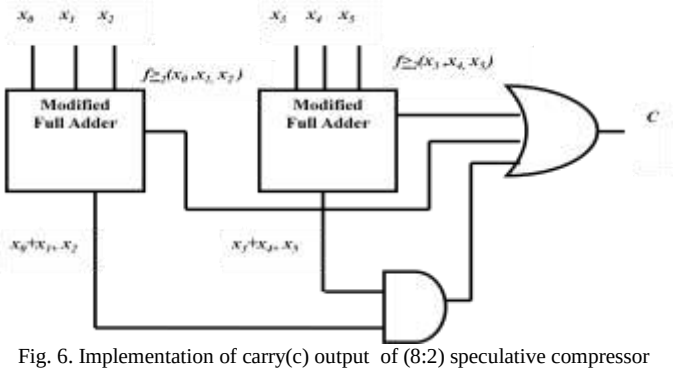

Fig. 6. Implementation of carry(c) output of (8:2) speculative compressor

To implement (7:2) compressor second modified full adder in Fig. 6. will be replaced by modified half adder.

V. CORRECTION BLOCK

The correction circuits are not critical for the overall multiplier. The error correction words EW are summed together in a two cycle subcircuit, while the error flags E are only ORed together to generate the single-cycle error output.

Error occurs in (5:2) when there is more than three inputs with high value (when five or four inputs are equal to one). The error flag can be computed as

$$E= f_{\geq 2}(\overline{x_0} ,\overline{x_1}, \overline{x_2}, \overline{x_3} ,\overline{x_4}) \quad (15)$$

Equation (15) can be computed as the carry output of a (5:2) speculative counter with inverted inputs and output as shown in Fig. 7. Please note that when E is high, the output of the speculative compressor is either 3 (when the five inputs are equal to 1) or 2 (when four inputs are equal to 1).when there is an error EW is always 3.

In (6:2) compressor to calculate E , the sum output of modified full adder and half adder is given to a half adder, the carry output of half adder c(ha) , function $f_{\geq 2}$ of both modified full adder and modified half adder is then given to full adder , whose carry will give error E signal is shown in Fig. 8. EW of (6:2) compressor depends on outputs of speculative compressor. If output is 1 then EW is either 3 or 5, if output is 3 then EW is either 3 or 2.

Similarly we can design for rest of compressor.

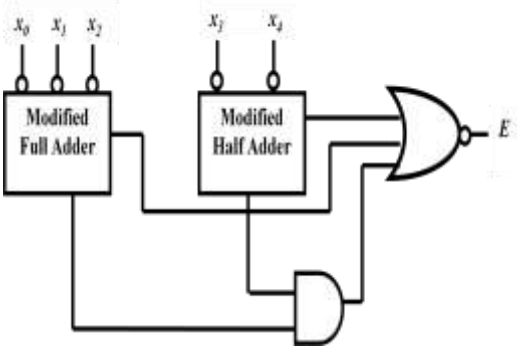

Fig . 7. Implementation of the (5:2) correction circuit.

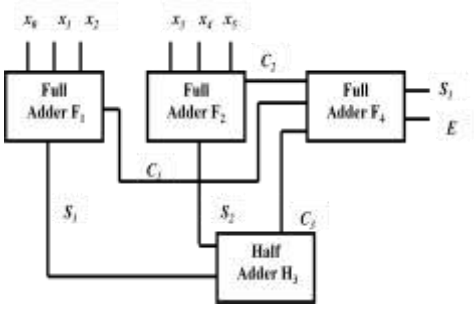

Fig . 8. Implementation of the (6:2) correction circuit.

VI. RESULT

Hardware description language that is used for design and development of different design module is verilog. Design is developed and simulated in ModelSim 6.3f. Verilog codes are synthesized in Xilinx ISE 13.3 and is implemented on Spartan 3E FPGA.

Table I compares speculative counter with non speculative counter. Conventional, non-speculative counters are obtained by composing full and half adders.

The data in the table confirms that speculative counters results in a substantial delay reduction, in addition to reduced area occupation and power dissipation. It is worth noting that speculative counters compute the sum bits just like a conventional counter, i.e., by an XOR tree, thus the gain reported in Table I is related to the other outputs of the counters.

Table I compares speculative counter with non speculative counter. Conventional, non-speculative counters are obtained by composing full and half adders.

The data in the table confirms that speculative counters results in a substantial delay reduction, in addition to reduced area. With non speculative counter there is considerable increase in delay and area.

From Table I, speculative counters introduce significant opportunities for decreasing the overall multiplication delay, being fast and having a large compressor ratio (i.e., they reduce the number of partial products effectively). This makes the following TDM carry save tree faster, because fewer bits from the speculative counters need to be added.

Table II compares speculative multiplier with non speculative multiplier. Non speculative multiplier uses non speculative counter and it uses CLA to perform final addition instead of speculative adder. Speculative multiplier with and without correction is compared. It has been found that for speculative multiplier without correction there is greater decrease in delay,

as well as in area . For speculative multiplier with correction high speed operation can be achieved. For speculative multiplier with correction there is increase in area because of the presence of correction block. The comparison between speculative multipliers and non speculative multipliers shows that speculative carry-save reduction tree is effective in improving the multiplier performance, especially in the 16 bit inputs case

TABLE I
PERFORMANCE COMPARISON OF SPECULATIVE COUNTER WITH NON SPECULATIVE COUNTER

	Speculative counter			Non speculative counter		
m	No of slices(out of 4656)	4 input LUTs(out of 9312)	Delay (ns)	No of slices(out of 4656)	4 input LUTs(out of 9312)	Delay (ns)
5	2	4	6.173	3	5	6.869
6	3	5	6.885	4	7	6.991
7	3	6	7.147	5	9	8.064
8	5	9	8.014	6	10	8.135

TABLE II
PERFORMANCE COMPARISON OF SPECULATIVE MULTIPLIER WITH NON SPECULATIVE MULTIPLIER

MULTIPLIER	No of slices(out of 4656)	4 input LUTs(out of 9312)	Delay (ns)
Non speculative multiplier	466	819	43.108
Speculative multiplier without correction	417	731	21.660
Speculative multiplier with correction	627	1105	31.578

VII. CONCLUSION

In the paper speculative multiplier for high-speed application is proposed using novel 3 step speculative carry save reduction tree which includes recoding partial products, partitioning partial products, speculative compression. The circuit recodes some of the multiplier partial products and uses a speculative compression tree to sum the partial products. A speculative adder is used to sum the final carry propagation stage. Implementation results show that 16 bit speculative multiplier is effective in high speed operation. Speculative multiplier can be used in application were speed is given more importance. Additional work may further improve the performance of speculative multipliers, by providing some optimisation on speculative carry save reduction tree

ACKNOWLEDGMENT

The authors are grateful to the reviewers. We are also grateful to Prof. Divya S for helping us understand the complexity of the algorithm. We express our gratitude to Dept. of ECE, Sree Narayana Gurukulam College for supporting us generously with their tools.

REFERENCES

[1] Alessandro Cilardo, DavideDeCaro,Nicola Petra, Member,Francesco Caserta, Nicola Mazzocca, Ettore Napoli, and Antonio Giuseppe Maria Strollo, "High Speed Speculative Multipliers Based on Speculative Carry-Save Tree", IEEE transactions on circuits and systems, vol. 61, no. 12, Dec.014
[2] K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition:A new paradigm for arithmetic circuit design," in Proc. Design, Autom., Test Eur. (DATE), Mar. 2008.
[3] K. Du, P. Varman, and K.Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Design, Autom., Test Eur. Conf. Exhib. (DATE)*, 2012.
[4] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in Proc. Design, Autom., Test Eur. Conf.Exhib. (DATE), Apr. 2009, pp. 664–669
[5] K. Du, P. Varman, and K.Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Design, Autom., Test Eur. Conf. Exhib. (DATE)*, 2012.
[6] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996.
[7] P. F. Stelling, C. U. Martel, V. G. Oklobdzija, and R. Ravi, "Optimalcircuits far parallel multipliers," IEEE Trans. Comput., vol. 47, no. 3, pp. 273–285, Mar. 1998