ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

Manual Placement and Routing in Multipliers with Trivial Operands

Siddalingesh S. Navalgund[1], Dr. Mrityunjaya V. Latte[2]

Assistant Professor, Department of ECE, S.D.M.College of Engineering and Technology, Dharwad, Karnataka, India[1]

Principal and Professor, JSS Academy of Technical Education, Bengaluru, Karnataka, India[2]

ABSTRACT: Multiplication is a frequent operation in many digital signal processing needs. Therefore, a faster multiplier improves the performance of applications. The applied input values are multiplied to obtain the result. This paper explores the occurrence of trivial operands at the input of such multipliers. The conditions for triviality for addition have been listed as zero value on input, while that for multiplication the conditions include values like zero, 1 and -1. In this paper, a multiplier is implemented for exploiting the trivial operands on an FPGA. Through a mechanism of checking for trivial operands, it is observed that significant time reductions in computations are possible. This procedure is based on Richardson's work. Further, the paper investigates the effect of manual placement and routing to improve the performance of implementations further. The RTL code is developed using the VHDL language. After checking for functional correctness, the manual procedures are used to study the impact of placement and routing on multiplier.

KEYWORDS: Trivial operands, FPGAs, manual placement, manual routing, device utilization.

I. INTRODUCTION

Arithmetic units are prevalent in majority of the applications. A microprocessor/microcontroller has it for carrying out basic computations like addition, subtraction, multiplication and division. According to Richardson[1,2], a significant number of trivial operations are performed during the execution of processor benchmarks and other numerically intensive applications. By trivial computations, Richardson meant those that can be simplified or where the result is zero, one or equal to one of the input operands. It is shown that 67% operations were involving trivial operands[2]. Hence, it may be easily seen that the unnecessary computations may be avoided, if the input operand or operands are trivial.

Number crunching and computation intensive operations such as multiplication and division of fixed-width binary numbers involve a significant amount of computation, such as adds, shifts, and combinatorial logic[3]. However, certain operands that might be presented to the operation can make much of this computation obvious, thus trivializing the operation. Attempts to exploit this phenomenon often involve such techniques as counting the leading zeroes of an operand. An eight- or sixteen-bit integer divide, for instance, would take less time to complete than a full 32-bit division.

A high-level power optimization technique includes the ability to estimate circuit power consumption quickly. Bit-level estimation techniques which take into account the glitch activity in a circuit take too long to provide power estimates. A method is followed here such that when an arithmetic operation is to be performed, the operands are concurrently sent to the arithmetic unit to perform the arithmetic operation and into an operand check mechanism which determines whether one or both of the operands is a specific instance of a trivial operand. If one of the operands is a specific instance of a trivial operand, the arithmetic operations are halted and the check mechanism rapidly outputs the result of

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

the arithmetic operation according to the trivial operand detected. Consequently, the need to perform arithmetic operations on trivial operands is avoided.
Fig. 1 shows the percentage of integer multiplication operations involving 0, 1 or -1.

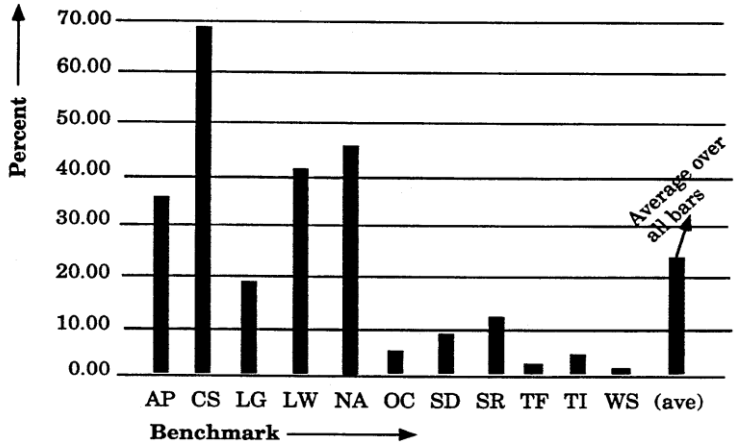

Fig. 1 Percentage of integer multiplication operations involving 0, 1 or -1[1]

The same argument is true with respect to floating-point numbers as well. Fig. 2 shows the percentage of floating-point multiplication operations involving 0, 1 or -1.

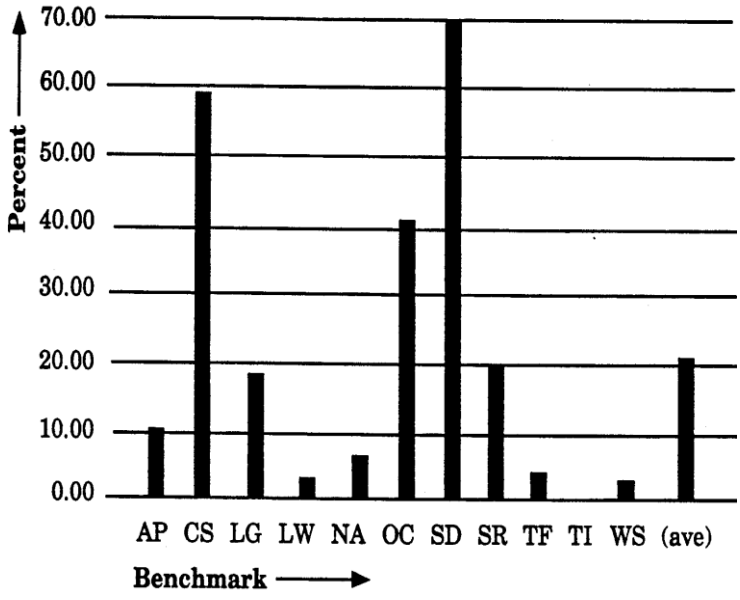

Fig. 2 Percentage of floating-point multiplication operations involving 0, 1 or -1[1].

Thus, a simple algorithm is evolved by Richardson to check for the occurrence of trivial inputs.

II. RELATED WORK

Richardson has worked on the concept of trivial operands for a multiplier[1]. There exists an US patent 5262973. Various details of operand check are mentioned in the literature. A brief summary of the performance of systems is tabulated in

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

the said work. In the year 2011, Kotresh Marali and S.S.Navalgund have investigated the design, Implementation and Analysis of Glitch effects on Multipliers and Arithmetic Unit for Trivial operands[3]. In the work titled "Improving processor performance by simplifying and bypassing trivial computations, the authors Joshua J. Yi and David J. Lilja[4], have pointed the presence of trivial operands in benchmarks suites like SPEC 95, SPEC 2000 and MediaBench. It is reported that 34.73% of all integer ADD instructions are trivial in SPEC benchmark, while only 13.18% are trivial in MediaBench.

The paper is organized as follows. In section III, the basic problem formulation is given. Section IV deals with the design and implementation of the above said multiplier for trivial inputs. Section V summarizes the experimental results obtained, while section VI presents the conclusions of the work.

III. PROBLEM FORMULATION

Given the two operands X and Y, perform the multiplication of both to obtain the product Z. check for the occurrence of trivial operands at inputs, and improve the performance of the multiplier.

i.e $Z = X * Y$……...…….…….……..…(1)

The FPGAs are to be used for the design and synthesis. It is also aimed to alter the placement of blocks, and the routing between them so that the performance difference can be measured.

Condition for Triviality:

This section uses a much stricter definition for triviality, searching for operations, so simple that they could complete in one cycle on even the simplest of machines. Table 1 displays more precisely the conditions for triviality.

It is also important to note that the definitions for add, sub, multiply and divide trivial computations in Table 1 are not limited only to integer operations, but are also equally applicable to floating-point operations. However, due to differences in how the number is represented (e.g. two's complement versus IEEE floating-point notation), how a floating-point computation is simplified and eliminated may differ as compared to its integer counterpart.

TABLE 1 CONDITIONS FOR TRIVIALITY

Operation	**Condition(s) for Triviality**
Addition	(i) X, Y = 0 (ii) X= -Y (iii) Either of two inputs are -1, 0, 1.
Multiplication	(i) X,Y =0 (ii) Either of two inputs are -1, 0, 1 (iii) X=Y=1. (iv) If either of one operands is 1
Division	(i) X=0 (ii) Y=1
Square root	(X=0 or X=1)

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

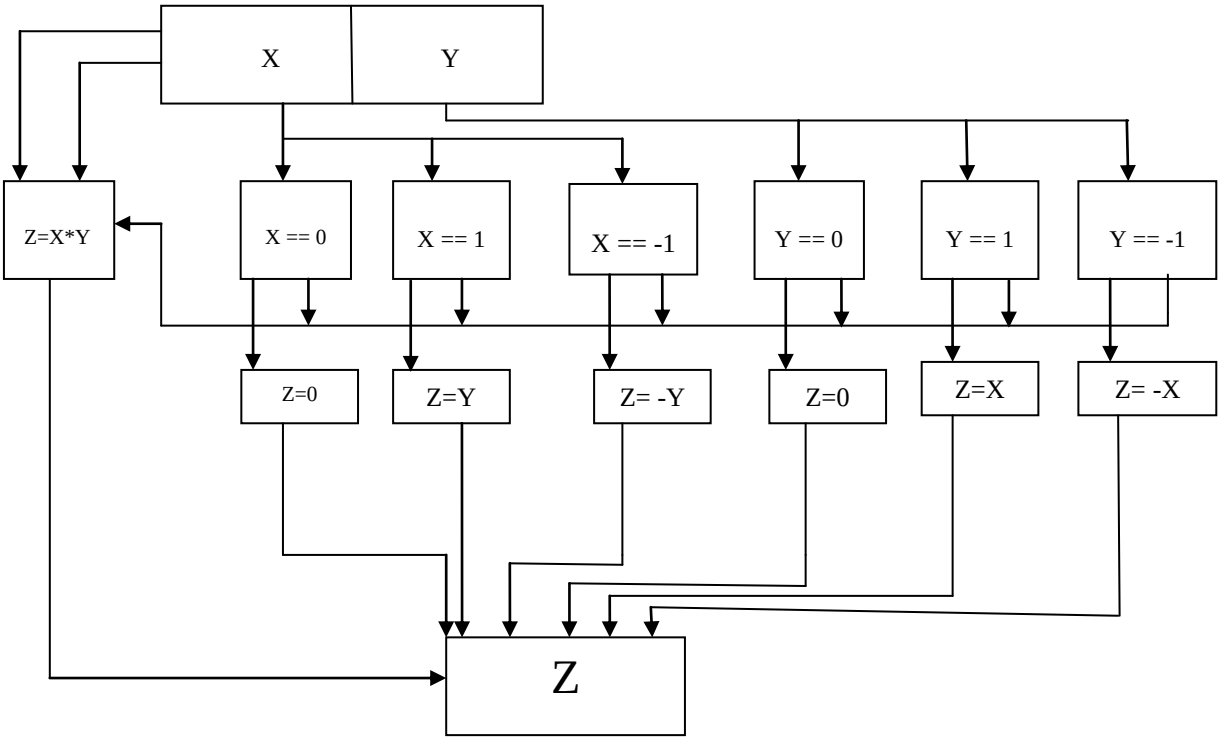

Fig. 3 Mechanism to check for trivial operands[1]

The above fig. 3 shows a mechanism to check for trivial operands. It is possible to perform the check for triviality and the computations parallel. Whenever one or both the operands are found to be trivial, then the output is obtained directly and the computation of product is halted with a signal.

This work is based on [100]. If a sufficient amount of computation were indeed trivial, some obvious changes in the style of computation could make programs run faster. Consider the following algorithm for multiplying two operands a and b to yield a result c:

OVERHEAD: if (|a|== 1.0 or b== 0.0) then
c = sign(a) ×b;
else if (|b| == 1.0 or a == 0.0) then
c = sign(b) × a;
else
goto MULTIPLY;
goto END;
MULTIPLY: c = mult(a, b);
END:

A trivial multiply—multiplication by 1.0, 0.0, or -1.0—will exit after passing through only the OVERHEAD portion of the algorithm. All other multiply operations will have the extra cost of the OVERHEAD portion added to the regular MULTIPLY portion of the algorithm. Because the conditions for triviality are so specific, a scheme for detecting them can add efficiency to generic "early-out" schemes requiring a "count-leading-zeroes" and/or a "count-leading-ones" type of operation. Provided that the OVERHEAD cost is smaller than the MULTIPLY cost, a sufficiently large ratio of trivial multiply operations to nontrivial multiply operations will justify the cost of adding the OVERHEAD.

The Trivial Arithmetic Multiplier:

The multiplication can take more cycles depending on the implementation when compared to other arithmetic operations. Detection of multiplicative operands having values of 0, 1 and the subsequent emission of the appropriate

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

result is a simple operation that should take a single cycle. A flow diagram illustration of the trivial operand optimizer is shown in Fig. 4. The operands are tested for trivial values. In the following illustration, multiplication is the arithmetic operation and 0 and 1 are specified as trivial operands. If either operand, i.e., X or Y, has the value of 0 or 1, the result, Z, is easily determined without the need to perform the complex arithmetic computation. Fig. 4 shows the sequential check for the trivial input operands.

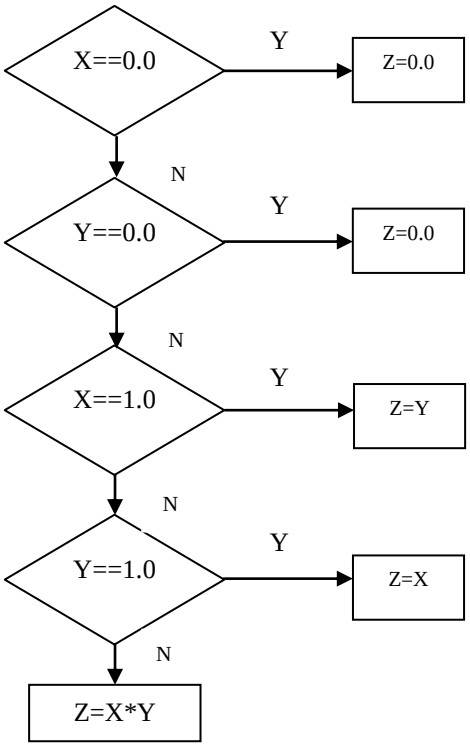

Fig. 4 Sequential check for the trivial input operands.

III. EXPERIMENTAL RESULTS

The arithmetic multiplier with trivial operands is implemented using the VHDL language [7] [10]. The functional verification is carried out by Xilinx ISE. The simulation is performed for different classes of FPGAs, and for different device families [5] [6]. The results are tabulated. Then the manual placement [8] [9] is carried out to see the changes in the routing complexity. This is followed by manual routing to affect the changes to .ncd file. The following table 2 illustrates the summary of conduction of experiment.

TABLE 2: SUMMARY OF CONDUCTION OF EXPERIMENT

Language used	VHDL
CAD Tool	Xilinx ISE 14.2
Target Device	Family : Virtex 5 Device : 5vlx110 Package : ff1153 speed grade : -3
Simulator	Xilinx ISE ISim
Status of the code	completely synthesizable

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

Table 3 and table 4 indicate Logic utilization and Logic distribution analysis of the implementation on FPGA respectively.

TABLE 3: LOGIC UTILIZATION ANALYSIS

Logic utilization	Used	Available
Number of Slice LUTs	1	69120
Number used as logic	1	69120
Number of bit slices used	1	-

TABLE 4: LOGIC DISTRIBUTION ANALYSIS

Logic distribution	Used	Available
Number of IOs	3	-
Number of bonded IOBs	3	800

The total number of paths is 2, and the number of destination ports is 1. The implementation is done using 3 levels of logic. Maximum combinational path delay is found to be 3.593 ns. Table 5 indicates the timing summary.

TABLE 5: TIMING SUMMARY

Cell	Fan out	Gate delay	Net delay	Logical name/ net name
IBUF:I->O	1	0.611	0.750	in1_IBUF (in1_IBUF)
LUT2:I0->O	1	0.080	0.213	prod1 (prod_OBUF)
OBUF:I->O	-	1.939	-	prod_OBUF (prod)

The gate delay is 2.630ns for logic implementation, whereas 0.963ns time is required for routing of nets. i.e. 73.2% of total timing is used for logic implementation, and 26.8% of total time is used for routing the nets. Total memory usage is 369248 kilobytes.

Fig. 5 indicates the Technology dependent RTL for 1-bit operands. Fig. 6. Represents the routing in FPGA Editor. The square box indicates a switch matrix. The size of ncd file is found to be 2.75 kb. Fig. 7. Indicates the implementation after manual placement and routing. The red block indicates the slice containing the logic implementation. The ncd file size is equal to 2.79kb. The complete layout with all the routing resources including local and long connecting wires is given in fig. 8.

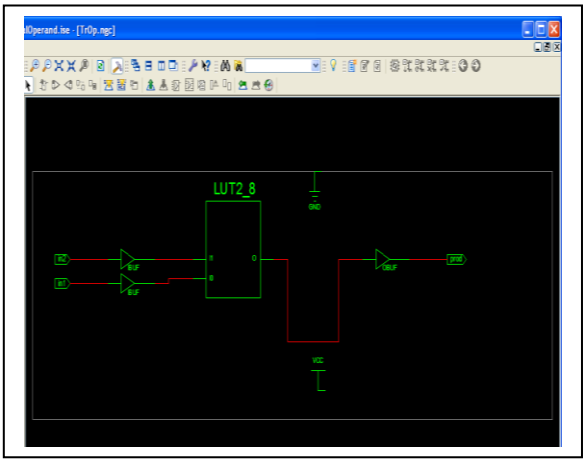

Fig. 5. Technology dependent RTL for 1-bit operands

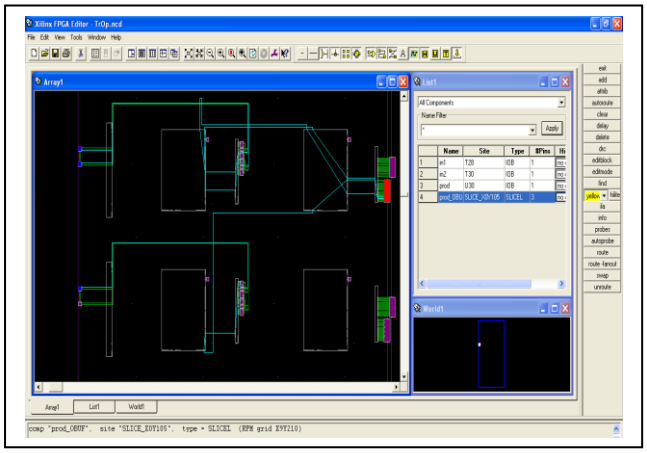

Fig. 6. Routing in FPGA Editor

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

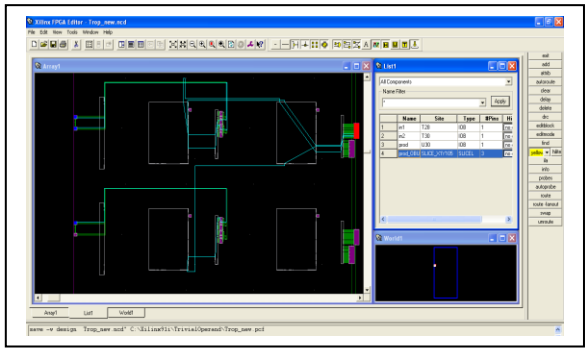

Fig. 7. After manual placement and routing

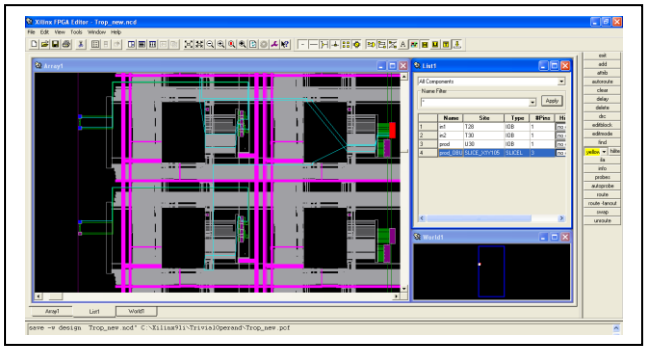

Fig. 8. Complete layout with all the routing resources

The experiment is repeated with another family of FPGAs, namely Virtex 4. The optimization goal is speed. The summary of conduction o experiment is given in table 6.

TABLE 6: SUMMARY OF CONDUCTION OF EXPERIMENT

Language used	VHDL
CAD Tool	Xilinx ISE 14.2
Target Device	Family : Virtex 4 Device : 4vlx100 Package : ff1148 speed grade : -12
Simulator	Xilinx ISE ISim
Status of the code	completely synthesizable

TABLE 7 : LOGIC UTILIZATION ANALYSIS

Logic utilization	Used	Available
Number of Slices	1	49152
Number of 4-input LUTs	1	98304

Table 7 and table 8 indicate Logic utilization and Logic distribution analysis of the implementation on FPGA respectively.

TABLE 8: LOGIC DISTRIBUTION ANALYSIS

Logic distribution	Used	Available
Number of IOs	3	-
Number of bonded IOBs	3	768

The total number of paths is 2, and the number of destination ports is 1. The implementation is done using 3 levels of logic. Maximum combinational path delay is found to be 4.976 ns. Table 9 indicates the timing summary.

TABLE 9 : TIMING SUMMARY

Cell	Fan out	Gate delay	Net delay	Logical name/ net name
IBUF:I->O	1	0.754	0.554	in1_IBUF (in1_IBUF)
LUT2:I0->O	1	0.147	0.266	prod1 (prod_OBUF)
OBUF:I->O	-	3.255	-	prod_OBUF (prod)

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

The gate delay is 4.156ns for logic implementation, whereas 0.820 ns time is required for routing of nets. i.e. 83.5% of total timing is used for logic implementation, and 16.5% of total time is used for routing the nets. Total memory usage is 342168 kilobytes.

Fig. 9. gives the placement and routing in FPGA. The ncd file size is 7.07kb. Fig. 10 indicates the implementation after manual placement and routing. The ncd file size is 7.11kb.

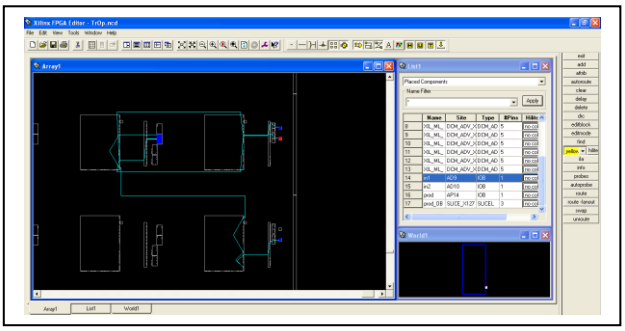

Fig. 9. Placement and routing in FPGA

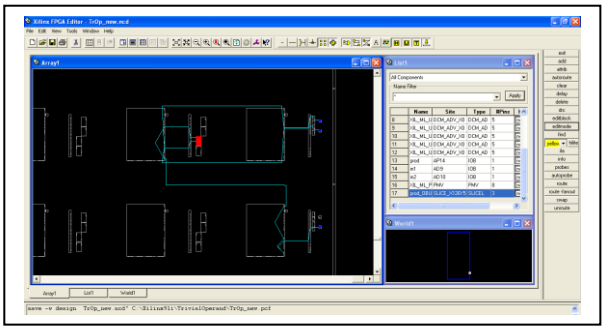

Fig. 10. Implementation after manual placement and routing

Fig. 11. Indicates an alternative routing, with the ncd file size of 7.14kb. In Fig. 12, an alternative routing option after changing the pin positions is explored. Fig. 13 shows a section of implementation showing the routing through switchbox and Ologic block.

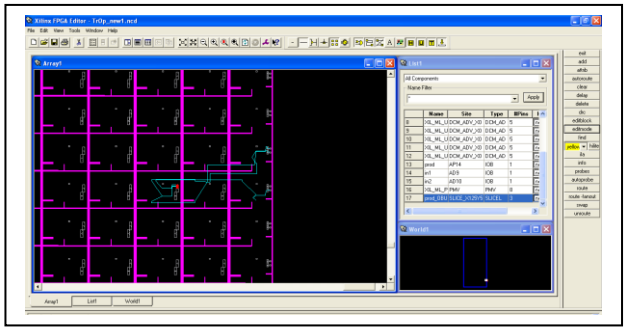

Fig. 11. An alternative routing

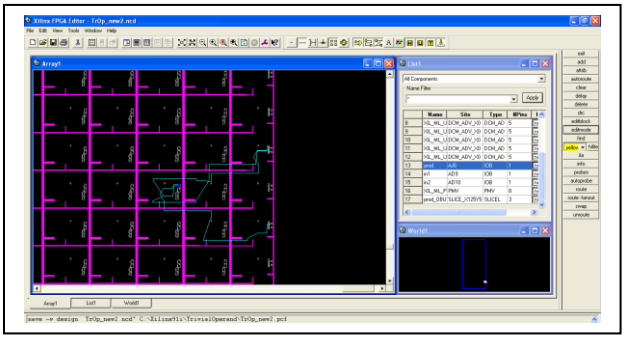

Fig. 12.An alternative routing option after changing the pin positions

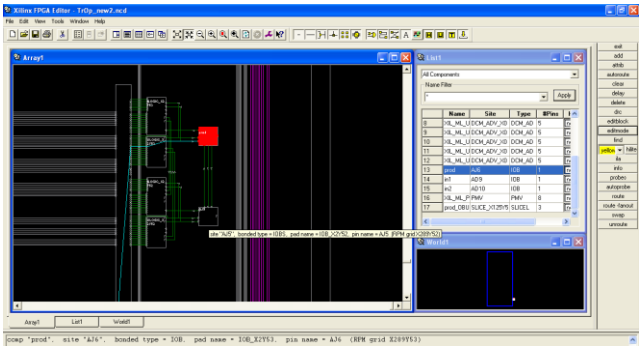

Fig. 13. A section of implementation showing the routing through switchbox and Ologic block

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

V. CONCLUSION

In this paper, a multiplier with trivial operands at the input is studied. The functional verification is followed by the assessment of design summary.

Using the trivial operands to the best of advantage, can decrease the execution time for computationally intensive implementations. It is observed that the combinational path delay has varied from 4.976ns to 3.593ns between Virtex 4 and Virtex 5 family of devices, hinting at faster implementations. Also, the manual placement and manual routing are studied for such a multiplier. These procedures result in the change in the file size of .ncd (7.07kb to 7.11kb to 7.14kb), which contains the information relevant to the final implementation on the hardware platform. Even though the sizes of .ncd files vary depending on the FPGA chosen, it is significant to note that the pin assignment/placement of blocks/routing are interdependent set of operations in high level synthesis step of VLSI design. The results indicate that the manual placement can affect the implementations for better performance if done with care. Similarly, the manual routing also needs to be done only for a few select nets. This is because, as the complexity of implementations goes on increasing, the manual routing process becomes very involved.

ACKNOWLEDGMENT

The authors thank the Management, the Principal/Director, Deans, HOD(ECE) and Staff of Sri Dharmasthala Manjunatheshwara College of Engineering and Technology, Dhavalgiri, Dharwad, Karnataka, India for encouraging us for this research work.

REFERENCES

1. Stephen Richardson, "Method and Apparatus for Optimizing Complex Arithmetic Units for Trivial Operands", US patent number 5262973, 1993.
2. Stephen E. Richardson, "Exploiting Trivial and Redundant Computation", Proceedings of the 11th Symposium on Computer Arithmetic, edited by Swartzlander, Irwin, and Jullien, IEEE Computer Society Technical Committee on VLSI, Ontario, June 29–July 2, pp. 220–227,1993.
3. Kotresh Marali, S.S.Navalgund, "Design, Implementation and Analysis of Glitch effects on Multipliers and Arithmetic Unit for Trivial operands", M.Tech thesis, SDMCET, 2010-2011.
4. Joshua J. Yi, David J. Lilja "Improving processor performance by simplifying and bypassing trivial computations".
5. Pawel P. Czapski, Andrzej Sluzek, "Power Optimization Techniques in FPGA Devices: A Combination of System- And Low-Levels", International Journal of Electrical and Electronics Engineering 1:3 2007, pp 148-154.
6. Xilinx, "Virtex-5 FPGA XtremeDSP Design considerations User Guide", UG193 (v3.4), June 1, 2010.
7. Charles H. Roth Jr. "Digital Systems Design using VHDL", Thomson Brooks/Cole, 7th reprint, 2005, ch. 7.
8. Xilinx, http://www.xilinx.com
9. FPGA Editor Guide, version 2.1i, Xilinx Corporation.
10. Bhasker J., "A VHDL Primer", revised ed. Upper saddle river, N. J., Prentice Hall, 1992

BIOGRAPHY

Siddalingesh S. Navalgund received Bachelor of Engineering Electronics and Communication Engineering from Karnataka University Dharwad in 2000 and Master of Technology in Microelectronics and Control Systems from Visvesvaraya Technological University, Belgaum in 2004. After spending two years in the industry, he joined as a Faculty in N. M. A. M. in Institute of Technology, Nitte, Karanataka. Since 2005, he is serving as faculty in the Department of Electronics and Communication Engineering, S. D. M. College of Engineering and Technology, Dharwad, Karnataka, India.

His main research interests include VLSI design, embedded systems, digital signal processing and fuzz logic. He is the member of Institution of Electronics and Telecommunication Engineers (IETE), Institute of Engineers (IE) and Indian Society for Technical Education (ISTE).

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 1, January 2016

Prof. Dr. Mrityunjaya V. Latte is currently working as the Principal of J.S.S. Academy for Technical Education(JSSATE), Bengaluru, Karnataka, India. (E-mail: mvlatte@rediffmail.com).

He received Bachelor of Engineering in Electrical Engineering from Karnataka University Dharwad, Master of Engineering in Digital Electronics from Karnataka University Dharwad, and Ph. D. from Karnataka University Dharwad in 1987, 1994, and 2004 respectively. From 2005 to 2008, he has served as full Professor in the Department of Electronics and Communication Engineering in S. D. M. College of Engineering and Technology, Dharwad. His main research interests include Digital Signal Processing and communication systems. Presently he is guiding five research scholars. He has authored more than 80 research papers in national and international journals.

Prof. Mritunjaya V. Latte is the member of Institution of Electronics and Telecommunication Engineers (IETE), Institute of Engineers (IE) and Indian Society for Technical Education (ISTE).