International Journal of Multidisciplinary Research and Development

Volume: 2, Issue: 5, 263-269
May 2015
www.allsubjectjournal.com
e-ISSN: 2349-4182
p-ISSN: 2349-5979
Impact Factor: 3.762

Jageshwar Prasad Sinha
Research Scholar, ME,
VLSI DESIGN, SSTC, SSGI (FET) Bhilai, India

Anil Kumar sahu
Asst. Prof VLSI DESIGN,
SSTC, SSGI (FET)
Bhilai, India

Correspondence:
Jageshwar Prasad Sinha
Research Scholar, ME,
VLSI DESIGN, SSTC, SSGI (FET) Bhilai, India

An Efficient Approach of Coprocessor design using RR4 Algorithm

Jageshwar Prasad Sinha, Anil Kumar sahu

Abstract
With the growth of VLSI processing in the industrial sector the design of efficient algorithms for designing compact functional circuits has led to a competition among various industries.Multiplication is basically a shift add operation. There are, however, many variations on how to do it. Some are more suitable for FPGA use than others. In the area of designing fast parallel algorithms for multiplying numbers, proposed algorithm for multiplying two *n*-bit signed binary numbers needs é 2.71 $\log_2$nù + 3 steps of single bit addition on an *n ´ n* systolic architecture which outperforms the then best VLSI implementable algorithm with *O(n)* time and $O(n^2)$ hardware. The subsequent algorithms proposed by him for multiplying numbers in ternary and redundant-radix-four (RR-4) representations require still less time with *2* é$\log_2$*n* ù *+ 2* and *é(1/2)* $\log_2$*n +1*ù steps of single digit addition, respectively. Here we have proposed a novel approach for the multiplication of two numbers in RR4 number system. The results has been evaluated in ISE environment and the performance giving satisfactory results.

Keywords: VLSI, RR4, FPGA, MULTIPLIER, COPROCESSOR

1. Introduction

Our RR-4 arithmetic coprocessor is a 8-bit RISC processor and its design consists of two separate units: **1) Arithmetic Logic Unit (ALU)** and **2) Bidirectional Interface Unit**

The RR-4 ALU consists of a i) binary ⇔ RR-4 conversion unit *(binary to RR4 module, full_adder module* and *RR4 to binary module*), ii) an arithmetic unit *(arithmetic_unit, RR4_adder, RR4_multiplier design* and *partial_product module),* iii) a logic unit *(logic_unit),* iv) a control unit *(control unit module in VHDL),* and v) top level of the RR-4 ALU *(system.vhd).*

1. Problem identification

In recent times, designing coprocessors for parallel fast multiplications of numbers has become an important field of research. Several algorithms have been developed for multiplication of binary numbers that are easily being implemented on a VLSI chip. For example, Nakamura in proposed an algorithm for iterative array multiplication that requires O (n) time to multiply 2 n-bit binary numbers and can be implemented on a VLSI chip using an almost regular interconnection structure among the processing element.
The main problem during the design is the power consumption issues,the LUTs od the design,momory Usage,elapsed time etc.In our design we are implementing a new design showing better results in every aspects.

3. Proposed Work

In this paper we have designed a coprocessor using the Redundant radix4 algorithm and evaluating various performances and result for analysis purpose.

Comparison among the methods

Parameters	Conventional Multiplier	Redundant Binary Radix-4 Multiplier	Proposed Redundant Radix-4 Multiplier
Power	459	226	124
No. of gates	1416	66	36

We have compare our proposed technique with the existing conventional multiplier as well as with the Redundant Binary Radix-4 Multiplier.From our results we found our proposed scheme has more efficiency with respect to the Power consumption and Number of

Gates.Hence we have extended our work to design a coprocessor.the details of the coprocessor is given as follows.

3.1 Architecture of the Coprocessor

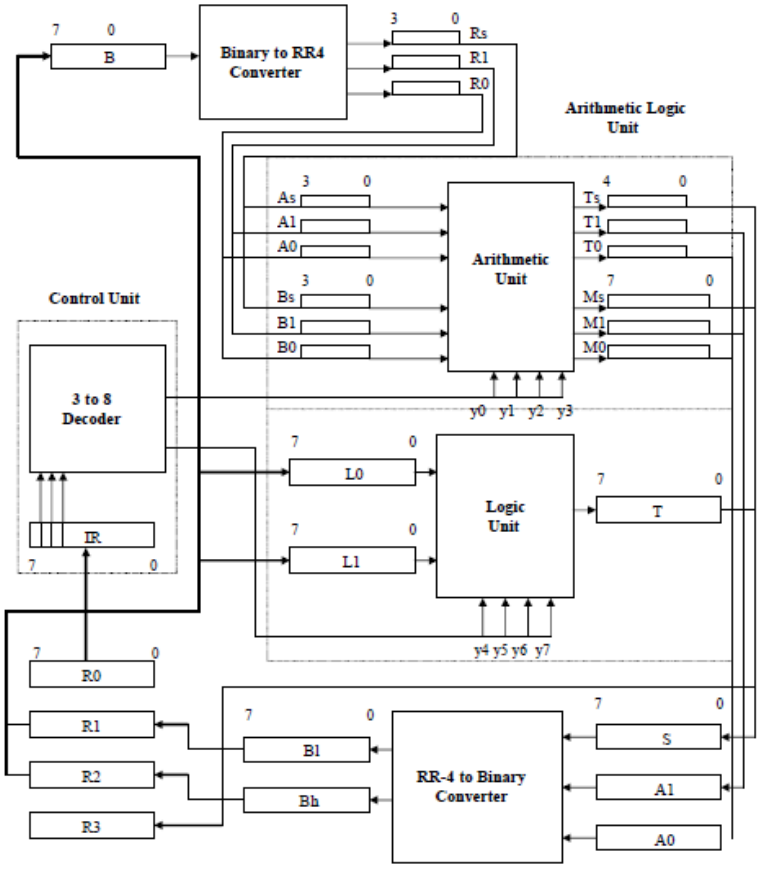

Fig 1.1: RR4 Coprocessor architecture.

3.2.1 The Arithmetic Logic Unit

The arithmetic logic unit is composed of the following separate logic blocks which are neatly interconnected by a detailed glue logic established in the top level VHDL file (system.vhd).

1. Binary to RR-4 Conversion Unit: The arithmetic operations in the RR-4 ALU are preceded by the conversion of 8 bit input operands into their 12 bit equivalent RR-4 representation (each RR-4 digit comprising of 3 binary bits).

After the completion of arithmetic operations, the RR-4 output digits are reconverted to binary bits following the logic described in the VHDL codes.
Finally, the eight, ten or sixteen bit binary outputs for logical, addition or multiplication operations respectively are redestined to the operands registers of the Bidirectional Interface Unit (BIU).

2. Arithmetic Unit: The arithmetic unit of the RR-4 coprocessor performs four basic arithmetic operations using the RR-4 algorithm as discussed earlier:

- **i.** addition,
- **ii.** subtraction,
- **iii.** multiplication, and
- **iv.** complementation

The addition circuitry is composed of two distinct modules RR4_adder module and partial_product module and gives the final sum in RR-4 representation.

The subtraction circuitry is nothing but the same circuit as the addition only with the difference of changing the sign bits of the second operand (as the subtraction can be performed by 2's complement addition of the second operand to the first).

The multiplication is the most complicated and time consuming operation of the coprocessor and it operates by repeated execution of the modules RR4_multiplier.vhd and partial_product.vhd. The final output digits are brought about by RR-4 addition of the partial products.

The fourth arithmetic operation of the coprocessor is the complementation of a given operand. This simple operation is carried out by taking the advantage of the sign-magnitude form of the RR-4 number system. The change of sign of the operand sign bits keeping the two magnitude bits of each RR-4 digit unchanged engenders the complemented output of the operand.

3. Logic Unit: The logic unit also performs four basic logic operations:

- i. AND,
- ii. OR,
- iii. NOT, and
- iv. X-OR.

The logic circuitries are nothing to do with the RR-4 representation of the numbers. Therefore, the logic unit is a conventional logic circuit as seen in a traditional processor.

The logic unit takes two 8-bit operands (for first, second and fourth operations), performs bitwise logic operations and produces an 8-bit output.

4. Control Unit: The control unit of the coprocessor is rather simple and consists of an instruction register (IR) and a 3-to-8 decoder unit. The decoder generates 8 control signals by decoding the instructions corresponding to the operations performed by the coprocessor.

5. Top level module: The top level module of the coprocessor encapsulates all the separate units described in it and provides the interconnection between the separate units.

3.3 Instruction set of the Coprocessor

This Coprocessor performs eight distinct operations whose command words are given below

1. Addition:	0	0	0	X	X	X	X	X
2. Subtraction	0	0	1	X	X	X	X	X
3. Multiplication	0	1	0	X	X	X	X	X
4. Complement	0	1	1	X	X	X	X	X
5. Logical AND	1	0	0	X	X	X	X	X
6. Logical OR	1	0	1	X	X	X	X	X
7. Logical NOT	1	1	0	X	X	X	X	X
8. Logical XOR	1	1	1	X	X	X	X	X

3.3.1 Multiplication in Redundant Radix-4 Number System

The multiplication technique proposed by De and Sinha [1] using radix-4 number representation uses one of the signed-digit (SD) number representation introduced by Avizienis[10] to multiply two m-digit numbers in RR-4 system.We will first discuss about the RR-4 number system.Next the algorithm for conversion from binary to RR-4 system is described, followed by multiplication algorithm as proposed.

3.3.2 The RR-4 Number System

In RR-4 number system the radix used is 4 and individual digits belong to the set, S={-3,-2,-1,0,1,2,3}, An m-digit redundant radix – 4 integer $Y=[y_{m-1}....y_0]_{RR-4}$,where for all i, $y_i \in \{-3,-2,-1,0,1,2,3\}$ and has the value $\Sigma\, y_i \,.\, 4^i$ where i ranges from 0 to (m-1). There are more than one possible representation of the same integer in RR-4 number system. For example, $[0\ 3\ 1]_{RR-4}$, $[1\ -1\ 1]_{RR-4}$, and $[1\ 0\ -3]_{RR-4}$, all represent the number $(13)_{10}$. This redundancy in number representation will be exploited to perform carry propagation – free addition, thereby allowing the parallel addition of four RR-4 numbers in *O(1)* time, independent of the length of the numbers.

Of the different possible representation of RR-4 digits, one possible way of writing the digits of set S using three binary bits for each digit are as follows, where the leftmost bit is 0(1) if the digit is positive(negative):

$(-3)_{RR-4} = 111$
$(-2)_{RR-4} = 110$
$(-1)_{RR-4} = 101$
$(0)_{RR-4} = 000$
$(1)_{RR-4} = 001$
$(2)_{RR-4} = 010$
$(3)_{RR-4} = 011$

The representation of any RR-4 number using binary bits can be visualized by a matrix of 0's and 1's as follows, where each column represents the respective RR-4 digit:

$$(2\ -1\ 0\ 3\ 1)_{RR-4} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The topmost bit in each column indicates the sign of the digit, where 0 stands for positive and 1 stands for negative digit.

Multiplication in RR4 Number System

Multiplicand 1 0 -3 0 — First step of partial product generation — First step of partial product generation
Multiplier 2 -1 1 -2

-2 0 6 0 $\Rightarrow$ { -2 0 2 0 ; 0 0 1 0 } $\Rightarrow$ 0 -2 1 2 0

1 0 -3 0 $\Rightarrow$ { 1 0 -3 0 ; 0 0 0 0 } $\Rightarrow$ 0 1 0 -3 0

-1 0 3 0 $\Rightarrow$ { -1 0 3 0 ; 0 0 0 0 } $\Rightarrow$ 0 -1 0 3 0

2 0 -6 0 $\Rightarrow$ { 2 0 -2 0 ; 0 0 1 0 } $\Rightarrow$ 0 2 -1 -2 0

An example explaining the steps involved in the partial product generation

3 -2 -1 1
0 -3 0 2 — Four RR-4 numbers to be added
1 2 -1 0
1 0 -3 0

---

1 1 -1 3 — Intermediate Sum
1 -1 -1 0 — Intermediate Carry
1 0 0 -1 3 — Final Sum

4. Results & Discussion

Results for RR4 Coprocessor

4.1 Device properties

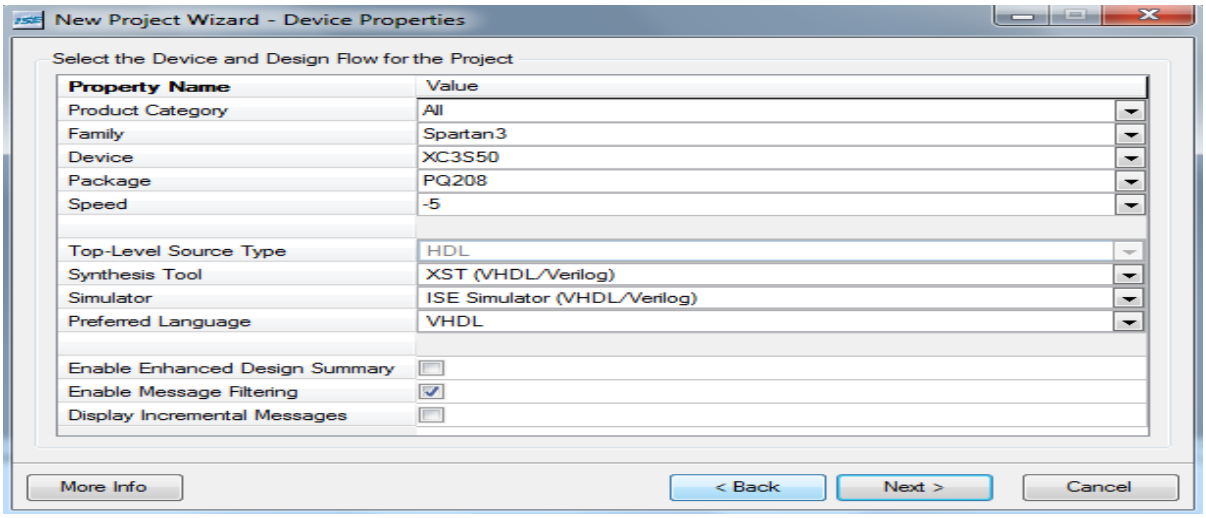

Fig 1.2: Device Utilization Summaries

The above figure shows the selected device properties for the synthesis and Simulation in Xilinx ISE environment.We have considered Spartan 3 family with Xc3S50 Device and XC

4 HDL Synthesis Report of the Coprocessor

==========

HDL Synthesis Report

Macro Statistics

# ROMs	: 40
16x6-bit ROM	: 24
32x6-bit ROM	: 16
# Adders/Subtractors	: 350
16-bit adder	: 2
3-bit adder	: 150
3-bit subtractor	: 4
4-bit adder	: 188
8-bit adder	: 6
# Latches	: 414
1-bit latch	: 403
3-bit latch	: 8
8-bit latch	: 3
# Comparators	: 126
2-bit comparator equal	: 4
2-bit comparator greatequal	: 20
2-bit comparator greater	: 24
3-bit comparator greatequal	: 20
3-bit comparator greater	: 20
3-bit comparator less	: 4
3-bit comparator lessequal	: 2
4-bit comparator greater	: 32
# Multiplexers	: 14
3-bit 4-to-1 multiplexer	: 8
8-bit 4-to-1 multiplexer	: 6
# Logic shifters	: 8
3-bit shifter logical right	: 8
# Xors	: 17
1-bit xor2	: 16
8-bit xor2	: 1

==========

5 Device Utilization Summary of the Coprocessor

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1206	2352	51%
Number of Slice Flip Flops	417	4704	8%
Number of 4 input LUTs	2171	4704	46%
Number of bonded IOBs	56	142	39%
Number of GCLKs	4	4	100%

Fig 1.3: Device Utilization summary of Coprocessor

The above figure shows the device utilization summary for the RR4 Coprocessor.The Summary includes Number of slices,Number of Slice Flipflops,Number of boondede IOBs and Number of Global Clocks.The results may be used while comparing the redundant multiplier with another system with similar results.

6 RTL Schematic Control unit

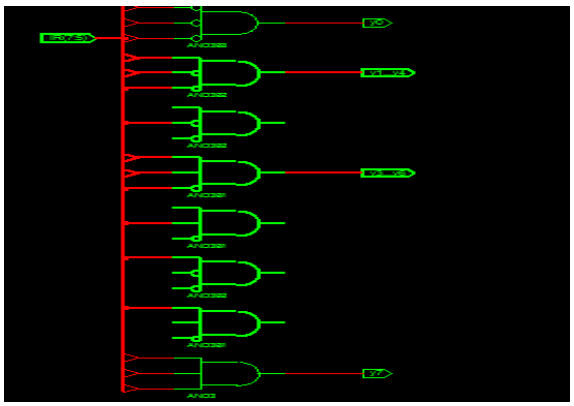

Fig 1.4: RTL Block for the Control Unit of Coprocessor

7 RTL Schematic Logic Unit

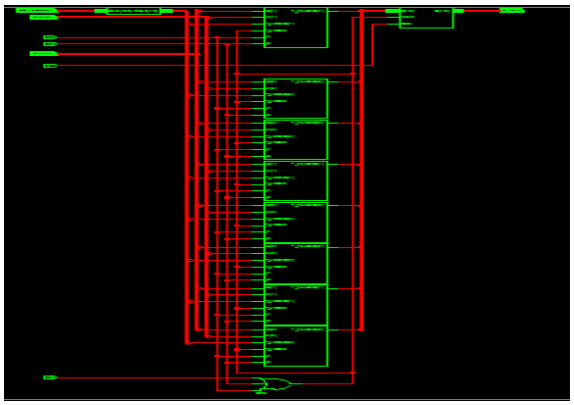

Fig 1.5: RTL Block for the Logic Unit of Coprocessor

8 RTL Schematic Arithmetic Unit

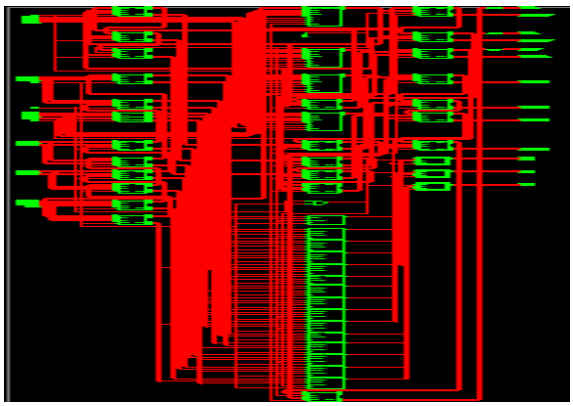

Fig 1.6: RTL Block for the Arithmetic Unit of Coprocessor

The above figure shows the synthesized output for the RR4-Coprocessor' control unit, Logic Unit and Arithmetic Unit For Implementing the program in silicon wafer we have to analyze the above synthesis circuit with the no. of components used for the design.(as shown in HDL Synthesis Report).

9 Simulations for Logic Unit

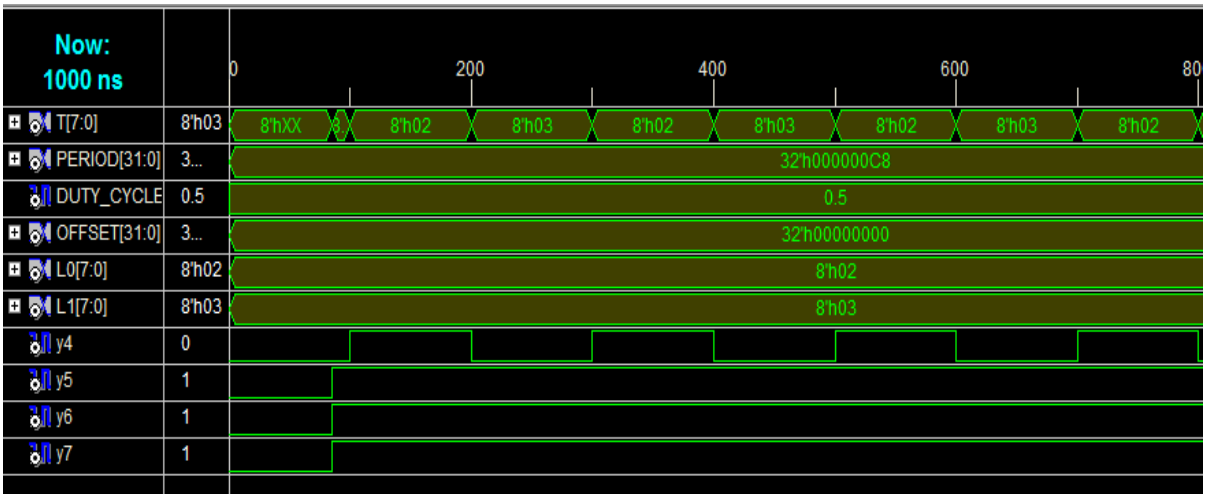

Fig 1.7: Simulation for theLogic Unit of Coprocessor

9.2 Simulations Results for Arithmetic Unit

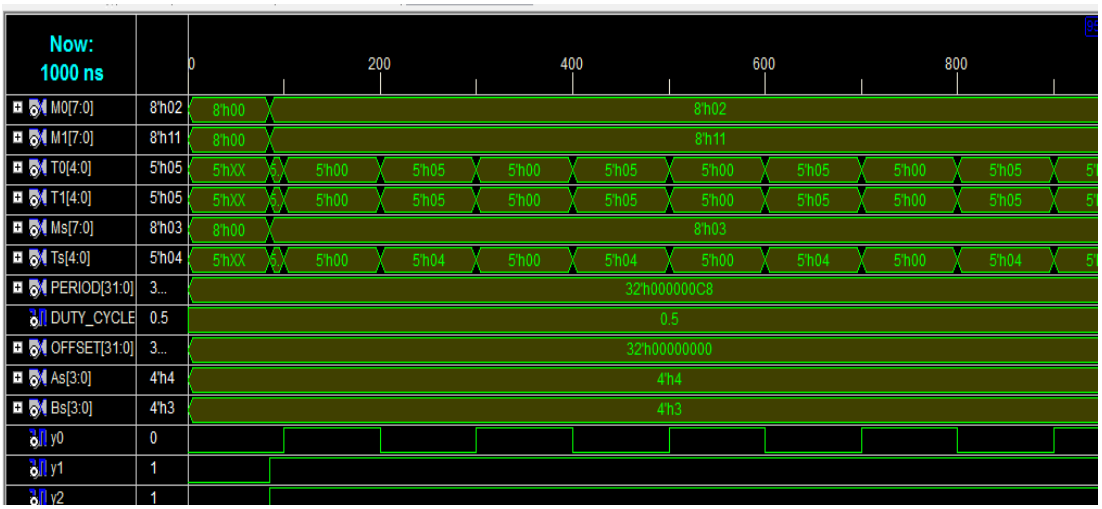

Fig 1.8: Simulation for the Arithmetic Unit of Coprocessor

5. Conclusions

In the proposed workt, we have first designed a RR4 multiplierand compared the results in terms of no. of gates and poer consumption,finally compared the results with the conventional multiplier and redundant binary radix4 multiplier,we found our proposed RR4 based scheme has greater efficiency with all aspects of result.Subsequently we have extended our work in designing 8 bit arithmetic co-processor with the help of redundant radix-4 arithmetic number system. This co-processor is supposed to work on a simple parallel quarternary multiplication algorithm which is the fastest known in the domain of parallel computing. The co-processor is able to perform arithmetic operations like addition, subtraction, multiplication and complementation, and logical operations like logical AND, logical OR, logical NOT and logical XOR.We have successfully synthesized the results and done the simulations in the Xilinx ISE environment.

5. References

1. Sriharish, L., & Kamaraju, M. (2014). A Novel VLSI Architecture of Multiplier on Radix 4 using Redundant Binary Technique. International Journal of Computer Applications, 103(2), 23-28.
2. Antelo, E., Villalba, J., Bruguera, J. D., & Zapata, E. L. (1997). High performance rotation architectures based on the radix-4 CORDIC algorithm.Computers, IEEE Transactions on, 46(8), 855-870.
3. Li, C. C., & Chen, S. G. (1997, April). A radix-4 redundant CORDIC algorithm with fast on-line variable scale factor compensation. In Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on(Vol. 1, pp. 639-642). IEEE.
4. Seo, Y. H., & Kim, D. W. (2010). A new VLSI architecture of parallel multiplier–accumulator based on Radix-2 modified Booth algorithm. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 18(2), 201-208.
5. He, Y., Chang, C. H., Gu, J., & Fahmy, A. H. (2005, May). A novel covalent redundant binary Booth encoder. In Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on (pp. 69-72). IEEE.
6. Wu, H., Hasan, M. A., Blake, I. F., & Gao, S. (2002). Finite field multiplier using redundant representation. Computers, IEEE Transactions on, 51(11), 1306-1316.
7. M. De and B. P. Sinha, “Fast Parallel multiplication using redundant quarternary number system”, Parallel Processing Letters, Vol. 7, pp. 13-23 1997.
8. Alodeep Sanyal, Rajat Shuvra Ghoshal, Achintya Das and Susmita Sur-Kolay A Reconfigurable Coprocessor for Redundant Radix-4 Arithmetic.
9. M. De and B. P. Sinha, “Fast Parallel multiplication using redundant quarternary number system”, Parallel Processing Letters, Vol. 7, pp. 13-23 1997.
10. S. Nakamura, “Algorithms for iterative array multiplication”, IEEE Trans. Comput., Vol.35, pp.713-719, 1986.
11. N. Takagi, H. Yassura, S Yajima, “High Speed VLSI multiplication algorithm with a redundnt binary addition tree” IEEE Trans. Comput., Vol. 34, pp. 789-796, 1985.
12. B. P. Sinha and P. K. Srimani, “Fast parallel algorithms for binary multiplication and their implementation on systolic architectures”, IEEE Trans. Comput.,Vol. 38, pp. 424- 431, 1989.
13. M. De and B. P. Sinha, “Fast parallel algorithm for ternary multiplication using multivalued I2 L technology”, IEEE Trans. Comput., Vol. 43, pp. 603-607,1994.
14. R. P. Brent and H.T. Kung, “A regular layout for parallel adders”, IEEE Trans. Comput., Vol. 31, pp. 260-264,1982.
15. K. Mehlhorn and F. P. Preparata, “Area-time optimal VLSI integer multiplier with minimum computation time”, Information and Control, Vol. 58, pp. 137-156, 1983.
16. A. Karatsuba and Y. Ofman, “Multiplication of multi-digit numbers on automata,” Soviet Physics Doclady , Vol. 7, pp-595-596, 1963.
17. N. Takagi and S. Yajima, “Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem,” IEEE Trans. Comput., Vol. 41, pp. 887-891,1992.

18. A. Avizienis, “Signed-digit number representation for fast parallel arithmetic,” IRE Trans. Electron. Comput., Vol. EC-10, pp. 389-400, 1961.
19. P. J. Ashenden, The designer’s guide to VHDL. San Francisco, California: Morgan Kaufman Publishers Inc. 1996.
20. V. Vetz, J. Rose, A Marquardt, Architecture and CAD for deep-submicron FPGAs. USA: Kluwer Academic Publishers, 1999.
21. Z. Salcic, VHDL and FPLDs in digital systems design, prototyping and customisation. USA: Kluwer Academic Publishers, 1998.
22. Xilinx Data Book, 2000.