International Journal of Engineering & Technology, 7 (2.20) (2018) 14-17

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Floating Point Multiplication Based on Schonhage Strassen Algorithm

B. Srikanth[1]*, M. Siva Kumar[2], K. Hari Kishore[3]

[1]Ph.D Scholar, K L University, Vaddeshwaram, Guntur, Andhra Pradesh, India
[2]Associate Professor K L University, Vaddeshwaram, Guntur, Andhra Pradesh, India
[3]Professor K L University, Vaddeshwaram, Guntur, Andhra Pradesh, India
**Corresponding author E-mail: srikanth.vlsi.2011@gmail.com*

Abstract

In this paper, the single precision float point multiplication is performed using the Schonhage Strassen Algorithm. There are several types of floating point multiplications like Karatsubha and Toom cook. The Schonhage Strassen algorithm is conventionally a fixed point integer multiplication algorithm. The main advantage of the Schonhage Strassen multiplication is that, the multiplication of integer values greater than 5 digits ranging from 2^{215} to 2^{217} bit values proves to be efficient. The validation of the proposed floating point multiplication is done using FPGA real time implementation. The analysis of parameters like area and power are evaluated.

Keywords: *Float point multiplication, Schonhage Strassen Algorithm, Field Programmable Gate Array, Power and Area Analysis*

1. Introduction

The multiplication is the vital arithmetic operation which is utilized in many communications, networks, image processing and video processing applications. The floating point application is challenging in the digital design based applications. The issues of design complexity, large area utilization, and consumption of power make the floating point multiplier to be seldom used easily.

The standard floating point representation is presented in the IEEE 754 as Single Precision Floating Point representation and Double Precision Floating Point representation. The IEEE 754 for SPFP utilizes 32 bits in which 23 for mantissa, 8 for exponent and 1 for sign. This paper considers only the SPFP for the SSA based multiplication method.

The Single Precision Floating point (32-bit IEEE 754) based on FPGA could be designed using only combinational circuits [1].The merging of single precision and double precision representation for adder proves to be optimized with FPGA implementation [2].The SPFP is utilized in the design of the RISC processor proves to be effective with accuracy and speed [3]. Implementation of FPGA based reversible SPFP produces less critical path delay [4]. The challenging portion in this work is interlinking of the SPFP with the SSA algorithm. That is, the 32 bit SPFP has to be converted into the integer equivalent for SSA manipulation. The resolution of 223 bits is used for the conversion of the mantissa into integer equivalent. The evaluation of large fixed point integer multiplication based on SSA is simply a speed comparison with other methods [5]. By selecting an appropriate prime root of unity, decrease in manipulation is achieved by using the modular arithmetic [6].

The FPGA is used for the implementation of the proposed work. The FPGA implemented digital designs are more prominent in multipliers and adders. The FPGA implementation of the Vedic multiplier has less area complexity due to crosswise and vertical calculation [7]. The vedic multiplier improve the speed performance and exhibits less delay when implemented on Xilinx Spartan 3E FPGA [8]. This paper deals with hardware implementation of SPFP-SSA for multiplication. The mixed style VHDL code is implemented using the Xilinx Spartan 3A DSP FPGA device. The parametric analysis of power and area are presented in the results section. The preceding section explains the architecture of the proposed method.

2. The Proposed Method: Single Precision Floating Point Multiplication Based on Schonhage Strassen algorithm

The SSA algorithm uses the Number Theoretic Transform algorithm for the multiplication of the two floating values. The floating values are represented using the fixed point representation. Conventionally, the NTT algorithm is used in the multiplication of the integer values. The proposed SSA algorithm involves in the VHDL coding of converting the Single precision binary values to integer equivalents, multiplying the two integer equivalent and format integer output to the Single precision binary value. The proposed SSA algorithm involves the convolution of two sequences of length N = 8 is performed using the NTT algorithm. The NTT algorithm uses the modulus function for the evaluation of the FFT. The procedure for the NTT algorithm is as follows.

i. The two Single precision values in binary are converted to the two decimal floating numbers.
ii. The two decimal floating numbers are formatted to two 8 valued integer equivalents, since the NTT algorithm works with the integer values.

iii. The two 8 valued integer equivalence are considered as the order of sequence into the NTT algorithm. Note: The zero padding with the 8 valued integer inputs, if required.
iv. The order of the sequence (n) for the FFT is specified as non-negative integer.
v. The modulus M is chosen such that every value of the input sequence is within range of 0 to M. (i.e.,) $1 \leq n \leq M$.
vi. The formula for the working module in the NTT algorithm is given by

$$N = kn+1 \quad (1)$$

Where k is an integer > 1, n is the order of the sequence.
Note: The N value generated using this formula should be a prime number using the Dirichlet's Theorem.
vii. For the n-point DFT, ω is the primitive nth root of unity. This is compensated in the NTT algorithm by using Euler's Theorem defined as

$$\omega = g^k \bmod N \quad (2)$$

Where g is the generator
The value of the generator "g" is selected by using following conditions
a) The value of g is assumed to be say a.
b) The prime number N is considered as N-1 and factorise as two product values say x and y.
c) Now the generator "a" is found by checking for the following

$$a^{N-1} \bmod M=1 \quad (3)$$
$$a^x \bmod M \neq 1 \text{ and } a^y \bmod M \neq 1 \quad (4)$$

viii. After finalizing with the values of N, M and a; the convolution based on NNT algorithm is evaluated
ix. The two n sequences are fed in through the 8X8 matrices. The 8X8 matrices of the NTT algorithm is given below

$$\begin{pmatrix} (W_8^0)^0 & (W_8^0)^1 & (W_8^0)^2 & (W_8^0)^3 & (W_8^0)^4 & (W_8^0)^5 & (W_8^0)^6 & (W_8^0)^7 \\ (W_8^1)^0 & (W_8^1)^1 & (W_8^1)^2 & (W_8^1)^3 & (W_8^1)^4 & (W_8^1)^5 & (W_8^1)^6 & (W_8^1)^7 \\ (W_8^2)^0 & (W_8^2)^1 & (W_8^2)^2 & (W_8^2)^3 & (W_8^2)^4 & (W_8^2)^5 & (W_8^2)^6 & (W_8^2)^7 \\ (W_8^3)^0 & (W_8^3)^1 & (W_8^3)^2 & (W_8^3)^3 & (W_8^3)^4 & (W_8^3)^5 & (W_8^3)^6 & (W_8^3)^7 \\ (W_8^4)^0 & (W_8^4)^1 & (W_8^4)^2 & (W_8^4)^3 & (W_8^4)^4 & (W_8^4)^5 & (W_8^4)^6 & (W_8^4)^7 \\ (W_8^5)^0 & (W_8^5)^1 & (W_8^5)^2 & (W_8^5)^3 & (W_8^5)^4 & (W_8^5)^5 & (W_8^5)^6 & (W_8^5)^7 \\ (W_8^6)^0 & (W_8^6)^1 & (W_8^6)^2 & (W_8^6)^3 & (W_8^6)^4 & (W_8^6)^5 & (W_8^6)^6 & (W_8^6)^7 \\ (W_8^7)^0 & (W_8^7)^1 & (W_8^7)^2 & (W_8^7)^3 & (W_8^7)^4 & (W_8^7)^5 & (W_8^7)^6 & (W_8^7)^7 \end{pmatrix} \quad (5)$$

x. The FFT manipulated values are multiplied for the evaluation of the 8 point values which is again fed the IFFT based NTT algorithm using the following 8X8 matrices as shown below

$$\begin{pmatrix} (W_8^0)^{-0} & (W_8^0)^{-1} & (W_8^0)^{-2} & (W_8^0)^{-3} & (W_8^0)^{-4} & (W_8^0)^{-5} & (W_8^0)^{-6} & (W_8^0)^{-7} \\ (W_8^1)^{-0} & (W_8^1)^{-1} & (W_8^1)^{-2} & (W_8^1)^{-3} & (W_8^1)^{-4} & (W_8^1)^{-5} & (W_8^1)^{-6} & (W_8^1)^{-7} \\ (W_8^2)^{-0} & (W_8^2)^{-1} & (W_8^2)^{-2} & (W_8^2)^{-3} & (W_8^2)^{-4} & (W_8^2)^{-5} & (W_8^2)^{-6} & (W_8^2)^{-7} \\ (W_8^3)^{-0} & (W_8^3)^{-1} & (W_8^3)^{-2} & (W_8^3)^{-3} & (W_8^3)^{-4} & (W_8^3)^{-5} & (W_8^3)^{-6} & (W_8^3)^{-7} \\ (W_8^4)^{-0} & (W_8^4)^{-1} & (W_8^4)^{-2} & (W_8^4)^{-3} & (W_8^4)^{-4} & (W_8^4)^{-5} & (W_8^4)^{-6} & (W_8^4)^{-7} \\ (W_8^5)^{-0} & (W_8^5)^{-1} & (W_8^5)^{-2} & (W_8^5)^{-3} & (W_8^5)^{-4} & (W_8^5)^{5} & (W_8^5)^{-6} & (W_8^5)^{-7} \\ (W_8^6)^{-0} & (W_8^6)^{-1} & (W_8^6)^{-2} & (W_8^6)^{-3} & (W_8^6)^{-4} & (W_8^6)^{-5} & (W_8^6)^{-6} & (W_8^6)^{-7} \\ (W_8^7)^{-0} & (W_8^7)^{-1} & (W_8^7)^{-2} & (W_8^7)^{-3} & (W_8^7)^{-4} & (W_8^7)^{-5} & (W_8^7)^{-6} & (W_8^7)^{-7} \end{pmatrix} \quad (6)$$

xi. The convoluted output of the 8 point input sequences are shifted and added to acquire the desired product of the integers.
xii. The integer product value is converted to the Single precision values. The block diagram for the proposed NTT method is shown in Fig.1

The design flow for the proposed SSA algorithm is given in Fig.2. The inputs are considered in the single precision binary form. The conversion of the single precision value to the integer equivalent is challenging in VHDL code. For example, let us assume the example as 0.1234 and 0.5678 respectively. Now the float values are considered as 1234 and 5678 with the zero padding. That is, the input A={4,3,2,1,0,0,0,0} and input B={8,7,6,5,0,0,0,0}. The VHDL code for the above conversion is developed in mixed style of modeling.
Now the VHDL code is scripted such that the FFT of the input A and B are evaluated using the multiplication of 8X8 matrices shown in equation (5). The convoluted output obtained by the multiplication is converted to the time domain by IFFT based on NTT matrices as given in equation (6). The output produces 8 values that are to be shifted and added to obtain the desired integer product output. The integer product is converted to Single precision representation of its floating point equivalence. The VHDL coding of the proposed method is real time implemented by using the Xilinx Spartan 3A DSP FPGA.

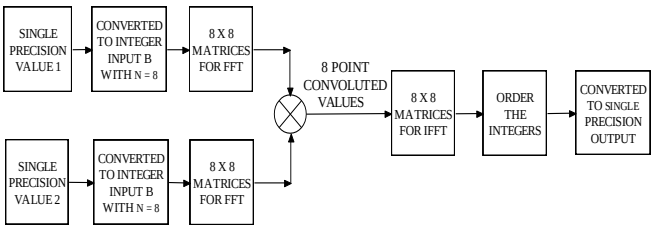

Fig.1: Block diagram of the Number Theoretic Transform with sequence n=8

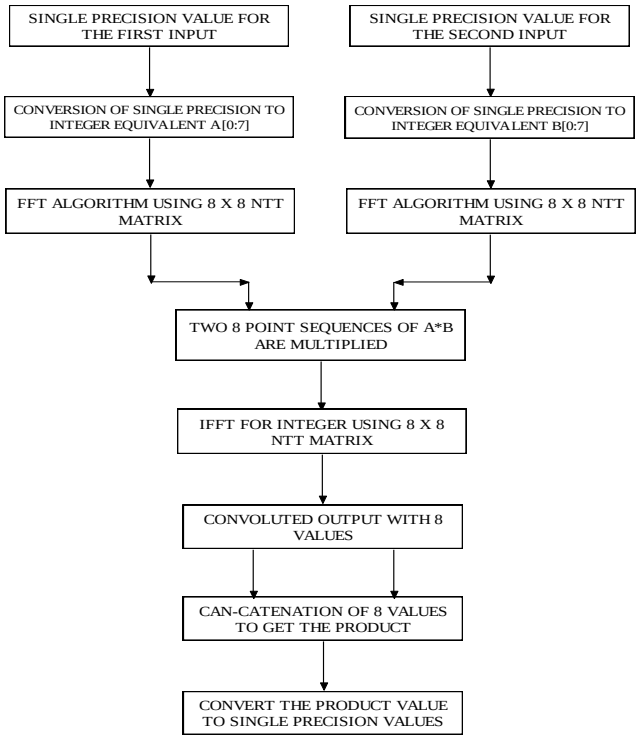

Fig.2 Design flow for the proposed SSA algorithm for floating point multiplication

3. Results and Discussion

The simulated output for the proposed method is depicted in the Fig.3. The SPFP values which are considered in 32 bits for the first set of inputs of 0.4323 and 0.2132 are seen as A={3,2,3,4,0,0,0,0} and B={2,3,2,1,0,0,0,0}.The 8 point sequence value is acceptable for SSA algorithm. The resulting product value of 0.09216636 is also shown in the Fig. 3. The converted SPFP equivalent for the product value is depicted in the below Fig. The second set of SPFP multiplier is tried for the inputs of 0.1233 and 0.5677. The SSA considers that as A={3,3,2,1,0,0,0,0} and

B={7,7,6,5,0,0,0,0}to yield the product output of 0.06999741 represented in SDFP representation. The RTL view of the proposed method is given Fig. 4. The area acquired by the Xilinx Spartan 3A DSP FPGA for the proposed method is given in in Table 1. The power analysis of the proposed method is given in Table 2. The total power consumed is as low as 0.114W by Spartan 3A DSP real time implementation.

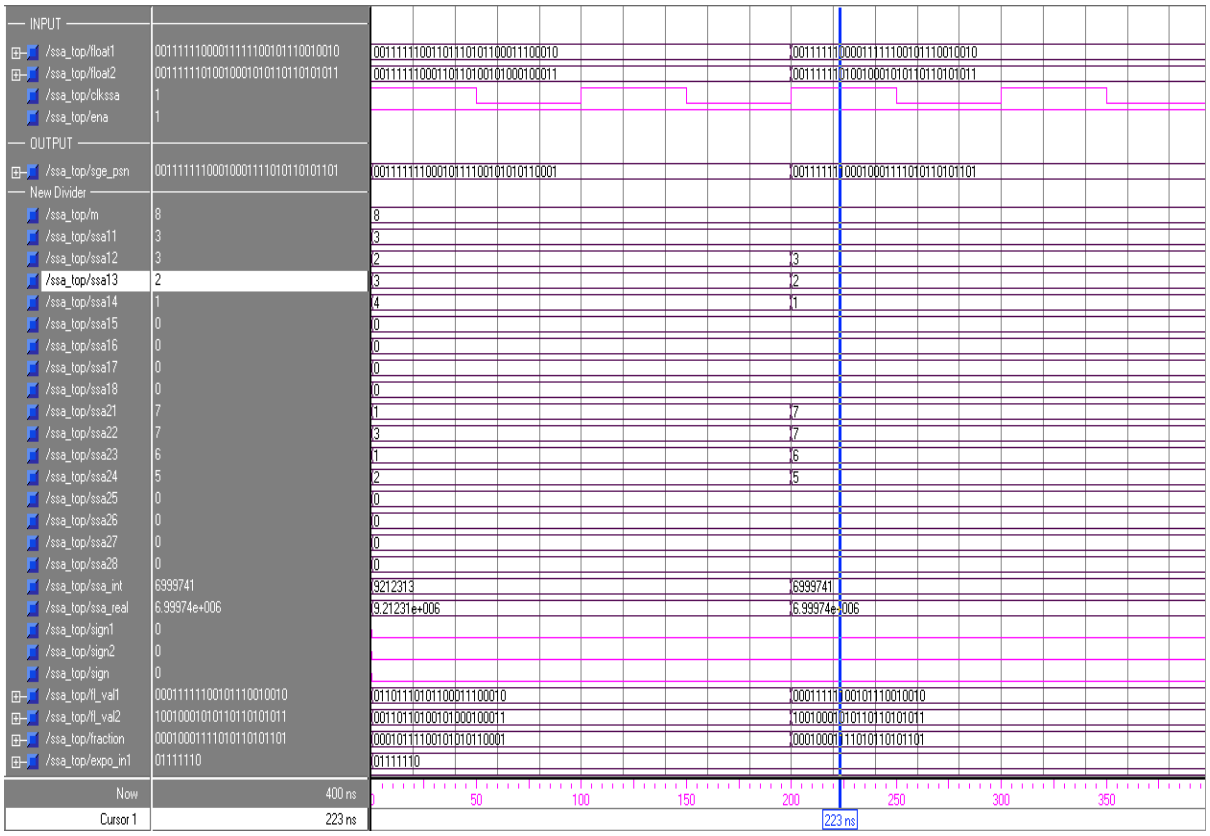

Fig. 3: Simulated output for the proposed SPFP-SSA for two different inputs using the ModelSim Software

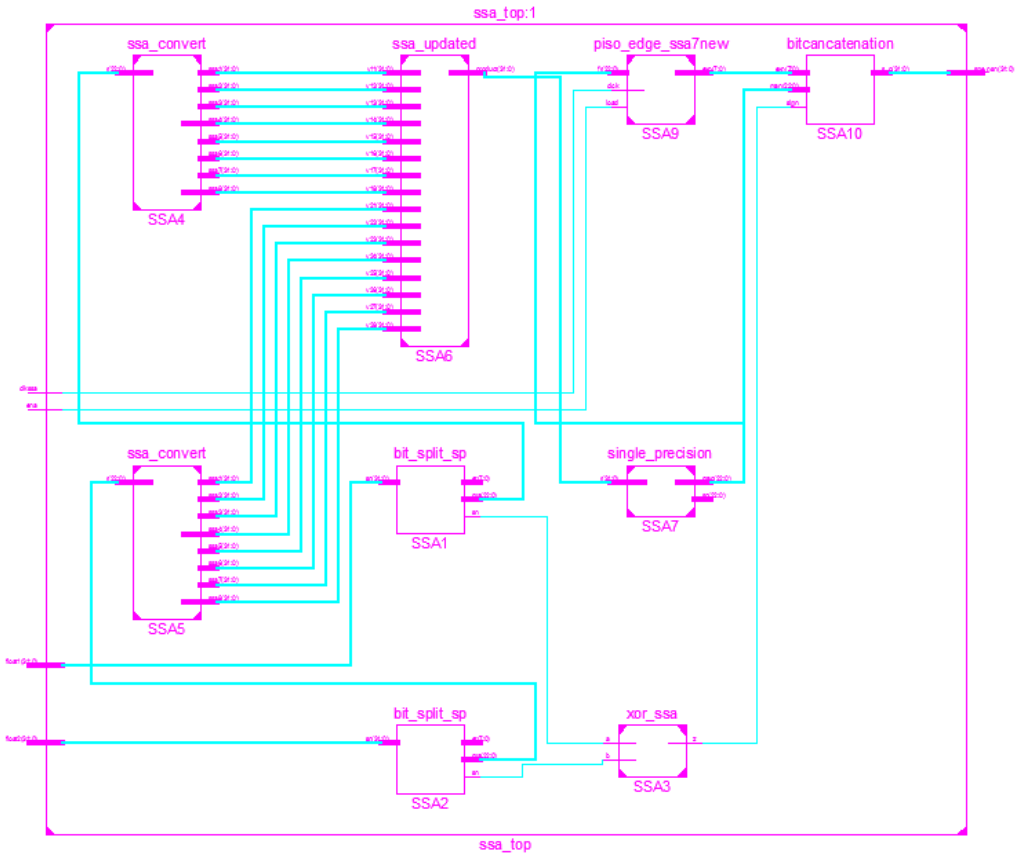

Fig. 4: RTL view of the proposed SPFP-SSA method for floating point multiplication

Table 1 Device Utilization Chart of the proposed method using the Xilinx Spartan 3A DSP FPGA

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Latches	56	33,280	1%
Number of 4 input LUTs	2,180	33,280	6%
Number of occupied Slices	1,282	16,640	7%
Number of Slices containing only related logic	1,282	1,282	100%
Number of Slices containing unrelated logic	0	1,282	0%
Total Number of 4 input LUTs	2,428	33,280	7%
Number used as logic	2,180		
Number used as a route-thru	248		
Number of bonded IOBs	80	519	15%
Number of DSP48As	46	84	54%
Average Fanout of Non-Clock Nets	1.65		

Table 3: Power and Thermal Analysis Chart of the proposed method using the Xilinx Spartan 3A DSP FPGA

Device	
Family	Spartan3adsp
Part	xc3sd1800a
Package	cs484
Grade	Commercial
Process	Typical
Speed Grade	-5

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	8	---	---
Logic	0.000	2366	33280	7.1
Signals	0.000	3966	---	---
IOs	0.000	80	309	25.9
DSPs	0.000	46	84	54.8
Leakage	0.114			
Total	0.114			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	18.0	82.9	27.1

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.042	0.000	0.042
Vccaux	2.500	0.025	0.000	0.025
Vcco25	2.500	0.000	0.000	0.000

	Total	Dynamic	Quiescent
Supply Power (W)	0.114	0.000	0.114

4. Conclusions

The design of Single Precision Floating Point multiplication based on Schonhage Strassen Algorithm is developed using the VHDL proves to be feasible. The area consumed by the proposed method is acceptable and the power consumption of 0.114W is comparatively less considering the complexity involved in the design. Further work could be directed towards the Double Precision Floating Point multiplication based on Schonhage Strassen Algorithm.

Acknowledgment

The authors express sincere gratitude to Department of ECE at K L University for their encouragement during this work. Further, M. Siva Kumar would like to express his gratitude to DST through grant EEQ/2016/000604

References

[1] Ujwal S. Ghate, "Single Precision Floating Point FFT", International Journal of Computer Applications, 2012, pp: 17-19.

[2] Hao Zhang, Dongdong Chen, Seok-Bum Ko"High performance and energy efficient single precision and double-precision merged floating-point adder on FPGA", IET Computers & Digital Techniques, Vol. 12 , No. 1, 2018, pp: 20-29.

[3] Omkar A. Shastri, Shubhangini Ugale and Vipin Bhure, "Review Paper on Parallel Processing Single Precision Floating Point Multiplier based RISC Processor", International Journal of Current Engineering and Technology, Vol.6, No.2, 2016, pp: 459-461.

[4] A.V. AnanthaLakshmi and G.F. Sudha, "Design of an efficient reversible single precision floating point adder", International Journal Computational Intelligence Studies, Vol. 4, No. 1, 2015, pp: 2-30.

[5] Raazesh Sainudiin and Thomas Steinke "A Rigorous Extension of the Sch¨onhage-Strassen Integer Multiplication Algorithm Using Complex Interval Arithmetic", Reliable Computing Vol 18, 2013, pp: 97-116.

[6] Anindya De, Piyush P. Kurur, Chandan Saha, And Ramprasad Saptharishi,"Fast Integer Multiplication Using Modular Arithmetic" Society For Industrial And Applied Mathematics", Vol. 42, No. 2, 2013, pp. 685–699.

[7] Kolli V Jayalakshmi1, Medikonda Ashok Kumar, "Implementation of High Speed IEEE 754 Single Precision Floating Point Multiplier using VEDIC" International Journal of Applied Sciences, Engineering and Management, Vol. 07, No.01, 2018, pp. 58-62.

[8] Pooja Hatwalne, Ameya Deshmukh, Tanmay Paliwal and Krupal Lambat, "Design and Implementation of Single Precision Floating Point Multiplier using Vhdl on Spartan 3", International Journal of Latest Trends in Engineering and Technology,Vol.8, No. 3, pp.263-269.