INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

Optimized Binary Floating Multiplier using Bypassing Technique

Saransh Shrivastava[1], Rajani Gupta[2]

[1] Kailash Narayan Patidar College of Science and Technology
Baghmugaliya, Bhopal 462001, India
saransh.stva@gmail.com
[2]Kailash Narayan Patidar College of Science and Technology
Baghmugaliya, Bhopal 462001, India
rajni_gupta2007@yahoo.com

Abstract: *Floating point multiplication is used in many digital signal processing (DSP) applications; this multiplication is costly in terms of area, delay and power consumption. Reduction in these parameters will greatly improve the performance of multiplier and this in turn will improve the performance of DSP systems. In this paper, a binary floating point multiplier is implemented using column bypass technique. The multiplier supports IEEE P754 standard, in this work IEEE P754 – 32 bit format is followed. Column bypass multiplier is used to reduce dynamic power consumption. The multiplier is implemented on Virtex 5-XC5VLX-3FF324 FPGA.*

Keywords: *Binary floating point multiplication, IEEE P754 standard, dynamic power consumption, resource utilization, delay, column bypass technique, FPGA*

1. INTRODUCTION

Floating point multiplication is the core process in any DSP application and optimizing the floating point multiplication process will improve the performance of overall application. Binary floating point multiplication is used in almost all the DSP applications that we are using today. IEEE has given a standard called IEEE P754 standard for floating point numbers, the two most commonly used formats are single precision format (32 bit) and double precision format (64 bit), in this work we have used single precision format for representing binary floating numbers. Figure 1 shows the single precision format. In single precision format first 23 bits (0 - 22) are used to represent mantissa, in binary floating point representation 1additional bit is concatenated as MSB in mantissa for normalization. Next 8 bits (23 - 30) are used to represent exponent, this exponent is biased to 127 so that the exponent never becomes negative. And the last bit (31) is used to represent sign; 1 for negative and 0 for positive numbers [1, 2].

32 bits (0 - 31)		
1 bit	8 bits	23 bits
Sign	Exponent	Mantissa

A fixed point multiplication is required in implementing floating point multiplication, this fixed point multiplication can be implemented with many algorithms like Braun's multiplier [3], shift and add multiplier [4], Vedic multiplier [5], Wallace tree multiplier [6], Booths algorithm [7], bypass techniques [8] and many more. In this work our focus was to reduce power consumption of our design so we have opted column bypass multiplication technique.

The paper is organized as follows: In section II previous work about floating point multiplication is discussed, in section III floating point multiplication algorithm is discussed along with its hardware implementation. Section IV will have details about column bypass technique and its effect on switching and dynamic power consumption. Section V contains details about behavioral simulation, synthesis report, delay report and power consumption report along with conclusion.

2. RELATED WORK

Many floating point multipliers are available in literature, [12] suggested a fast 32 bit floating point multiplier, this multiplier is optimized for low latency; the design makes use of adders which has least delay-power product. [13]

Presents a quadruple precision floating point multiplier, this design uses smaller multiplier for implementing large multipliers; this in turn reduces the number of DSP48 by 50% with a slight overhead of additional slices. [14] Presents a twin precision multiplier with 2D bypassing technique, this paper suggested a fixed point multiplier optimized for power, in this design the full adders inside the multiplier unit are turnoff when any of

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

the input bits is/are zero. The paper suggested a 29% reduction in power with an area overhead. [15] Suggested two designs of hybrid fixed and floating point multipliers, one optimized for cost (area) and second optimized for performance (latency), design one suggested a 45% area improvement over the other design (low latency), whereas design two has 35% less latency compared to design one, the multiplier is capable of multiplying both fixed and floating point numbers. [5] Presents a multiplier which makes use of vedic technique, this design uses the Urdhva - triyakbhyam sutra to multiply two numbers, using this technique reduces the number of partial products generated during multiplication and this in-turn reduces the resource utilization of FPGA, this also reduces the latency as gate count reduces due to less hardware. [8] Suggested a column bypass multiplier, this multiplier is similar to an array multiplier but with lower power consumption, the design makes use of the fact that if any of the input bit is zero then the result of addition is same as the other bit to be added, hence in this method the full adders of that column is turned off and this in-turn reduces the dynamic power consumption of design. This design has lower power consumption than Braun's multiplier with slight area overhead. In this work we have used column bypass technique to implement binary floating point multiplier; this technique reduces the dynamic power consumption of the target device (Virtex 5) by reducing the switching activity inside FPGA.

3. FLOATING POINT MULTIPLIER

Figure 2 depicts the proposed binary floating point multiplier, in this work we have implemented multiplier with both Vedic multiplication technique and column bypass technique, it is found in results that column bypass technique is better in of dynamic power consumption with very small increase in latency.

As seen in figure 2 floating point multiplier can be portioned into following units

a. IEEE P754 decoder unit
b. 24 bit multiplier
c. Exponent calculation unit
d. Sign calculation
e. Flag generation unit
f. IEEE P754 encoding unit

a) IEEE P754 decoder unit: The standard representation of floating point number is $(-1)^s 2^e (b_0.b_1 b_2 \dots \dots \dots \dots b_{p-1})$

The biased exponent E is original exponent e + 127 and the fraction M = $b_1 b_2 \dots \dots \dots \dots b_{p-1})$

The decoder unit extracts the three information from the incoming packet of 32 bits, first sign information S which is bit 31 refer figure 1.

Second, it extracts exponent information E which lies between bits 30 - 23. And last the mantissa from bits 22 - 0. Additional 1 is concatenated at the MSB location in mantissa for normalization and normalized mantissa M is produced. Here 2 decoders are used one for input A and other for input B. we get Sa, Ea and Ma from decoder A and Sb, Eb, Mb form decoder B

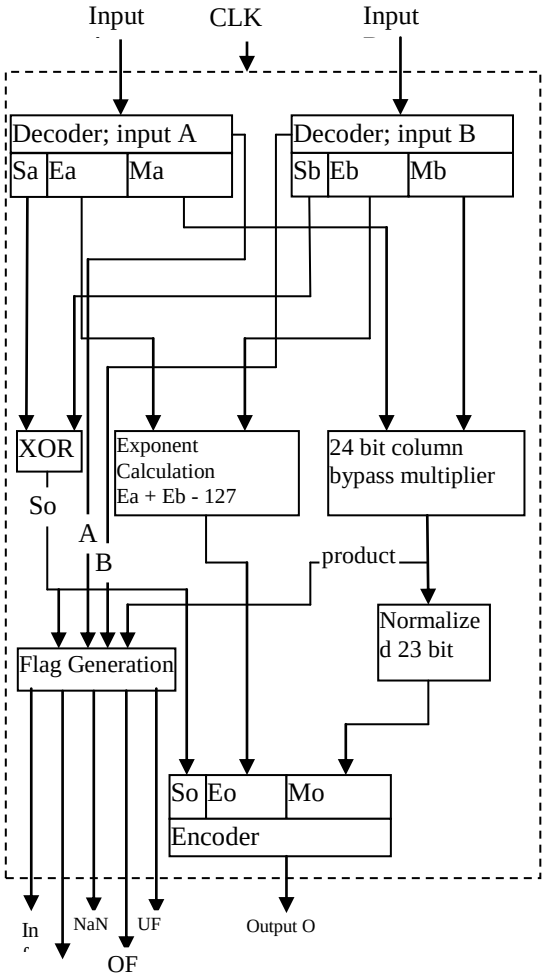

Figure 2: Proposed floating point multiplier architecture; single precision format

b) 24 bit multiplier: A fixed point multiplication is required to perform floating point multiplication. Many fixed point multiplication algorithms are available in literature. Here our focus was to reduce power consumption of multiplier so we have chosen column bypass technique. This technique is will be explained in detail in section III.

c) Exponent Calculation unit: In IEEE P754 single precision format the exponents is biased to 127. To calculate the final output exponent first we need to add

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS.....

the two exponents Ea and Eb obtained from decoder unit and then subtract 127.

d) We get Sa and Sb from decoder, one is used to represent negative number and zero is used to represent positive number. Output sign can be obtained by simply XORING the two sign bits Sa and Sb.

e) Flag generation unit: Five different flags are generated namely; InF (infinity), Zero, NaN (not a number), OF (overflow) and UF (underflow) see figure 2. The various conditions which affect these flags are listed below.

1) E = 255 and M ≠ 0, then NaN
2) E = 255 and M = 0, then InF
3) If 0 < E < 255, then number is $(-1)^S 2^{E-127}(1.M)$
4) If E = 0 and F ≠ 0, then $(-1)^S 2^{126}(0.M)$
5) If E = 0 and M = 0, then Zero
6) If some information is lost in truncation of mantissa then overflow and negative of overflow is underflow.

f) IEEE P754 Encoding unit: So, Eo and Mo are encoded to produce the output result format. The 48 bits from multiplier are truncated to most significant 24 bits

g) Then the MSB is also truncated to produce the 23 bit normalized mantissa Mo.

Algorithm for Binary floating point multiplication

1. Extract Sa, Ea, Ma, Sb, Eb, Mb by decoding the incoming packets.
2. So <= Sa XOR Sb
3. Eo <= Ea + Eb – 127
4. Product <= Ma * Mb
5. Truncate product and the normalize to produce Mo
6. Generate appropriate flags InF, NaN, Zero, OF,UF.
7. Encode to output result format.

Figure 3: Algorithm for binary floating point multiplication

Consider two floating point numbers A = -19.0 and B = 9.5, there normalized binary representation are A = –1.0011x 2^4 and B = 1.0011 X 2^3. IEEE representation of operands is

Packet	sign	Exponent	Mantissa
A =	1	10000011	00110000000000000000000
B =	0	10000010	00110000000000000000000

Here, MSB of both packets A and B shows the sign bits Sa (1_2) and Sb (0_2) respectively, the output sign is calculated by XORING Sa and Sb (1_2 XOR 0_2), so output sign bit So will be 1_2 here. The exponents are biased to 127_{10}. Ea is 4_{10} + 127_{10} = 131_{10} (10000011b) and Eb is 3_{10} +127_{10} = 131_{10} (10000010_2). The output exponent is calculated by adding both exponents and then subtracting the biased value which is 127_{10} in single precision format, so output exponent will be 132_{10} + 131_{10} - 127_{10} = 136_{10} (10001000_2). To calculate the output mantissa first, the two mantissa are concatenated with 1_2 and mantissa for multiplication are formed:

Ma = 10011000000000000000000

Mb = 10011000000000000000000

Now multiplication of mantissa is accomplished using a 24 bit multiplier and 48 bit product is formed:

010110100100000000000000000000000000000000000000

Now the most significant 1 is truncated and least significant 24 bits are also truncated to form the output mantissa Mo.

01101001000000000000000.

Now the output packet is formed by encoding So, Eo and Mo:

Packet	sign	Exponent	Mantissa
O =	1	10000110	01101001000000000000000

The result can be understood as: A X B = -19.0 X 9.5 = 180.5 = -1.01101001 X $2^{134-127}$ = $(-1.0110100.1)_2$ = $(-180.5)_{10}$ [9].

4. COLUMN BYPASS MULTIPLIER DESIGN

The performance of fixed point multiplier unit dominates the performance of overall floating point multiplier unit. In this work our focus was to reduce the dynamic power consumption of design so we have opted column bypass multiplier.

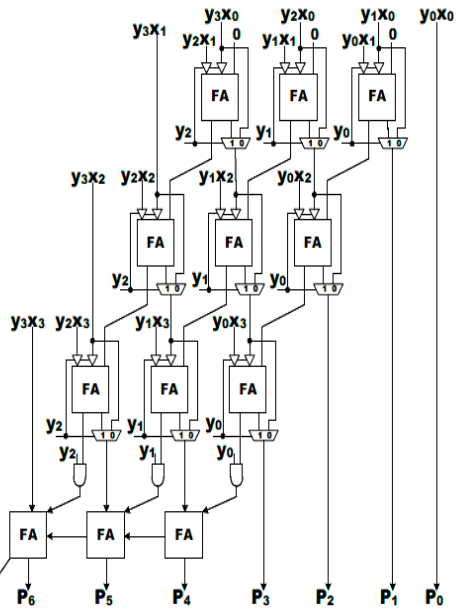

Figure 3: Column bypass multiplier

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

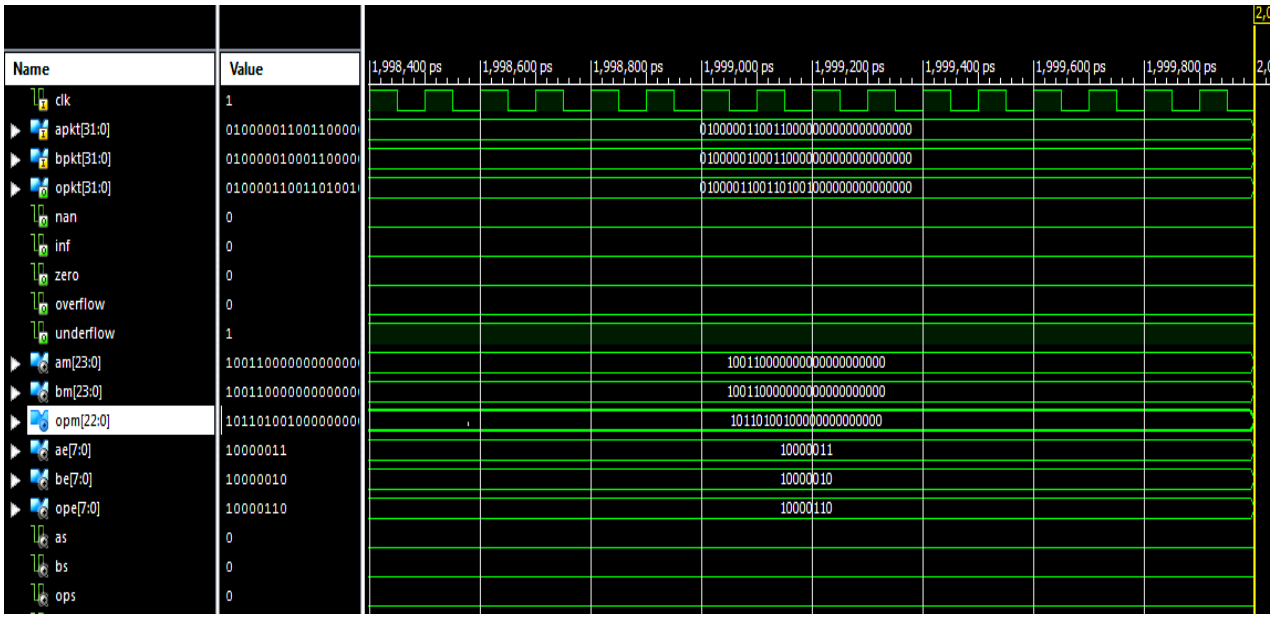

Figure 4: Behavioral Simulation of binary floating point multiplier

Consider an array multiplier [3], which has many full adders it, when one of the two bits is zero the resultant sum, is same as the other bit. In simple array multiplier this zero is added and this causes unwanted switching inside the full adder. Bypass techniques [10], uses this property of full adder and suppress unwanted switching, this switching is the cause of dynamic power consumption so by reducing this switching dynamic power consumption can be reduced. Column bypass multiplier as shown in figure 3 is a bypass method which is used to reduce unwanted switching of full adder and this in-turn reduces dynamic power consumption. In our design we have used a 24 bit column bypass multiplier. The advantage of choosing this architecture is suppressing unwanted switching inside the multiplier unit. The column addition can be bypassed, when the bit of multiplicand, y_i is 0, $0 \leq i \leq n-2$. This causes all partial products $y_i x_i = 0$, $0 \leq j \leq n-1$, thus all full adders can be disabled in the i^{th} column. This reduces the unwanted switching and in turn reduces the dynamic power consumption [8].

5. RESULTS AND CONCLUSION

In this work we have implemented a binary floating point multiplier using column bypass technique. We have used Virtex 5-XC5VLX-3FF324 FPGA. Xilinx ISIM was used for behavioral simulation; Xilinx XST was used to implement the design. X-Power Analyzer was used to calculate power consumption of device. Table 1 shows the design summary. Figure 4 shows the behavioral simulation of binary floating point multiplier.

TABLE 1: Design Summary

Parameters	***Our Design***	***Vedic***	***[11]***
Device	*Virtex 5*	*Virtex 5*	*Virtex 2p*
Power Consumption	*23mW*	*27.29mW*	*55mW*
Time Delay	*5.745ns*	*5.246ns*	*3.070ns*
Number of BUFGs	*1*	*NA*	*NA*
Number of OLOGICs	*33*	*NA*	*NA*
Number of LUTs	*859*	*966*	*1316*
Number of Slices	*299*	*NA*	*NA*
Number of Slice Registers	*0*	*NA*	*NA*
Number of Slice LUT FF Pairs	*859*	*NA*	*NA*
Number of IO	*102*	*99*	*100*
Power Delay Product	*132.135pJ*	*143.16pJ*	*168.85Pj*

NA: Not Available

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS.....

As we can see from the table 1 that power consumption is reduced to 23mW from 27.29mW as compared to binary floating point multiplier using Vedic multiplication technique [5]. This is because of the use of column bypass multiplier. A small increase in delay is observed this is because of the use of complex multiplier. But overall performance of the device was improved. Also Number of slice registers used in our design is zero as compared to Vedic technique which uses 406, so the overall resource usage comparable.

References

[1] IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008

[2] Brian Hickmann, Andrew Krioukov, and Michael Schulte, Mark Erle,"A Parallel IEEE 754 Decimal Floating-Point Multiplier," In 25th International Conference on Computer Design ICCD, Oct. 2007.

[3] R Anitha and V Bagyaveereswan "Braun's Multiplier Implementation using FPGA with Bypassing Techniques" International journal of VLSI design and communication system (VLSICS) vol2, no 3. September 2011.

[4] C.N. Marimuthu, P. Thangaraj and Aswathy Ramesan "Low power shift and add multiplier design" International jaournal of computer scince and information technology 2.3 (2010) 12-15.

[5] Aniruddha Kanhe, Shishir Kumar Das and Ankit Kumar Singh "Design and implementation of floating point multiplier based on vedic multiplication technique " International Conference on communication, information & computing technology (ICCICT), oct 2011 IEEE.

[6] Fadavi-Ardekani "M*N Booth encoded multiplier generator using optimized Wallace trees" IEEE transactions on very large scale integration (VLSI) systems, vol1 issue 2, june 1993 pp 120-125

[7] Shanbang N.R. "Parallel implementation of a 4*4 bit multiplier using a modified Booth's algorithm" IEEE journal of solid stste circuits vol 23 issue 4 Aug 1988 pp 1010-1013

[8] Ming-chen Wen, Sying-Jyan Wang and Yen-Nan Lin "Low power multiplier with column bypassing" Internation symposium on circuits and systems, may 2005 vol. 2 pp 1638-1641.

[9] S. S. Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur, Girish V A, "Implementation of Vedic Multiplier for Digital Signal Processing," International Journal of Computer Applications (IJCA) 2011.

[10] Jun-ni Ohban, Moshnyaga V.G. and Inoue K "Multiplier energy reduction through bypassing of partial products" 2002 Asia-pacific conference on circuits and systems, vol. 2 pp 13-17.

[11] Kavita Khare, R.P.Singh, Nilay Khare,"Comparison of pipelined IEEE-754 standard floating point multiplier with unpipelined multiplier" Journal of Scientific & Industrial Research Vol.65, pages 900-904 November 2006.

[12] Anna Jain, Baisakhy dash and Ajit Kumar Panda "FPGA design of fast 32-bit floating point multiplier unit"

[13] Manish Kumar Jaiswal and Ray C.C. Cheung "Area-Efficient FPGA implementation of quadruple precision floating point multiplier" 26th International parallel and distributed processing symposium workshop and Phd forum, IEEE computer society, 2012.

[14] Syed Ershad Ahmed, Sibi Abraham, Sreehari Veeramanchaneni and Moorthy Muthukrishan and M.B. Srinivas "A modified Twin Precision Multiplier with 2D Bypassing technique" International Symposium on electronic system design, IEEE computer society 2012.