International Journal For Technological Research In Engineering
Volume 4, Issue 10, June-2017 *ISSN (Online): 2347 - 4718*

FPGA IMPLEMENTATION OF POWER EFFICIENT SINGLE PRECISION BINARY FLOATING POINT PROCESSOR

Ankit Pandya[1], Nitesh Dodkey[2], Niraj Umale[3]
Dept. of Electronics and Communication Engineering, Surabhi group of institution, India

Abstract: The use of floating point unit has lot of application in real time embedded systems. Algorithms like fast Fourier transform(FFT) from the digital signal processing (DSP) domain often make extensive use of floating-point arithmetic. This paper presents the design and implementation of an efficient single precision binary floating-point processor in FPGA. The operations offered are floating point addition, floating subtraction, integer addition, integer subtraction, floating point multiplication and integer multiplication. Column bypass multiplication is used to implement multiplier for lower power consumption. Also hardware sharing technique is used for lower resources usage.
Keywords: ALU, Low Power, Dynamic Power, Column bypass technique, FPGA.

I. INTRODUCTION

Floating point unit is the core process in any DSP application and optimizing the floating point unit will improve the performance of overall application. Binary floating point unit 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

Figure 1. IEEE single precision format 32 bit format for binary floating point numbers

In this work we have implemented a single precision binary floating point unit which can perform six operations: floating point addition, floating point subtraction, floating point multiplication, integer addition, integer subtraction and integer multiplication. We have used hardware sharing technique for performing subset operations and column bypass multiplier for lower power consumption.

II. BINARY FLOATING POINT ARITHMETIC & LOGIC UNIT

Figure 2 shows the high level block diagram of floating point arithmetic point unit. 32 bit IEEE P754 numbers is assigned to “apkt” and “bpkt”, the operation to be selected is assigned to the “operation” input of the machine and to synchronize the process clock is assigned to “clk” input of the machine. Table 1 shows the operation to be performed over the machine.

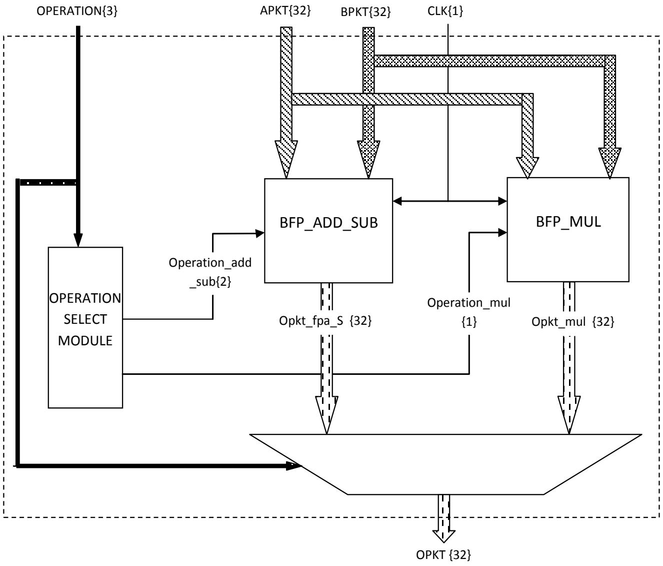

Figure 2. High level block diagram of binary floating point arithmetic and logic unit

Table 1: Operation selection table

S.no	Operation	Operation to be performed
1	000	Floating point addition
2	001	Floating point Subtraction
3	010	Integer addition
4	011	Integer subtraction
5	100	Floating point multiplication
6	101	Integer multiplication

The inputs “apkt” and “bpkt” is applied to the binary floating point adder_subtraction unit BFP_ADD_SUB and binary floating point multiplier unit BFP_MUL. BFP_ADD_SUB can perform four operation: binary floating addition, binary floating point subtraction, integer addition and integer subtraction, the operation to be performed is instructed via “operation_add_sub” signal. BFP_MUL can perform two operation binary floating point multiplication and integer multiplication, the operation to be performed is instructed via “operation_mul” signal. Output multiplexer selects the output from the two units depending upon the operation input.

III. BINARY FLOATING POINT ADDER SUBTRACTOR UNIT

Figure 3 shows the algorithm for floating point addition/subtraction.

Step 1: Decode the inputs apkt and bpkt to obtain
(as,aE,am) and (bs,bE,bm)
Step 2: Determine effective operation (EOP)
If operation = FPA then EOP <= As XOR Bs; else EOP <= not (As xor Bs)
EOP = 0 → Floating Point Addition
EOP = 1 → Floating Point Subtract
Step 3: if apkt < bpkt, then determine large and small number
Step 4: Calculate d <= exp_large – exp_small
Step 5: Shift right 'mant_small' by d;mant_small_shifted
Step 6: Compute rma_unnormalized <= mant_large + mant_small_shifted (if EOP = 0)
Else
Compute rms_unormalized <= mant_large – mant_small (if EOP = 1);
Step 7: Normalize rma_unnormalized; rma_normalized & rma_exp and normalize rms_unnormalized; rms_normalized & rs_exp
Step 8:
if apkt >= bpkt then
RA_sign <= as;
RS_sign <= as;
else
RA_sign <= bs;
RS_sign <= not bs;
Step 9: Encode to IEEE P754-2008 format

Figure 3. Binary Floating Point addition/subtraction algorithm

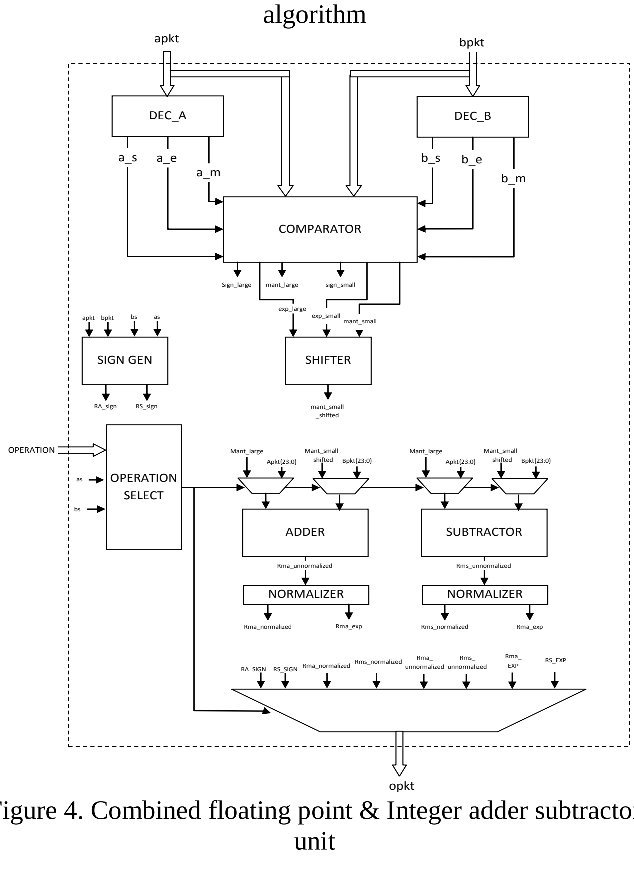

Figure 4. Combined floating point & Integer adder subtractor unit

Figure 4 shows the combined floating point & integer adder subtractor unit. if the selected operation is integer addition then the 24 bits of apkt(23:0) and bpkt(23:0) are directly applied to the adder, rma_unormalized is the final output available at opkt(23:0). Similarly if the selected operation is subtraction then the 24 bits of apkt(23:0) and bpkt(23:0) are directly applied to the subtractor, rms_unormalized is the final output available at opkt(23:0). If the selected operation is floating point addition then the algorithm of figure 3 is followed. Step 1 decoding is performed by input decoders DEC_A and DEC_B. Effective operation is determined by operation select unit step 2. Step 3 comparison is performed by COMPARATOR, the smaller mantissa is shifted by d by SHIFTER; step 4 & step 5. Step 6 is performed by either adder or subtractor depending on EOP. Step 7 normalization is performed by NORMALIZOR. Step 8 is implemented by SIGN GEN. Step 9 is implemented by output multiplexer.

IV. BINARY FLOATING MULTIPLIER UNIT

Figure 5 shows the binary floating point multiplication algorithm.

Algorithm for Binary floating point multiplication
Step 1: Extract As, Am, Be, Bs, Bm, Be
Step 2: Ops <= As XOR Bs
Step 3: Ope <= Ae + Be – 127
Step 4: Product <= Am * Bm
Step 5: Truncate product and the normalize to produce opm
Step 6:Encode result data in IEEE P754.

Figure 5. Binary floating point multiplication algorithm

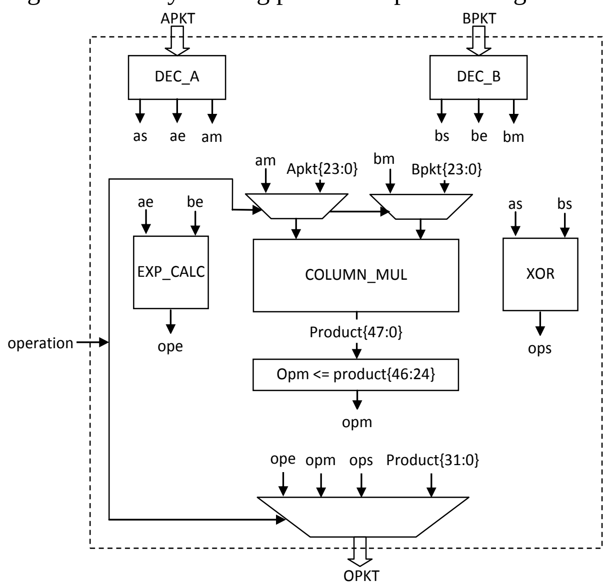

Figure 6. Combined floating point & integer multiplier

Figure 6 shows the internal architecture of combined floating point & integer multiplier. If the selected operation is integer multiplication then 24 bits of apkt(23:0) and bpkt(23:0) is directly assigned to column multiplier COLUMN_MUL and 32 bits of product is assigned to the output port opkt. If the selected operation is binary floating point multiplication then algorithm of figure 5 is followed. Decoding is implemented using two decoders DEC_A and DEC_B. Result exponent ope is calculated using EXP_CALC unit. Output sign is calculated by simply XORing the two sign inputs. The

mantissa multiplication is implemented by low power column bypass multiplier. The output of the floating point multiplier needs to be converted into IEEE P754 format, so the output of the column bypass multiplier is truncated and exponent is updated accordingly.
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.

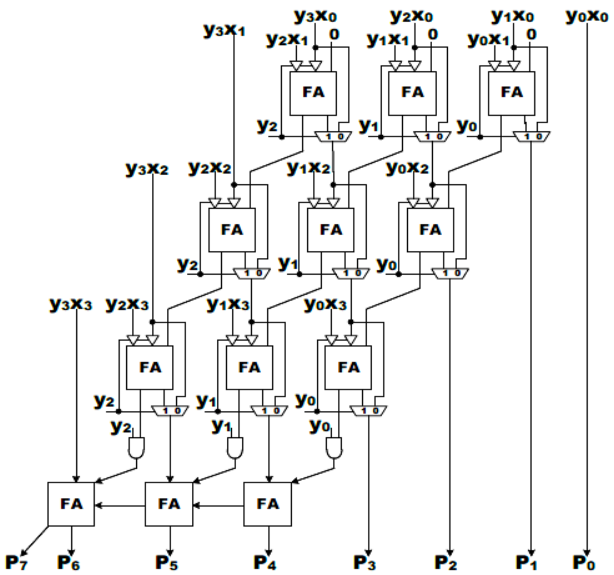

Figure 7. Column Bypass Multiplier

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, yi is 0, $0 \leq i \leq n - 2$. This causes all partial products yixi = 0, $0 \leq j \leq n - 1$, thus all full adders can be disabled in the ith column. This reduces the unwanted switching and in turn reduces the dynamic power consumption [8].

V. RESULTS

The design shown in this paper is targeted for Xilinx Virtex 5 device. Table 2 shows the device utilization summary.

Table 2: Device Utilization Summary

Design	Parameter	Proposed Work	[15]
Adder	Slice LUTs	24	48
	Delay	9.838ns	23ns
Multiplier	Slice LUTs	860	1165
	Delay	16.641ns	22.61ns
Binary Floating Point Add/Sub	Slice LUTs	225	NA
	Delay	8.196ns	NA
Binary Floating Point Multiplier	Slice LUTs	912	NA
	Delay	5.745ns	NA
Combined Floating Point Unit	Slice LUTs	1537	NA
	Delay	14.47ns	NA

VI. CONCLUSION

In this work binary floating point processor is developed and implemented on Virtex 5 device. Six operations can be performed using the proposed unit. It can be observed form table 2 device utilization summary that major portion of the resources is obtained by multiplier, hence in this work we have used Column bypass multiplier in addition with Vedic multiplier to reduce the area and power consumption. It can also be observed form the device utilization summary that proposed design is better in terms of resource usage.

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 state 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.

[15] Lasith K K, Anoop Thomas "Efficient Implementation Of Single Precision Floating Point Processor In FPGA" AICERA-2014 iCMMD.

