ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 7, July 2015

DESIGN AND COMPARISON OF HIGH SPEED SINGLE PRECISION FLOATING POINT MULTIPLIER USING RIPPLE CARRY ADDER AND CARRY LOOKAHEAD ADDER

Reshmashree.M.S , Meenakshi L Rathod

Abstract **– This paper presents designing of a single precision floating point multiplier which follows the Dadda algorithm using Ripple carry adder and carry look ahead adder in order to attain higher speed for IEEE754 format. In microprocessors, industrial area and applications involving arithmetic operation, the operations are expected to be carried out at a faster speed. Dadda algorithm is thus used to increase the speed of operation of multiplier. The multiplier is designed using ripple carry adder and carry look ahead adder and the results are compared. The basic modules are designed using Verilog language and implemented using Xilinx 12.4 ISE software and the same is simulated using isim simulator. Also the design is implemented using Cadence NC simulator to obtain the statistics for both adder cases for higher speed criteria.**

I . INTRODUCTION

In signal processing operation, multiplication is the arithmetic operation which is used widely. In order to increase the speed of multiplication, the number of steps in partial product reduction stage is reduced by using Dadda algorithm. The multiplier is first designed using Ripple carry adder. The speed of multiplier can be increased by using faster adders. Thus in the final stage, ripple carry adder is replaced by carry look ahead adder and the results are compared. Floating point multiplication involves multiplying two numbers that are in IEEE754 format. There are two types of floating point numbers namely, single precision (32 bit) and double precision (64 bit) floating point numbers. In this paper we have considered single precision floating point numbers. Single precision number has 32 bits comprising of three parts namely, sign, exponent and mantissa. Sign bit is 1 bit wide and is '0' for positive numbers and '1' for negative numbers. Exponent is 8 bit wide and mantissa is 23 bit wide. This can be represented as shown.

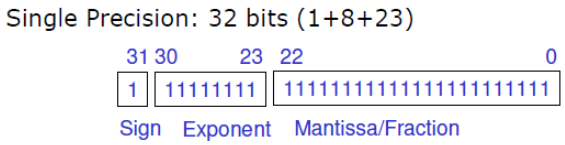

Fig.1 Representation of single precision floating point number

II .FLOATING POINT MULTIPLICATION OPERATION

Many algorithms are used to design floating point numbers. The important goal is to obtain high speed, less delay, minimize area, minimize power and obtain better performance. The important steps involved in multiplying two floating point numbers are as follows.

1) Multiplying the mantissa of two numbers i.e., significand of two numbers.
2) Placing the decimal point after multiplication in the result.
3) Adding the exponents of two numbers.
4) Obtaining sign of final result by EXORing the sign bits of two numbers.
5) Normalizing the result to get 1 at the MSB of significand's multiplication result.

Thus, the two single precision floating point numbers are multiplied and the final result is also kept in single precision IEEE754 format.

III . BLOCK DIAGRAM OF FLOATING POINT MULTIPLIER

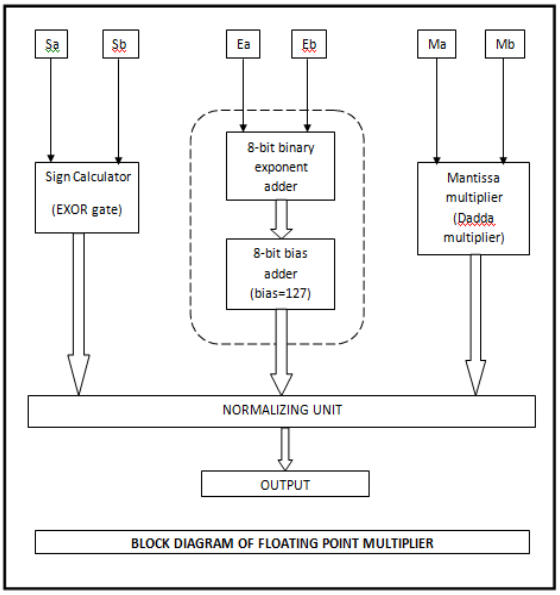

Fig.2 Block diagram of floating point multiplier

The block diagram of floating point multiplier is as shown above. The main sub-blocks includes sign calculator, 8 bit binary exponent adder, 8 bit bias adder, mantissa multiplier and normalizing unit.

A) Sign calculator: This block is used to find out sign of the result. The sign of two numbers is calculated by EXORing the sign bit of two numbers. The result is positive if both the sign bits are positive or negative and is negative if either of the sign bits are negative.

B) Exponent calculator: This block is used to find the exponent of the result. The exponent of two numbers in binary format are added together and bias is added to the result to convert it to floating point number format. The bias for single precision floating point number is 127.

C) Mantissa multiplier: This block is used to find the multiplication of of two 23 bit mantissa. The multiplication is done using Dadda algorithm to obtain the final product. Dadda algorithm is the fastest way of multiplying two numbers with reduced number of stages for partial product reduction The dot matrix representation of 8x8 Dadda algorithm is as shown.

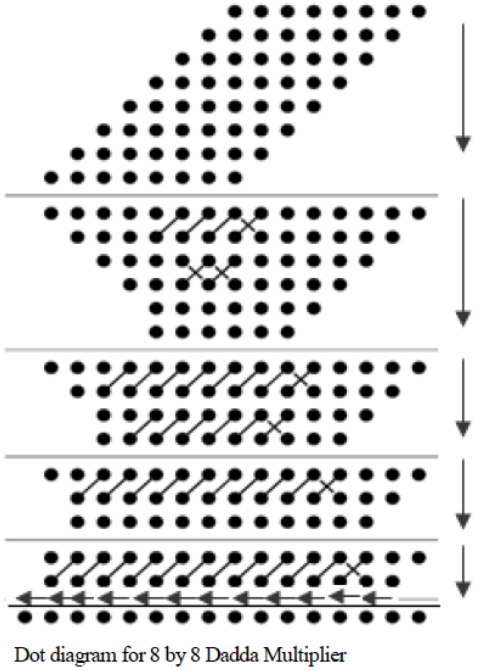

Fig.3 Dot diagram for 8x8 Dadda multiplier

D) Normalizing unit: Normalized result means they have leading '1' to the left hand side of mantissa multiplication result.

IV . SCHEMATIC OF FLOATING POINT MULTIPLIER

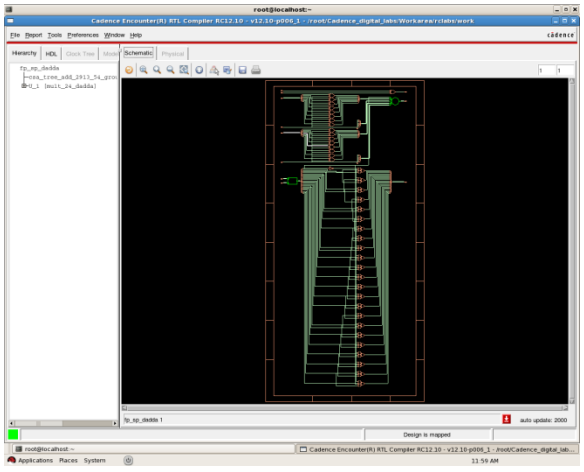

Fig.4 Schematic of Floating point multiplier

ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 7, July 2015

V . SIMULATION RESULTS OBTAINED IN XILINX 12.4 ISE SOFTWARE

A) For ripple carry adder:

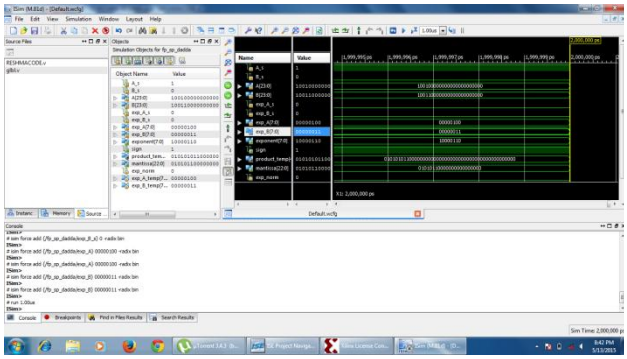

Fig. 5 Simulation results for ripple carry adder

B) For carry look ahead adder:

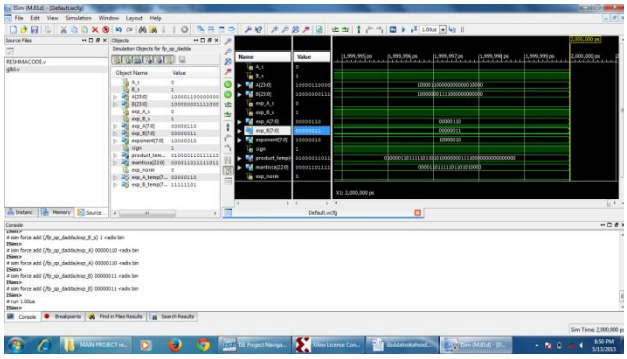

Fig. 6 Simulation results for carry look ahead adder

VI . SIMULATION RESULTS OBTAINED IN CADENCE NC SIMULATOR SOFTWARE:

A) For Ripple carry adder:

```
====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10 - v12.10-
p006_1
  Generated on:           May 19 2015  12:43:04  pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt  1.0
  Operating conditions:   slow  (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================

          Instance              Cells  Cell Area  Net Area  Total Area
Wireload
-------------------------------------------------------------------------
-----------
fp_sp_dadda                      1368      12579         0       12579
<none>  (D)
  U_1                            1290      12095         0       12095
<none>  (D)
```

Fig. 7 Area report for ripple carry adder

```
====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10 - v12.10-
p006_1
  Generated on:           May 19 2015  12:43:41  pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt  1.0
  Operating conditions:   slow  (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================

                                  Leakage     Dynamic      Total
          Instance        Cells  Power(nW)   Power(nW)   Power(nW)
-------------------------------------------------------------------
fp_sp_dadda                1368  13962.800  712269.683  726232.484
  U_1                      1290  13554.776  689691.002  703245.778
```

Fig. 8 Power report for ripple carry adder

```
====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10 - v12.10-
p006_1
  Generated on:           May 19 2015  12:43:19  pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt  1.0
  Operating conditions:   slow  (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================

   Pin          Type        Fanout  Load  Slew  Delay  Arrival
                                    (fF)  (ps)   (ps)    (ps)
 ------------------------------------------------------------
U_1/p[47]
csa_tree_add_2913_54_groupi/in_0
  g474/CI                                            +0    10331
  g474/CO      ADDFX1TH           1   2.5    91   +385    10716  F
  g473/CI                                            +0    10716
  g473/CO      ADDFX1TH           1   2.5    91   +244    10960  F
  g472/CI                                            +0    10960
  g472/CO      ADDFX1TH           1   2.5    90   +244    11203  F
  g471/CI                                            +0    11203
  g471/CO      ADDFX1TH           1   2.5    92   +244    11447  F
  g470/CI                                            +0    11447
  g470/CO      ADDFX1TH           1   2.5    92   +244    11691  F
  g469/CI                                            +0    11691
  g469/CO      ADDFX1TH           1   2.5    92   +244    11935  F
  g468/CI                                            +0    11935
  g468/CO      ADDFX1TH           1   1.7    85   +235    12170  F
  g467/A                                             +0    12170
  g467/Y       XOR2XLTH           1   0.0    81    +86    12256  F
csa_tree_add_2913_54_groupi/out_0[7]
exponent[7]    out port                             +0    12256  F
 ------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : A[1]
End-point    : exponent[7]
```

Fig. 9 Timing report for ripple carry adder

B) For Carry look ahead adder:

```
====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10 - v12.10-
p006_1
  Generated on:           May 22 2015  12:00:59  pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt  1.0
  Operating conditions:   slow  (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================

          Instance              Cells  Cell Area  Net Area  Total Area
Wireload
-------------------------------------------------------------------------
-----------
fp_sp_dadda                      1506      12940         0       12940
<none>  (D)
  U_1                            1428      12455         0       12455
<none>  (D)
```

Fig. 10 Area report for carry look ahead adder

```
╞=====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10  - v12.10-
p006_1
  Generated on:           May 22 2015  12:01:16 pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=====================================================

                                    Leakage     Dynamic      Total
          Instance          Cells  Power(nW)   Power(nW)   Power(nW)
-----------------------------------------------------------------------
fp_sp_dadda                  1506 14092.543  731315.980  745408.523
  U_1                        1428 13684.518  708855.282  722539.800
```

Fig. 11 Power report for carry look ahead adder

```
╞=====================================================
  Generated by:           Encounter(R)  RTL Compiler  RC12.10  - v12.10-
p006_1
  Generated on:           May 22 2015  12:01:32 pm
  Module:                 fp_sp_dadda
  Technology library:     slow_highvt 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=====================================================

     Pin            Type       Fanout  Load  Slew  Delay  Arrival
                                       (fF)  (ps)   (ps)    (ps)
-------------------------------------------------------------------
U_1/p[47]
csa_tree_add_2999_54_groupi/in_0
  g474/CI                                                    +0    7643
  g474/CO          ADDFX1TH        1   2.5    91   +385    8028  F
  g473/CI                                                    +0    8028
  g473/CO          ADDFX1TH        1   2.5    91   +244    8272  F
  g472/CI                                                    +0    8272
  g472/CO          ADDFX1TH        1   2.5    90   +244    8516  F
  g471/CI                                                    +0    8516
  g471/CO          ADDFX1TH        1   2.5    92   +244    8759  F
  g470/CI                                                    +0    8759
  g470/CO          ADDFX1TH        1   2.5    92   +244    9003  F
  g469/CI                                                    +0    9003
  g469/CO          ADDFX1TH        1   2.5    92   +244    9247  F
  g468/CI                                                    +0    9247
  g468/CO          ADDFX1TH        1   1.7    85   +235    9482  F
  g467/A                                                     +0    9482
  g467/Y           XOR2XLTH        1   0.0    81    +86    9568  F
csa_tree_add_2999_54_groupi/out_0[7]
exponent[7]        out port                                 +0    9568  F
-------------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : A[5]
End-point    : exponent[7]
```

Fig. 12 Timing report for carry look ahead adder

VII . COMPARISON RESULTS FOR RIPPLE CARRY ADDER AND CARRY LOOK AHEAD ADDER:

DADDA MULTIPLIER:

PARAMETER	RIPPLE CARRY ADDER	CARRY LOOK AHEAD ADDER
AREA	12095	12455
POWER(nW)	703245.778	722539.800
TIMING(DELAY) (ns)	10.331	7.643

FLOATING POINT MULTIPLIER:

PARAMETER	RIPPLE CARRY ADDER	CARRY LOOK AHEAD ADDER
AREA	12579	12940
POWER(nW)	726232.484	745408.523
TIMING(DELAY) (ns)	12.256	9.568

Fig. 13 Comparison between floating point multiplier and Dadda multiplier using ripple carry adder and carry look ahead adder

VIII . CONCLUSION AND FUTURE WORK

In this paper, a high speed floating point multiplier is presented that is compatible along IEEE 754 form for single precision binary floating point number format. Thus, wide ranges of numbers are covered due to the usage of FPM. To reduce the delay, Dadda algorithm is used and also faster adder i.e., carry look ahead adder is utilized. Thus, speed of multiplier increases but the area is also slightly increased. The Dadda multiplier is designed using RCA and CLA can be shown that CLA exhibits lesser delay than RCA and is faster in operation.

As the hardware is required for calculating the propagate and generate signals, the area of the entire circuit is increased. Thus, in future we can design the circuit that is optimized for area. Also the power consumption is slightly increased due to large area. This can also be rectified by designing an optimized circuit. Also the design can be implemented on FPGA module and the results can be verified on the hardware as well. These are the future works that can be carried out for further efficient multiplier circuit.

REFERENCES

1]. Supriya Sarkar, Sanghita Deb, G. Dilip, “A New Architecture for Signed and Unsigned Multiplier by Using Radix-4 Process”, International Journal of Emerging Science and Engineering ISSN:2319-6378, Volume-2, Issue-1,November 2013.

2]. Brian Hickmann, Andrew Krioukov and Michael Schulte, “A Parallel IEEE P754 Decimal Floating-Point Multiplier”, IEEE-2007.

3]. Mohamed Al-Ashrafy, Ashraf Salem, Wagdy Anis, :An Efficient Implementation of Floating Point Multiplier”, IEEE-2011.

4]. Vinod Buddhe, Prasanna Palsodkar, Prachi Palsodakar, “Design and Verification of Dadda Algorithm based Binary Floating point Multiplier”, International Conference on Communication and Signal Processing, April 3-5, 2014,India.

5]. T.Arunachalam and S.Kirubaveni, ”Analysis of High Speed Multipliers”, International Conference on Communication and Signal Processing, April 3-5, 2013, India.

6]. V.M.Dhivya, A.Sridevi,A.Ahilan, “A High Speed Area Efficient FIR Filter Using Floating Point Dadda Algorithm”, International Conference on Communication and Signal Processing, April 3-5,2014, India.

Author’s Details

1] Reshmashree.M.S, Mtech, Department of VLSI Design and Embedded Systems, Dr. AIT, Bengaluru.

2] Meenakshi L Rathod, Assistant Professor, Department of Electronics and communication engineering, Dr. AIT, Bengaluru.