Efficient Floating Point Multiplier Architectures On FPGA

PERLA SWATHI[1], KUMPATI NAGALATHA [2]
[1]PG Scholar, ECE Department, ALTS, Anantapur
[2] Assistant Professor, ECE Department, ALTS , Anantapur

ABSTRACT: This paper presents FPGA based equipment structures for skimming point (FP) multipliers. The proposed multiplier models are gone for single exactness (SP),double accuracy (DP), twofold broadened accuracy (DEP)and fourfold exactness (QP) execution. This paper pursues the standard computational stream for FP increase. The mantissa duplications, the most perplexing unit of the FP augmentation, are manufactured utilizing productive utilization of Karatsuba procedure coordinated with the upgraded utilized of in-built25x18 DSP48E squares accessible on the Xilinx Virtex-5 forward FPGA gadgets. It additionally joined with the other techniques(radix-4 corner encoding for little multipliers, halfway items decrease utilizing 4:2, 3:2, 2:2 counters; pressure of multi operands adders) utilized at spots, to enhance the outline. The proposed models out-plays out the accessible condition of - theart , and utilized just 1-DSP48, 3 DSP-48, 6 DSP48 and 18 DSP48for SP, DP, DEP, and QP multipliers separately. Watchwords Floating Point Arithmetic, Multipliers, Digital Arithmetic, FPGA, DSP48E.

INTRODUCTION

In spite of the fact that PC number juggling is some of the time saw as a specific piece of CPU configuration, still the discrete part outlining is likewise a critical viewpoint. A colossal assortment of calculations have been proposed for use in drifting point frameworks. Genuine usage are normally founded on refinements and varieties of the couple of essential calculations displayed here. Notwithstanding picking calculations for expansion, subtraction, increase, and division, the PC designer must settle on different decisions.

Our discourse of coasting point will center solely around the IEEE drifting point standard (IEEE 754) in view of its quickly expanding acknowledgment. Albeit skimming point number-crunching includes controlling types and moving divisions, the greater part of the time in gliding point tasks is spent working on portions utilizing whole number calculations. Hence, after our dialog of drifting point, we will investigate proficient calculations and designs.

Numerous applications require numbers that aren't whole numbers. There are various ways that non-whole numbers can be spoken to. Including two such numbers should be possible with a whole number include, though augmentation requires some additional moving. There is different approaches to speak to the number frameworks. In any case, just a single non-whole number portrayal has increased broad utilize, and that is skimming point. Current computerized flag preparing (DSP) frameworks are making the progress from settled point number juggling (utilized at first on account of its straightforwardness) to floating point math.

The last has a few favorable circumstances including the opportunity from flood and sub-current and simplicity of interfacing to whatever remains of the framework. To enhance the execution of coasting point math, a few combined gliding point tasks have been presented: Fused Multiply– Add (FMA) utilized Add– Subtract, and Fused Two-Term Dot-Product. The combined drifting point activities enhance the execution, as well as diminish the territory and power utilization contrasted with discrete coasting point usage

This plan presents enhanced engineering outlines and usage for an intertwined coasting point add– subtract unit. Numerous DSP applications, for example, quick Fourier Transform (FFT) and Discrete Cosine Transform (DCT) butterfly tasks can profit by the melded drifting point add–

subtract unit. In this manner, he enhanced combined drifting point add– subtract unit will add to the cutting edge skimming point number juggling and DSP application improvement. The proposed combined coasting point add– subtract unit takes two standardized drifting point operands and produces their whole and contrast at the same time. It underpins each of the five adjusting modes determined in IEEE-754 Standard. A few strategies are connected to accomplish low region, low power utilization and fast:

1) Instead of executing two indistinguishable skimming point adders, the intertwined gliding point add– subtract unit shares the basic rationale to produce the aggregate and contrast all the while. In this way, it spares a great part of the territory and power utilization contrasted with a discrete coasting point add– subtract unit.

2) A double way calculation can be connected to build speed. The double way rationale comprises of a far way and a nearby way. In the far way, the expansion, subtraction and adjusting rationale are performed in parallel. By adjusting the significands to the insignificant number of bits, the expansion, subtraction and adjusting rationale are disentangled. There are three cases for the nearby way relying upon the distinction of the examples. For each case, expansion, subtraction and driving zero expectation (LZA) are performed in parallel and adjusting isn't required. Subsequently, the double way configuration lessens the inertness of the basic way.

3) To build the throughput, pipelining can be connected. In view of information stream investigation, the proposed double way configuration is part into two pipeline stages. By appropriately masterminding the parts, latencies of the two pipeline stages are adjusted with the goal that the throughput of the whole plan is expanded. Additionally, it decreases the idleness by improving the control signals.

VEDIC MULTIPLIER:

The design of multiplier depends on the opposite and askew calculation. The engineering of multiplier is shown with two 4-bit numbers; the multiplier and multiplicand, each are assembled as 4-bit numbers so it disintegrates into 4x4 improvement modules. After the deterioration, vertical and transversely calculation is helpful to complete the duplication on initial 4x4 increase modules. The results of initial 4x4 increase module are used in the wake of getting the halfway item bits parallel from the consequent module to create the last 16-bit item.

Henceforth any complex NXN increase can be productively executed all through utilizing minimal 4x4 multiplier utilizing the anticipated design where N is a different of 4, for example, 8 , 16, ...2N. Henceforth ingenious augmentation calculation achievement with modest numbers such the equivalent as 4-bits, can be effectively complete and implanted for executing efficient NXN increase strategy. The arranged strategy for Urdhva Triyag bhyam can be produced for twofold framework similarly as decimal framework. This Sutra has been generally utilized for the duplication of two numbers in the decimal number framework.

In this paper, we apply a similar plan to the double number framework to make it perfect with the advanced equipment. Give us initial a chance to show this Sutra with the assistance of a precedent in which two decimal numbers are increased. Line graph for the increase of two numbers.

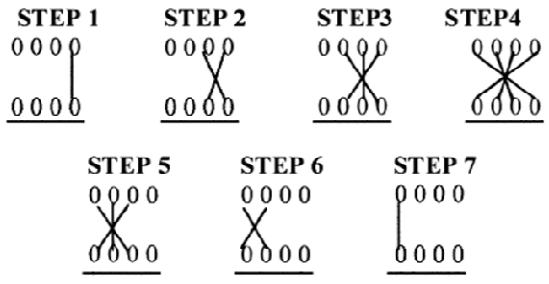

Fig1: Example of 4x4 multiplier

The 4 x 4 duplication has been done in a solitary line in Urdhva technique, though in move and include strategy (Conventional) four fractional items must be added to get the outcome. This infers the expansion in speed. Urdhva Tiryakbhyam (Vertically and Crosswise), manages the duplication of numbers [8], [9], [12]. This Sutra has been customarily utilized for the increase of two numbers in the decimal number framework. In this paper, we apply a similar plan to the parallel number framework to make it perfect with the advanced equipment. Give us initial a chance to outline this Sutra with the assistance of a precedent in which two decimal numbers are duplicated. Line outline for the duplication of two numbers.

The digits on the two finishes of the line are duplicated and the outcome is included with the past convey. At the point when there are more lines in a single step, every one of the outcomes are added to the past convey. The slightest noteworthy digit of the number along these lines got goes about as one of the outcome digits and the rest go about as the convey for the subsequent stage. At first the convey is taken to be zero.

We currently stretch out this Vedic increase calculation to parallel number framework with the primer learning that the duplication of two bits aO and bO is only an AND activity and can be executed utilizing basic AND door. To show this duplication conspire in parallel number framework, let us consider the augmentation of two paired numbers a3a2alaO and b3b2blbO. As the aftereffect of this duplication would be in excess of 4 bits, we express it as ... r3r2rlrO.

Line outline for duplication of two 4-bit numbers is appeared in Fig. 1 which is only the mapping of the Fig. 2 in double framework. Fractional items are computed in parallel and subsequently the defer included is only the time it takes for the flag to engender through the entryways.

Floating POINT MULTIPLICATION:

Floating Point Multiplication Algorithm:

Drifting point augmentation process can be partitioned in to four units; mantissa estimation unit, type computation unit, sign count unit and normaliser unit . Standardized gliding point number have the type of

$$Z = (-1^{s}) * 2^{(E-bias)} * (1.M).$$

.

To perform multiplication of two single precision floating point numbers following steps are followed:

Step1: Multiplication of significand i.e. $(1.M_1 * 1.M_2)$.
Step2:Placing the decimal point in the result.
Step3:Addition of the exponent i.e. $(E_1 + E_2 - bias)$.
Step4:Obtaining the sign bit i.e. $(S_1 \; XOR \; S_2)$.
Step5:Normalizing the result i.e. obtaining 1 at MSB of the results significand.
Step6: Rounding the result to fit in available bits
Step7: Checking for overflow /under flow

MANTISSA MULTIPLICATION:

Overall performance of designed multiplier depends upon the performance of mantissa multiplier unit faster the multiplication unit faster is the overall calculation. Mantissa multiplication unit is designed using Vedic multiplication technique. "Urdhwa-triyagbhyam" sutra is used for multiplication. 3by3 multiplier is used as basic multiplier. Vedic mathematics technique improves the performance of multiplier unit in terms of speed and power.

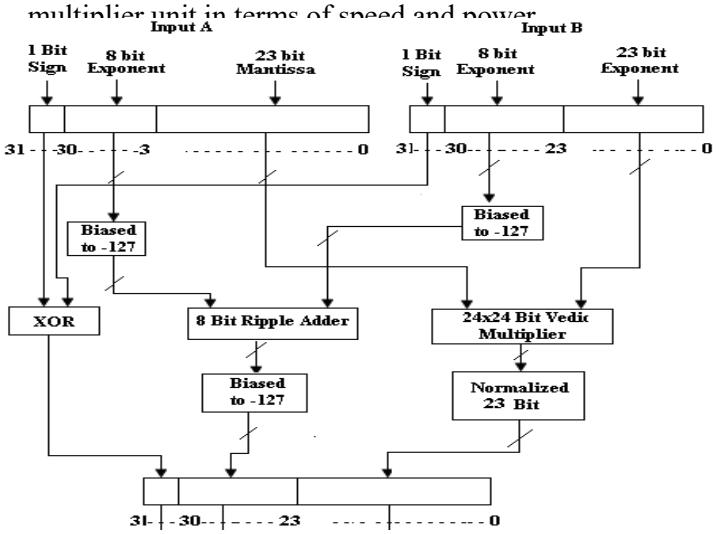

Fig 2: block diagram of 32 bit length floating point multiplication.

THE STEPS FOR PERFORMING BOOTH MULTIPLICATION ARE AS FOLLOWS:

Give the multiplicand a chance to be 'B' and multiplier be 'Q'.

Expect at first estimation of 'An' and 'Q-1' is zero.

The principle step is to check last two bits.

There will be cycles as indicated by the quantity of multiplier.

For instance, if the multiplier is of 2-bit then 2 Iterations will be done, for 4-bit multiplier 4 cycles are done, et cetera.

Presently, the calculation begins, first the last two digits are checked and if the two bits are "00" or "11" the main Arithmetic Right Shift is finished.

Also, if the last two bits are "01", at that point An is included with B, and result is put away into A.

On the off chance that the last two bits are "10", at that point An is subtracted from B, and result is put away into A.

MANTISSA CALCULATION UNIT:

The execution of Mantissa estimation Unit commands by and large execution of the Floating Point Multiplier. The Vedic Multiplication system is decided for the usage of this unit. This method gives result as far as speed and control and for single accuracy multiplier 3*3 piece multiplier outlined as a fundamental multiplier.

Example SUBTRACTOR:

The example subtractor is utilized for type correlation and can be executed as a viper. A settled point viper has been widely considered and can be utilized in the example snake. Adders, for example, bring down part-OR adders (LOA), rough mirror adders, inexact XOR/XNOR-based adders, and equivalent division adders can be found in the writing. For a quick FP viper, a changed LOA snake is utilized, in light of the fact that it essentially decreases the basic way by disregarding the lower convey bits.

NORMALIZER:

The aftereffect of the significand augmentation (middle item) must be standardized to have a main „1‟ just to one side of the decimal point (i.e. in the bit 46 in the middle of the road item). Since the information sources are standardized numbers then the middle of the road item has the main one at bit 46 or 47. On the off chance that the main one is at bit 46 (i.e. to one side of the decimal point) at that point the middle of the road item is as of now a standardized number and no move is required. In the event that the main one is at bit 47 then the transitional item is moved to one side and the type is augmented by 1.

Flood/UNDERFLOW DETECTION:

Flood/sub-current implies that the result‟s example is too vast/little to be spoken to in the type field. The example of the outcome must be 8 bits in size, and should be somewhere in the range of 1 and 254 generally the esteem isn't a standardized one .A flood may happen while including the two types or amid standardization. Flood because of type expansion perhaps remunerated amid subtraction of the predisposition; bringing about a typical yield esteem (ordinary activity). An undercurrent may happen while subtracting the inclination to frame the middle type. On the off chance that the middle type < 0 then it‟s a sub-current that can never be redressed; if the moderate example = 0 then it‟s a sub-current that might be repaid amid standardization by adding 1 to it .When a flood happens a flood hail flag goes high and the outcome swings to ±Infinity (sign decided by the indication of the coasting point multiplier inputs). At the point when a sub-current happens a sub-current banner flag goes high and the outcome swings to ±Zero (sign decided by the indication of the skimming point multiplier inputs). Denormalized numbers are motioned to zero with the fitting sign ascertained from the sources of info and an undercurrent hail is raised.

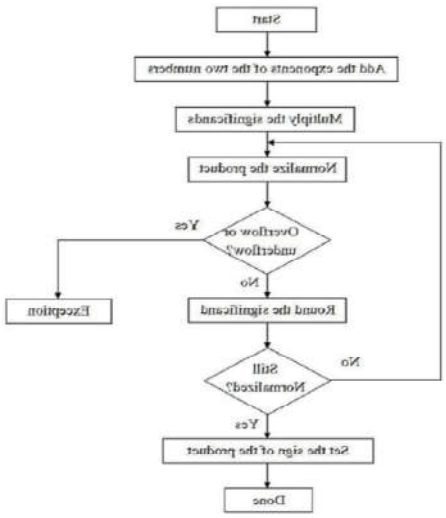

Fig 3: Flowchart of floating point multiplication.

SIMULATION RESULTS:

Below figure shows out put wave form of fused floating point add-subtract unit. Any number can be converted into floating point number that is having 32-bit format. Now we can give a (32-bit), and b (32-bit) as an input then we gets as 32-bit output.

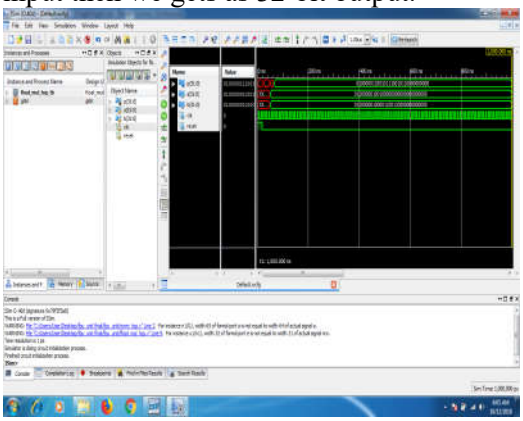

Fig 4: simulation results

SYNTHESIS RESULTS:

In the actualized framework, the quantity of cuts look into table is 863 and the quantities of completely utilized LUT-FF sets are 32 sets which will be diminished when contrasted with the current framework. it will used to diminish the circulate rationale in the executed framework. Thus the zone will be diminished because of utilizing of DSP48E cut.

Delay:

In the current framework the information way delay is 1.250 ns yet in the proposed framework the way delay is 0.817 ns . consequently the way delay is lessened when the postpone diminishes the working recurrence will increments.

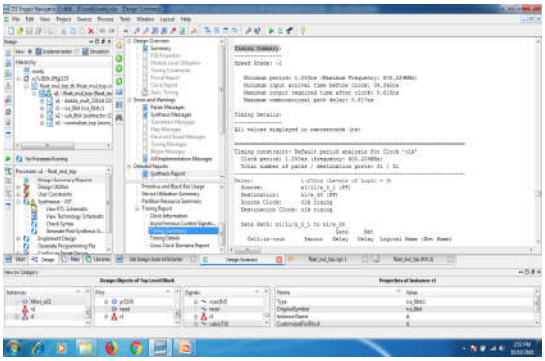

Fig 5: delay

CONCLUSION AND FUTURE SCOPE

CONCLUSION:

This undertaking has displayed an arrangement of skimming point multiplier structures on Xilinx FPGA gadget for Virtex-5 forward arrangement. The engineering for single accuracy, twofold exactness, twofold broadened exactness and fourfold exactness multiplier are proposed utilizing proficient utilization of in-assembled DSP48E texture in blend with Karatsuba procedure of duplication. Different procedures, similar to radix-4 corner encoding for little multipliers, decrease of fractional items utilizing 4:2, 3:2, 2:2 counters, decrease of multi operands adders, and so on are additionally utilized at spots, to enhance the outline. The recommendations prompt a noteworthy change in DSP48E use, while at cost of more LUTs, be that as it may, has better in general equal LUTs necessities. Additionally, as appeared in results and correlation segment, the proposed models are better in equipment usage towards higher exactness executions, ie for twofold accuracy and past structures. It comprises of

4 modules, i.e. Corner Radix-4 mantissa multiplier, normalizer, type viper and the underwriter. Corner radix-4 duplication is one of the appropriate calculation to be utilized to plan the rapid 24-bit mantissas multiplier since this calculation is considerably less difficult than the intricate Wallace Tree multiplier, along these lines less door deferral and ready to perform such complex increase quicker. Also, Booth radix-4 execution is multiplied contrasted with Booth radix-2 that permits fast augmentation can be accomplished.

FUTURESCOPE:

In this manner the enhanced melded gliding point add– subtract unit will add to the cutting edge skimming point number juggling and DSP application improvement. The proposed combined drifting point add–subtract unit takes two standardized skimming point operands and produces their total and contrast all the while. This outline unit is likewise utilized for improvement of multiplexer. This is additionally helpful for rapid plan units.

REFERENCES:

1. *IEEE Standard for Floating-Point Arithmetic*, ANSI/IEEE Standard 754-2008, New York: IEEE, Inc., Aug. 29, 2008.
2. R. K. Montoye, E. Hokenek, and S. L. Runyon, "Design of the IBM RISC system/6000 floating-point execution unit," *IBM J. Res. Develop.*, vol. 34, pp. 59–70, 1990.
3. E. Hokenek, R. K. Montoye, and P. W. Cook, "Second-generation RISC floating point with multiply-add fused," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1207–1213, Oct. 1990.
4. T. Lang and J. D. Bruguera, "Floating-point fused multiply-add with reduced latency," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 988–1003, Aug. 2004.
5. H. H. Saleh and E. E. Swartzlander, Jr., "A floating-point fused add–subtract unit," in *Proc. 51st IEEE Midwest Symp. Circuits Syst.*, 2008, pp. 519–522.
6. H. H. Saleh and E. E. Swartzlander, Jr., "A floating-point fused dotproduct unit," in *Proc. IEEE Int. Conf. Comput. Design*, 2008, pp. 427–431.
7. E. E. Swartzlander, Jr. and H. H. Saleh, "FFT implementation with fused floating-point operations," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 284–288, Feb. 2012.
8. E. E. Swartzlander, Jr. and H. H. Saleh, "Fused floating-point arithmetic for DSP," in *Proc. 42nd Asilomar Conf. Signals, Syst., Comput.*, 2008.
9. M. P. Farmwald, "On the design of high performance digital arithmetic units," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 1981.
10. N. Quach and M. Flynn, "Design and implementation of the SNAP floating-point adder," Tech. Rep. CSL-TR-91-501, 1991, Stanford Univ..
11. E. Hokenek and R. Montoye, "Leading-zero anticipator (LZA) in the IBMRISC system/6000 floating-point execution unit," *IBMJ. Res. Develop.*, vol. 34, pp. 71–77, 1990.
12. A. Beaumont-Smith, N. Burgess, S. Lefrere, and C. Lim, "Reduced latency IEEE floating-point standard adder architectures," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, 1999, pp. 35–43.