| | |
|---|---|
| Title | Architecture for dual-mode quadruple precision floating point adder |
| Author(s) | Jaiswal, MK; Bogaraju, SV; So, HKH |
| Citation | The 2015 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Montpellier, France, 8-10 July 2015. In Conference Proceedings, 2015, p. 249-254 |
| Issued Date | 2015 |
| URL | http://hdl.handle.net/10722/214074 |
| Rights | IEEE Computer Society Annual Symposium on VLSI. Copyright © IEEE Computer Society. |

# Architecture for Dual-Mode Quadruple Precision Floating Point Adder

Manish Kumar Jaiswal, B. Sharat Chandra Varma, and Hayden K.-H So
*Department of EEE, The University of Hong Kong, Hong Kong*
*Email: {manishkj, varma, hso}@eee.hku.hk*

***Abstract*—This paper presents a configurable dual-mode architecture for floating point (F.P.) adder. The architecture (named as QPdDP) works in dual-mode which can operates either for quadruple precision or dual (two-parallel) double precision. The architecture follows the standard state-of-the-art flow for floating point adder. It is aimed for the computation of normal as well as sub-normal operands, along with the support for the exceptional case handling. The key sub-components in the architecture are re-designed & optimized for on-the-fly dual-mode processing, which enables efficient resource sharing for dual precision operands. The data-path is optimized for minimal multiplexing circuitry overhead. The presented dual-mode architecture provide SIMD support for double precision operands, along with high (quadruple) precision support. The proposed architecture is synthesized using UMC $90nm$ technology ASIC implementation. It is compared with the best available literature works, and have shown better design metrics in terms of area, period and $area \times period$, along with more computational support.**

***Keywords*-Floating Point Addition, Configurable Architecture, Dual-Mode Arithmetic, ASIC, Digital Arithmetic.**

## I. Introduction

Floating point (FP) number system [1], due to its wide dynamic range, is a common choice for a large set of scientific, engineering and numerical processing computations. Generally, the performance of these computations greatly depends on the underlying floating point arithmetic processing unit. Furthermore, the availability for double precision (DP) computation is not enough and the demand for high precision arithmetic is increasing in many application areas [2], [3].

The contemporary processing units achieve high performance requirement by using multiple units of single precision and double precision arithmetic hardware. Like, the Synergistic Processing Element (SPE) in Cell-BE processor [4] contains a vector array of 4 single precision and an array of 2 double precision. The ARM VFU co-processor (VFU9-S) [5] provides a vector array of 16 single precision FP units and 8 double precision vector array. Similarly, it can be seen in recent Intel Xeon Phi$^{TM}$ and Nvidia Kepler$^{TM}$ GK110 [6]. In general, these computing systems contain separate units/arrays for single precision and double precision computations. However, if an unified and configurable unit can support a double precision with dual/two-parallel single precision (DPdSP) arithmetic, or quadruple precision (QP) with dual/two-parallel double precision (QPdDP) arithmetic, it can save a large silicon area in the above computing machines. In view of above, this paper is aimed towards the design of a configurable dual-mode floating adder/subtractor architecture, with high precision support.

Some literature [7], [8], [9] have proposed dual-mode architectures for adder. These works have tried to improve the resource utilization for the hardware with multi-precision computational support. However, the overhead of extra hardware, and un-optimized data-path and resource sharing lead to large overhead of area and period metrics. Furthermore, they have limited support only for normal operands. The dual-mode adder architectures of [7], [8] used a large number of multiplexers (to support dual mode) at various level of architecture, and have less tuned data path for dual mode operation. Further the extra use of resources (like more adders/subtractors for exponent & mantissa, relatively larger dual shifters, extra mantissa normalizing shifters for dual mode support) made their area & period overhead larger. Some recent literature [10], [11] have also worked on the dual-mode architectures, but with low precision support.

This paper proposes an architecture for dual-mode QPdDP (quadruple precision with dual/two-parallel double precision) adder/subtractor arithmetic. The computational sub-components are designed for configurable dual-mode support. The data-path is tuned for better resource sharing and to minimize the multiplexing circuitry. The proposed architecture provides full support for normal as well as sub-normal operands computation, exceptional case handling, and with round-to-nearest rounding method. Other rounding methods can also be easily included. A pipelined architecture is designed and synthesized using $90nm$ standard cell based ASIC implementation. The proposed architecture is compared with the best available literature.

## II. Proposed Architecture of Quadruple Precision / Dual (Two-Parallel) Double Precision (QPdDP) Adder/Subtractor

The present work on the dual-mode floating point adder architecture follows the basic single-path algorithm for this computation. A floating point arithmetic computation involves computing separately the sign, exponent and mantissa part of the operands, and later combine them after rounding and normalization [1]. The standard format for floating point numbers are as follows:

$$SP : \overbrace{1-bit}^{\text{Sign}}\ \overbrace{8-bit}^{\text{Exponent}}\ \overbrace{23-bit}^{\text{Mantissa}}$$

978-1-4799-8719-1/15 $31.00 © 2015 IEEE
DOI 10.1109/ISVLSI.2015.70

$$DP : \overbrace{1-bit}^{\text{Sign}}\overbrace{11-bit}^{\text{Exponent}}\overbrace{52-bit}^{\text{Mantissa}}$$

$$QP : \overbrace{1-bit}^{\text{Sign}}\overbrace{15-bit}^{\text{Exponent}}\overbrace{112-bit}^{\text{Mantissa}}$$

A basic state-of-the-art computational flow of the floating point adder is shown in the Algorithm 1. Here, steps 6-7 and step-22 are require for sub-normal processing. In this work, each steps of the flow are constructed for the support of the dual-mode operation with resource sharing and tuned data-path with minimum multiplexing circuitry.

**Algorithm 1** F.P. Adder Computational Flow [1]

1: ($IN1, IN2$) Input Operands;
2: **Data Extraction & Exceptional Check-up:**
3: {S1(Sign1), E1(Exponent1), M1(Mantissa1)} $\leftarrow IN1$
4: {S2, E2, M2} $\leftarrow IN2$
5: Check for INFINITY, NAN
6: Check for SUB-NORMALs
7: Update Exponents & Mantissa's MSB for SUB-NORMALs
8: **COMPARE, SWAP & Dynamic Right SHIFT:**
9: $IN1\_gt\_IN2 \leftarrow \{E1, M1\} \geq \{E2, M2\}$
10: Large_E,M $\leftarrow IN1\_gt\_IN2$ ? E1,M1 : E2,M2
11: Small_E,M $\leftarrow IN1\_gt\_IN2$ ? E2,M2 : E1,M1
12: Right_Shift $\leftarrow$ Large_E - Small_E
13: Small_M $\leftarrow$ Small_M >> Right_Shift
14: **Mantissa Computation:**
15: $OP \leftarrow \overline{S1 \oplus S2}$
16: **if** $OP == 1$ **then**
17: Add_M $\leftarrow$ Large_M + Small_M
18: **else**
19: Add_M $\leftarrow$ Large_M - Small_M
20: **Leading-One-Detection & Dynamic Left SHIFT:**
21: Left_Shift $\leftarrow$ LOD(Add_M)
22: Left_Shift $\leftarrow$ Adjustment for SUB-NORMAL or Underflow
23: Add_M $\leftarrow$ Add_M << Left_Shift
24: **Normalization & Rounding:**
25: Mantissa Normalization & Compute Rounding ULP based on Guard, Round & Sticky Bit
26: Add_M $\leftarrow$ Add_M + ULP
27: Large_E $\leftarrow$ Large_E + Add_M[MSB] - Left_Shift
28: **Finalizing Output:**
29: Update Exponent & Mantissa for Exceptional Cases
30: Determine Final Output

The architecture for proposed dual-mode QPdDP adder is shown in Fig. 1. The input/output register for this architecture is assumed as shown in Fig. 2. The two 128-bit input operands, contain either 1 set of quadruple precision or 2 sets of double precision operands. Based on the mode deciding control signal ($qp\_dp$), the dual-mode architecture switched to either quadruple precision or dual (two-parallel) double precision computation mode ($qp\_dp$: 1 $\rightarrow$ QP Mode, $qp\_dp$: 0 $\rightarrow$ Dual DP Mode). All the computational steps in QPdDP dual mode adder are discussed below in detail.

The data-extraction, sub-normal and exceptional handling are shown in the Fig. 3. Based on the precision format, the sign, exponent and mantissa parts of the operands are extracted for both, the quadruple precision and double precision.

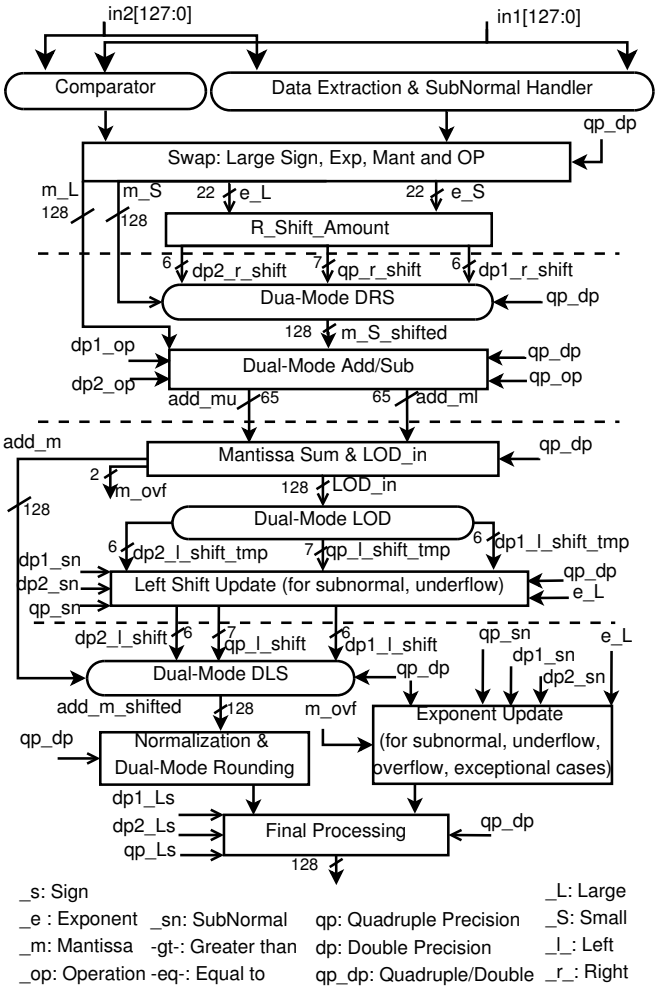


Figure 1: QPdDP Adder Architecture.

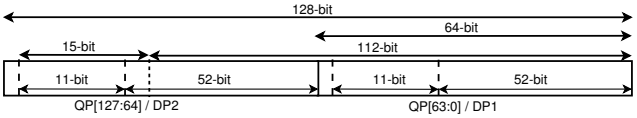


Figure 2: QPdDP Adder: Input / Output Register Format.

dp1_sn1=~|in1[62:52]
dp1_sn2=~|in2[62:52]
dp1_sn=dp1_sn1 & dp1_sn2
dp1_e1={in1[62:53],in1[52] | dp1_sn1}
dp1_e2={in2[62:53],in2[52] | dp1_sn2}
dp2_e1={in1[126:117],in1[116] | dp2_sn1}
dp2_e2={in2[126:117],in2[116] | dp2_sn2}
qp_e1={in1[126:113],in1[112] | qp_sn1}
qp_e2={in2[126:113],in2[112] | qp_sn2}
dp2_sn1=~|in1[126:116]
dp2_sn2=~|in2[126:116]
dp2_sn=dp1_sn1 & dp2_sn2
dp1_m1={~dp1_sn1,in1[51:0]}
dp1_m2={~dp1_sn2,in2[51:0]}
dp2_m1={~dp2_sn1,in1[115:64]}
dp2_m2={~dp2_sn2,in2[115:64]}
qp_m1={~qp_sn1,in1[111:0]}
qp_m2={~qp_sn2,in2[111:0]}
qp_sn1=~|in1[115:112] & dp2_sn1
qp_sn2=~|in2[115:112] & dp2_sn2
qp_sn = qp_sn1 & qp_sn2

Figure 3: QPdDP Adder: Data Extraction and Subnormal Handler.

As shown in Fig. 2 that the exponent portion of QP and

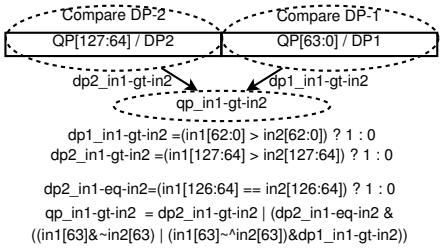


Figure 4: QPdDP Adder: Comparator.

second DP (DP-2) operand are overlapped.

$$\overbrace{\underbrace{xxxxxxxxxx}_{DP-2\ Exponent}\ xxxx}^{QP\ Exponent}$$

This scenario is used to share the resources related to sub-normal, `infinity`, and `NaN` checks computations of QP and second DP operands (the checks of sub-normal is shown in the Fig. 3, similarly the checks for `infinity` and `NaN` are handled). After these exceptional checks the exponent and mantissa are updated accordingly. In comparison to only QP computation, this unit requires extra related resources for first DP (DP-1) operands.

The dual-mode comparator unit for dual-mode QPdDP adder is shown in Fig. 4. The comparator unit determines which operand is large and which one is small. This unit is shared among the QP and both DP operands. It comprises of two comparator units for both DPs operands, which generates their corresponding comparison results. These DP results are further combined to form QP comparison. In terms of resources, this comparator unit requires similar resources as needed in only QP comparator, and there is no area overhead in this unit.

The next computational unit in this architecture is the Dual-Mode SWAP, which generates large sign (effectively output sign-bit), small & large exponents, small & large mantissas and effective operations (to be performed between large and small mantissas). This computational unit is shown in Fig. 5. For SWAP, in general to handle both DPs and QP, it needs four 11-bit (for both DP exponents), two 15-bit (for QP exponents), four 53-bit (for both DP mantissas) and two 113-bit (for QP mantissa) SWAP components for all the computations of this section. However, to minimize the swapping overhead, the unified exponents, mantissas and greater-than control signals are generated, by multiplexing either of the quadruple precision or both double precision operands (as shown in Fig. 5). This is an important step included in the dual-mode QPdDP architectural flow, which helps to design a tuned data-path computation in later stages, with reduced multiplexing circuitry. Using these unified exponents, mantissas and greater-than control signals, it requires only four 11-bit (for exponents) and four 64-bit (for mantissas) SWAP circuitry for entire processing. Effectively,

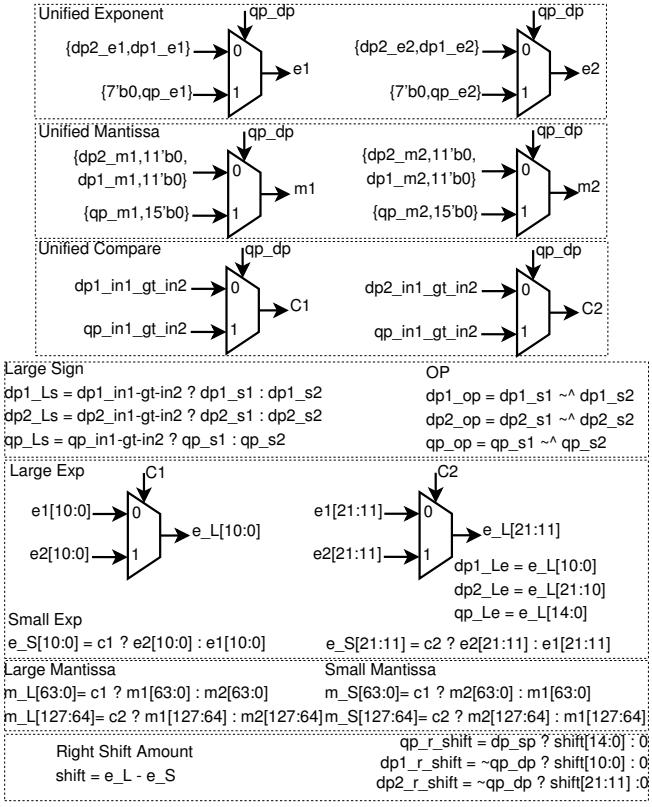


Figure 5: QPdDP Adder: SWAP - Large Sign, Exponent, Mantissa and OPERATION; Right Shift Amount.

it needs SWAP components slightly more than it requires for only QP (only QP requires two 15-bit SWAP for exponents and two 113-bit SWAP for mantissas), along with extra multiplexing circuitry needed to generate unified signals, however, facilitates the tuned data-path processing. Further, among extra appended LSB ZEROs in mantissa multiplexing (for m1 and m2), 3-bit are for Guard, Round and Sticky bit computations in rounding phase, and remaining can provide extended precision support to the operands.

The *m_L* contains mantissa of either large QP operand or both of large DP operands. Similarly, *m_S* contains small mantissas. Likewise, *e_L* contains large exponent, and *e_S* contains small exponents, either of QP or both DP operands.

Now, the small mantissa needs right shifting by the difference of large and small exponents. The right shift amount for small mantissas are determined using the component shown in Fig. 5. In general, it requires two 11-bit subtractors for both double precision and one 15-bit subtractor for quadruple precision. However, because of effective multiplexing of operands in SWAP section, it needs only one a 22-bit subtractor. A subtraction of unified large exponent (*e_L*) and unified small exponent (*e_S*) will produce right shift amount either for quadruple precision or for both double precision. For right shift amount, compared

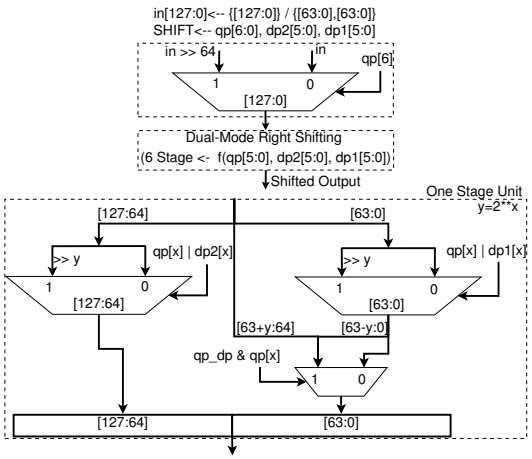


Figure 6: QPdDP Dual Mode Dynamic Right Shifter (DRS).

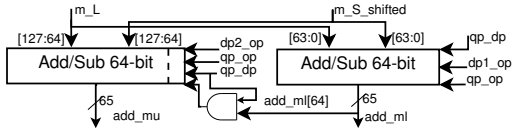


Figure 7: QPdDP Adder: Dual Mode Mantissa Addition/Subtraction.

to only quadruple precision, it requires extra resources for 7-bit subtraction. Other processing in this section are bit-wise operations, and are done separately for all operands.

For right shifting of small mantissas of quadruple and both double precision operands, a dual-mode dynamic right shifter (DRS) is designed. The QPdDP dual-mode dynamic right shifter is shown in Fig. 6, which is used to right-shift the small mantissas of either QP or both DPs. The initial step in it right-shifts the operand by 64-bit in case of QP mode with its true shift bit. The later 6-stages in it works in dual mode, either for QP or for both DPs operands. Each dual-mode stage contains two shifters for each of 64-bit blocks, which right-shifts their inputs corresponding to their shifting bit (either for quadruple or double precision). Each of these stages also include one multiplexer which selects between lower shifting output or their combination with primary input to the stage, based on the mode of the operation.

Further to the right shifting of small mantissas, the core operation of mantissa addition/subtraction fall in the computational flow. The large mantissas and right-shifted small mantissas undergo addition/subtraction based on their effective operation. This computation is performed in dual-mode using two 64-bit integer adder-subtraction unit, which individually works for each double precision, and collectively works for quadruple precision computation (as shown in Fig. 7). This unit generates the lower and upper parts of addition/subtraction separately. This component requires effectively similar resources as present in only QP adder.

The lower and upper mantissa addition/subtraction results

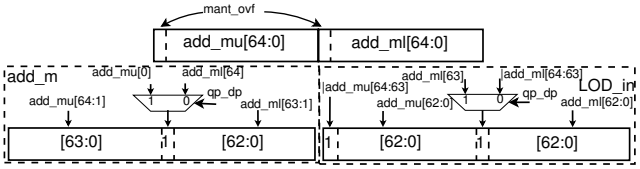


Figure 8: QPdDP Dual Mode Mantissa SUM and LOD_in.

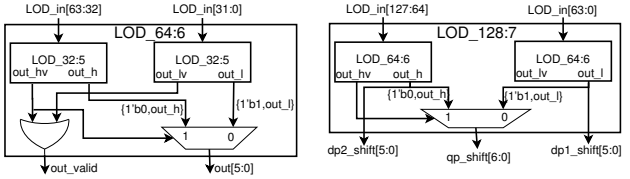


Figure 9: QPdDP Dual Mode Leading-One-Detector.

generated in previous unit combined in "Mantissa SUM and LOD_in unit", to provide the actual sum (either for QP or both DPs), mantissa overflow, and the input for next level unit, leading-one-detector (LOD). This unit is shown in Fig. 8.

The mantissa sum now requires to check for any underflow, which requires a leading-one-detector (LOD), and further a dynamic left shifter for mantissa. This situation occurs when two very close mantissa undergoes subtraction operation. The LOD requires to compute the left-shift amount. In present context, the dual-mode leading-one-detector for QPdDP processing is shown in Fig. 9. The input of LOD is either a QP LOD_in or two DP LOD_in. The dual mode LOD is designed in a hierarchical manner, which leads to 64-bit LOD. It is comprised of two 64-bit LOD. The individual 64-bit LOD provides left shift information for both DP operands, and collectively for QP operand. It effectively requires resources equivalent to that of only QP LOD.

The left shift amount, thus generated from LOD, is then updated for sub-normal input cases (both sub-normal input operands) and underflow cases (if left shift amount exceeds or is equal to the corresponding large exponent). For both sub-normal input operand case, the corresponding left shift is forced to zero, and for the underflow case, the corresponding left shift will be equal to corresponding large exponent decremented by one. For the exponent decrements, one of the related subtractor is shared for the QP and first DP, as done in the case of computation of right shift amount. This becomes possible because the required LSBs of $e\_L$ are shared among the exponents of QP and first DP. This exponent decrements requires one 7-bit (shared for QP and a DP) and one 6-bit (for another DP) decrement. All other computations, related to left shift update need to be computed separately for QP and both DPs. With true $qp\_dp$, both DPs' left shift are set to zero, and for false $qp\_dp$ the QP left shift is forced to zero.

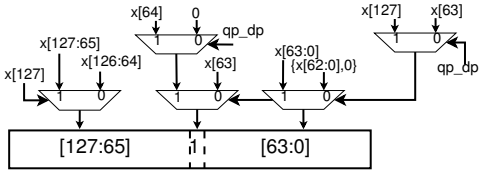


Figure 10: QPdDP Dual Mode 1-Bit Left Shifter.

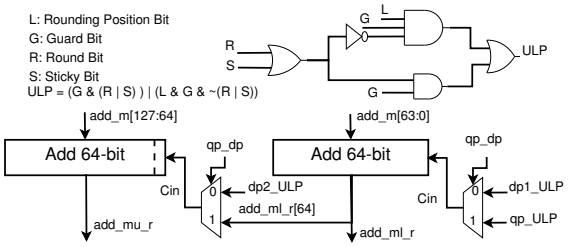


Figure 11: QPdDP Dual Mode ULP Addition.

The mantissa sum is then shifted left using a dual-mode dynamic left shifter (DLS). The basic design concept for dual-mode DLS architecture is similar to the dual-mode DRS, except that there is change in the shifting direction. (Architecture of DLS is not shown due to space limitation).

The output of dual-mode DLS then undergoes 1-bit left shifting (normalization), in-case of mantissa overflow in mantissa-addition. The dual-mode 1-bit left-shifter unit is shown in Fig. 10. It either performs a 1-bit left shifting for QP mode, or carries out 1-bit left-shifting for both DPs, with-respect-to their corresponding mantissa overflow. The resource requirement for this unit is similar to that of a only QP 1-bit shifter, except two 1-bit 2:1 MUX.

The output from 1-bit left shifter is further processed for rounding computation and ULP-addition (Fig. 11). In present work, the round-to-nearest method is included, however, other method can be included easily. The rounding ULP computations are done based on LSB precision bit, Guard bit, Round bit and Sticky bit. Here, the ULP computation is required for separately for each of QP and both DP. However, the ULP-addition is shared among both, as shown in Fig. 11.

Parallel to above mantissa processing, in Exponent-update unit, the exponents are updated for mantissa overflow and mantissa underflow. In this, the large exponents need to be incremented by one or decremented by left shift amount ($LargeExp + mant\_ovf - Left\_Shift$). Since large exponent ($e\_L$) either contains large QP exponent or both DPs exponents, this update is shared for the QP and DP-1, by sharing a subtractor, similar to left shift update computation. In effect it requires a 15-bit shared subtractor and a 11-bit subtractor for DP-2. Thus it needs an extra 11-bit subtractor for DP-2 processing, and a 7-bit multiplexer for left shift amount multiplexing for the shared subtractor, as an overhead over only QP processing.

Finally, the exponents and mantissas are updated for underflow, overflow, sub-normal and exceptional cases to produce the final output, and each requires separate units for QP and both DPs. For overflow, the exponent will be set to infinity and mantissa will be set to zero, and for underflow case exponent will be set to zero and mantissa will take its related computed value. The computed signs, exponents and mantissas of quadruple precision and both double precision are finally multiplexed to produce the final 128-bit output, which either contains a QP output or two DP outputs.

## III. Implementation Results and Comparisons

The proposed dual-mode QPdDP adder architecture is synthesized using UMC90 *nm* standard-cell based ASIC platform, using Synopsys Design Compiler. An architecture for QP only and DP only adder is also designed (using similar data path computational flow) and synthesized for area & period overhead measurements. These architectures are designed with four pipeline stages (as shown in Fig. 1). The implementation details are shown in Table-I. Architectures are synthesized for best possible period. The functionality of proposed architecture is verified using 5-millions random test cases in each mode, with all possible pairs of operands (normal, sub-normal, exceptional cases).

The proposed dual-mode QPdDP adder architecture requires approximately 17% more hardware resources and roughly 5.45% extra period than only DP adder. However, in comparison with a combination of 1-unit QP only and two-units of DP only adder, the proposed QPdDP adder requires approximately 35.86% smaller area ((QP+2*DP-QPdDP)/(QP+2*DP)).

A comparison of dual-mode QPdDP architecture with previous works is shown in Table-II. The comparisons are carried out in terms of % area-overhead and % period-overhead over corresponding only QP adder. Moreover, for a technological independent comparison, gate-equivalent or scaled area equivalent, and "Fan-Out-of-4' (FO4) delay are used. An unified comparison of $area \times period$ is also performed.

A dual-mode QPdDP adder architecture is presented by A. Akkas [7] with 3 & 6 pipelining stages, using 250 *nm* technology. It requires approximately 15% more area and roughly $8-14\%$ more period than their only QP design. The proposed QPdDP architecture has similar area-overhead, with smaller period overhead. Moreover, the $area \times period$ of proposed architecture is much smaller than QPdDP adder of [7]. Furthermore, the architectures shown in [7] does not support sub-normal operands computation and exceptional case handling.

A 110 *nm* based dual-mode QPdDP adder is proposed by [8], with 3-stage and 5-stage pipelines. These architectures do not provide computational support for sub-normal operands and without any exceptional case handling. For their architectures, the area-overhead ranges between

Table I: ASIC Implementation Details

| | DP | QP | QPdDP |
|---|---|---|---|
| Latency | 4 | 4 | 4 |
| Area($\mu m^2$) | 31863 | 76779 | 90116 |
| Area(gates) | 10621 | 25593 | 30038 |
| Period(ns) | 0.95 | 1.1 | 1.16 |
| Period(FO4) | 21.11 | 24.44 | 25.78 |
| Power(mw) | 7.26 | 12.87 | 16.93 |

Table II: Comparison of QPdDP Architecture with Related Work

| | [7] 250*nm* | | [8] 110*nm* | | Proposed 90*nm* |
|---|---|---|---|---|---|
| Latency | 3 | 6 | 3 | 5 | 4 |
| Area OH[1] | 15.3% | 14.01% | 35.80% | 27.31% | 17.37% |
| Period OH[1] | 14.12% | 8.71% | 18.65% | 10.11% | 5.45% |
| Scaled Area[2] | - | - | 239250 | 199723 | 90116 |
| Gate Count[3] | 26967 | 33702 | - | - | 30038 |
| Period (FO4)[4] | 65.28 | 35.92 | 28.9 | 17.81 | 25.78 |
| Area × Period ($10^6$) [#1] | - | - | 6.91 | 3.55 | 2.32 |
| Area × Period ($10^6$) [#2] | 1.76 | 1.21 | - | - | 0.77 |

[1]Area/Period OH = (QPdDP - QP) / QP
[2]in $\mu m^2$ @ 90*nm* = (Area @ 110*nm*) * $(90/110)^2$
[3]Based on minimum size inverter
[4]1 FO4 (ns) $\approx$ (Tech. in $\mu m$) / 2
[#1]Scaled Area × Period (FO4), [#2]Gate Count × Period (FO4)

$27-35\%$ and period overhead is approximately $10-18\%$. Compared to this work, the proposed dual-mode QPdDP architecture outperforms them in terms of design overheads, as well as in terms of design metrics: the area, period and $area \times period$.

Thus, compared to previous works, the proposed dual-mode QPdDP adder architecture has smaller area-overhead and period-overhead when compared to only QP adder. The proposed QPdDP architecture shows an improvement of approximately 50% in terms of unified metrics $area \times period$ products.

## IV. Conclusions

A configurable architecture for dual-mode floating point adder arithmetic is presented in this paper. The proposed dual-mode QPdDP adder architecture provides normal & sub-normal computational support and exceptional case handling. The data path and sub-components in the architecture are constructed/re-designed for on-the-fly dual-mode processing, with minimal required multiplexing. The presented dual-mode QPdDP adder architecture needs approximately 17% more resources and 5.45% more period than the QP only adder. When compared with the best literature work, the proposed dual-mode design has approximately 50% smaller $area \times period$ product, and has smaller area & period overhead over only QP adder. It also provides more computational support than previous literature work. Our future work is aiming towards a tri-mode adder architecture, which along with proposed computation can also be configured to handle four-parallel single precision computation.

## V. Acknowledgments

This work is party supported by the "The University of Hong Kong" grant (Project Code. 201409176200), the "Research Grants Council" of Hong Kong (Project ECS 720012E), and the "Croucher Innovation Award" 2013.

## References

[1] "IEEE Standard for Floating-Point Arithmetic," Tech. Rep., Aug. 2008.

[2] F. de Dinechin and G. Villard, "High precision numerical accuracy in physics research," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 559, no. 1, pp. 207–210, 2006.

[3] D. H. Bailey, R. Barrio, and J. M. Borwein, "High-precision computation: Mathematical physics and dynamics," *Applied Mathematics and Computation*, vol. 218, no. 20, pp. 10 106–10 121, 2012.

[4] H.-J. Oh, S. Mueller, C. Jacobi, K. Tran, S. Cottier, B. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, N. Yano, T. Machida, and S. H.Dhong, "A fully pipelined single-precision floating-point unit in the synergistic processor element of a cell processor," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 4, pp. 759–771, 2006.

[5] NXP Semiconductors, "AN10902 : Using the LPC32xx VFP," in *Application note*, Feb 2010. [Online]. Available: www.nxp.com/documents/application_note/AN10902.pdf

[6] Nvidia, "NVIDIA's Next Generation CUDA$^{TM}$ Compute Architecture: Kepler$^{TM}$ GK110," in *White Paper*, 2014. [Online]. Available: www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

[7] A. Akkas, "Dual-Mode Quadruple Precision Floating-Point Adder," *Digital Systems Design, Euromicro Symposium on*, vol. 0, pp. 211–220, 2006.

[8] ——, "Dual-mode floating-point adder architectures," *Journal of Systems Architecture*, vol. 54, no. 12, pp. 1129–1142, Dec. 2008.

[9] M. Ozbilen and M. Gok, "A multi-precision floating-point adder," in *Research in Microelectronics and Electronics, 2008. PRIME 2008. Ph.D.*, 2008, pp. 117–120.

[10] M. Jaiswal, R. Cheung, M. Balakrishnan, and K. Paul, "Unified architecture for double/two-parallel single precision floating point adder," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 61, no. 7, pp. 521–525, July 2014.

[11] ——, "Configurable architecture for double/two-parallel single precision floating point division," in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, July 2014, pp. 332–337.