A Vector Systolic Accelerator for Multi-Precision Floating-Point High-Performance Computing

Kai Li, Wei Mao, *Member, IEEE,* Junzhuo Zhou, Boyu Li, Zhengke Yang, Shuxing Yang, Laimin Du, Sixiao Huang, and Hao Yu, *Senior Member, IEEE*

***Abstract*—There is an emerging need to design multi-precision floating-point (FP) accelerators for high-performance-computing (HPC) applications. The commonly-used methods are based on high-precision-split (HPS) and low-precision-combination (LPC) structures, which suffer from low hardware utilization ratio and various multiple clock-cycle processing periods. In this paper, a new multi-precision FP processing element (PE) is developed with proposed bit-partitioning method. Minimized redundant bits and operands are achieved. The proposed PE supports 16× half-precision (FP16), 4× single-precision (FP32) and 1× double-precision (FP64) operations with 100% multiplication hardware utilization ratio. Besides, vector systolic structure is designed for PE array to increase the system-level throughput and energy efficiency. The proposed design is realized in a 28-nm process with 1.351-GHz clock frequency. Compared with the existing multi-precision FP methods, the proposed work exhibits the best energy-efficiency performance of 1193 GFLOPS/W at FP16, 317 GFLOPS/W at FP32 and 77.3 GFLOPS/W at FP64 with at least 22.3%, 30% and 3.3% improvement, respectively.**

***Index Terms*—Multi-precision, floating-point, PE, MAC, vector, systolic, HPC, accelerator.**

I. Introduction

Floating-point (FP) based high-performance computing (HPC) has been widely utilized in scientific simulation, model training and big data analysis [1], [2]. To implement different applications, the processing elements (PE) for multiply-accumulation (MAC) operations in the accelerators have to fulfill the multi-precision computing requirements, especially the operations of 16-bit half-precision (FP16), 32-bit single-precision (FP32) and 64-bit double-precision (FP64) modes [3]. However, high-precision FP calculation brings enormous computing power and long latency time [4]. Consequently, FP arithmetic of assorted multiple precisions is commonly employed in many intensive computing applications to improve performance, especially to accelerate scientific computations.

So far, great efforts have been spent on the designs for multi-precision FP MAC operations [5], [6]. One of main problem is that the multiplication part has redundant operations, resulting in low hardware resource utilization [7]. On the other hand, some works such as [8] achieved multi-precision configurable implementation by truncating the mantissa to lower bitwidth, which causes accuracy loss. [9] proposed a bit-partitioning method of mantissa to reduce redundant bits and increase resource utilization. However, total redundant bits per segment are still high due to fused multiplier. For [10], different cycles for different FP modes are required and the ratios of computation times between different precisions are inconsistent, which is not compatible to construct geometric and compact dataflow for large-scale data processing. In [11] and [12], new FP processing units with supporting flexible posit format are proposed to achieve efficient power performance.

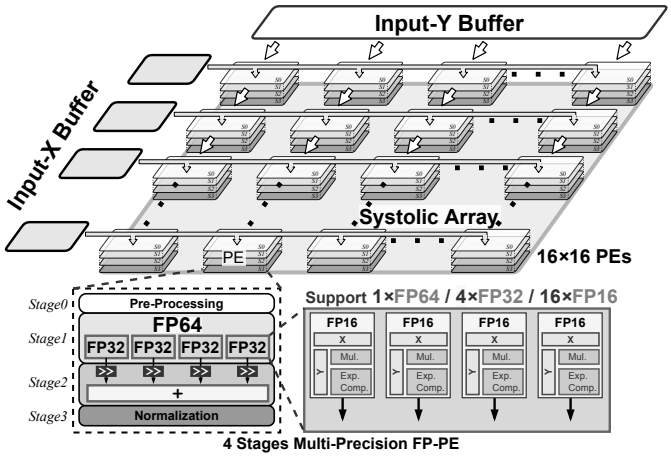

Fig. 1. Architecture of vector systolic array with proposed multi-precision FP vector PEs for FP16, FP32 and FP64 operations.

In this work, we propose a vector systolic accelerator integrated with multi-precision FP PEs to overcome above issues. Fig. 1 shows the overall architecture of the accelerator. The proposed 4-stage multi-precision FP PE can support 16× half-precision (FP16), 4× single-precision (FP32) and 1× double-precision (FP64) operations with minimized redundancy and geometric ratio in different precision modes. The proposed systolic accelerator is constructed by 16 PE vectors (PEVecs) that contains 16 proposed PEs. During computing period, input-X matrix data is stationary while input-Y dataflow is systolic, which benefits from data-reuse strategies and achieve excellent energy-efficient and low-latency performance.

The remainder of this paper is organized as follows. Section II provides optimized multi-precision bit-partitioning method. Section III illustrates the working procedures of proposed PE and vector systolic accelerator. Section IV and V provides the experiment results and conclusion, respectively.

Manuscript received XXXX XX, 2022; revised XXXX XX, 2022; accepted XXXX XX, 2022. This work is supported by the National Natural Science Foundation of China (NSFC) (Key Program Grant No. 62034007), the National Key R&D Program of the Ministry of science and technology (Grant No. 2021YFE0204000), the Shenzhen Science and Technology Program (Grant No. KQTD20200820113051096) and NSQKJJ (Grant No. K21799101). This paper was recommended by XXXX. Kai Li and Wei Mao are joint first authors. *(Corresponding author: Wei Mao)*

K. Li, W. Mao, J. Zhou, B. Li, Z. Yang, S. Yang, S. Huang and H. Yu are with the School of Microelectronics, and Engineering Research Center of Integrated Circuits for Next-Generation Communications, Ministry of Education, Southern University of Science and Technology, Shenzhen China.

TABLE I
PARAMETER SUMMARY

Parameter	Description
s_{an}, s_{bn}	The sign bit of n^{th} element of vector$\boldsymbol{A}$ and $\boldsymbol{B}$.
M_{an}, M_{bn}	The mantissa of n^{th} element of vector $\boldsymbol{A}$ and $\boldsymbol{B}$.
E_{an}, E_{bn}	The exponent of n^{th} element of $\boldsymbol{A}$ and $\boldsymbol{B}$.
$bias$	The $bias$ is 15, 127 and 1023 for FP16, FP32 and FP64.
M	M is 11, 24 and 53 for FP16, FP32 and FP64.
L	The vector length of the $\boldsymbol{A}$ and $\boldsymbol{B}$.
σ	The length of splitting mantissa per segment.

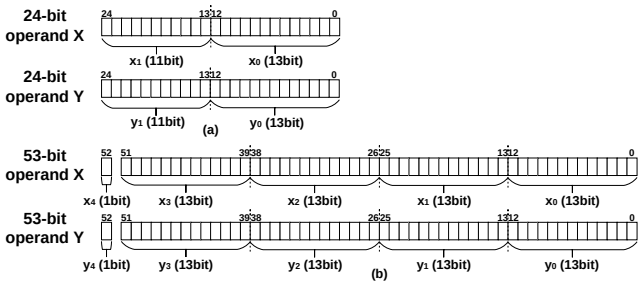

Fig. 2. Proposed bit-partitioning method on FP32 and FP64 operands: (a) 24-bit mantissa for FP32; (b) 53-bit mantissa for FP64.

II. MULTI-PRECISION BIT-PARTITIONING METHOD

A. Floating-Point MAC Operation

Based on the IEEE-745 standard [3], floating-point operands and operations are given as following equations, where the parameters used are summarized in Table I.

$$A = (-1)^{s_a} \times M_a \times 2^{(E_a - bias)} \tag{1}$$

$$B = (-1)^{s_b} \times M_b \times 2^{(E_b - bias)} \tag{2}$$

$$A \times B = (-1)^{s_a + s_b}(M_a \times M_b)2^{(E_a + E_b - 2bias)} \tag{3}$$

The operands A and B are represented by (1) and (2). A times B can be expressed by (3).

$$\begin{aligned}\boldsymbol{A}\cdot\boldsymbol{B} &= \sum_{n=1}^{L}(-1)^{s_{an}+s_{bn}}(M_{bn}\times M_{an})2^{(E_{an}+E_{bn}-2bias)} \\ &= \sum_{l=0}^{\lceil M/\sigma\rceil-1}\sum_{k=0}^{\lceil M/\sigma\rceil-1}\sum_{n=1}^{L}\sum_{j=0}^{\sigma-1}\sum_{i=0}^{\sigma-1}(-1)^{s_n}(M_{an}[j+l\sigma] \\ &\quad \times M_{bn}[i+k\sigma])2^{i+j+(l+k)\sigma-2(M-1)+E_n}\end{aligned} \tag{4}$$

For MAC operation, the dot product implementation of L-length vector $\boldsymbol{A}$ and vector $\boldsymbol{B}$ can be expressed by (4).

B. Multi-Precision Bit-Partitioning Analysis

In this part, the redundant bits and operands are analyzed to evaluate the bit-partitioning method. The redundant bits are equal to the differences between the values of segment lengths multiplying segment numbers and mantissa bitwidths M. For redundant operands, the sum of redundant operands is the difference between the total operands and valid operands.

TABLE II
HARDWARE COST ANALYSIS OF BIT-PARTITIONING METHOD

Design		-	[9]	-	-	[6]	Proposed
Segments Length (σ)		11	12	13	14	15	13
Segments ($\lceil M/\sigma \rceil$)	FP16	1	1	1	1	1	1
	FP32	3	2	2	2	2	2
	FP64	5	5	5	4	4	4 ($\lfloor M/\sigma \rfloor$)
Redundant Bits	FP16	0	1	2	3	4	2
	FP32	9	0	2	4	6	2
	FP64	2	7	12	3	7	0
Total Redun. Bits		**11**	**8**	**16**	**10**	**17**	**4**
Total Operations		27	25	25	16	16	16
Vector Length (L)	FP16	27	25	25	16	16	16
	FP32	3	6	6	4	4	4
	FP64	1	1	1	1	1	1
Total Operands		3267	3600	4225	3136	3600	2704
Redundant Operands	FP16	0	575	1200	1200	1664	768
	FP32	1539	144	769	832	1296	400
	FP64	458	791	1416	327	791	0
Total Redun. Operands		**1997**	**1510**	**3385**	**2359**	**3751**	**1168**

The total operands and valid operands are the counts of all cycle in (4) and the counts that FP operation actually needed, respectively. In Table II, the redundant bits and redundant operands are summarized and compared for different values of σ. Equation (5) to (7) show the details for operand analysis.

$$OP_{total} = \lceil M/\sigma \rceil^2 \times L \times \sigma^2 \tag{5}$$

$$OP_{valid} = L \times M^2 \tag{6}$$

$$OP_{redun} = OP_{total} - OP_{valid} \tag{7}$$

In general, the segment number is depended on the ceiling of M/σ. The total operations have to satisfy FP16, FP32 and FP64 precisions. By considering the maximum effective calculation, the total operations should be the least common multiple (LCM) of the square of segment numbers in three precisions or as close to LCM as possible. In addition, to minimize the hardware cost, the cycle of n can be reused by changing the cycle of l and k in (4). Therefore, the total cycle of l, k and n ($\lceil M/\sigma \rceil^2 \times L$) should be less than the square of segment numbers. Under this principle, the analysis results are presented in Table II.

In this work, the value of σ 13 is chosen to minimize the hardware cost. As shown in Fig.2, the 53-bit mantissa of FP64 is divided into 1-bit and 52-bit parts. Then the 52-bit part is divided into four segments: 13, 13, 13 and 13. The extra 1 bit is processed separately in the accumulation stage. Thus, the proposed work reduces the number of segments of FP64 to 4 because the value of M/σ is 4.07 which is closer to 4 instead of rounding up to 5. In this way, the total redundant bits and operands are minimized to 4 and 1168, respectively.

III. PROPOSED ARCHITECTURE

A. Multi-Precision PE

The proposed multi-precision PE is designed with 4 stages S0∼S3, which is illustrated in Fig. 3. S0 is the input prepro-

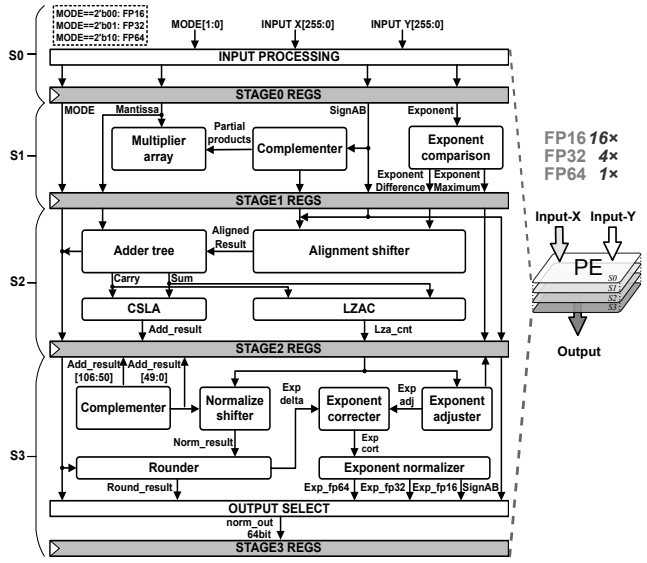

Fig. 3. Architecture of proposed multiple-input vector FP PE with four pipelined stages.

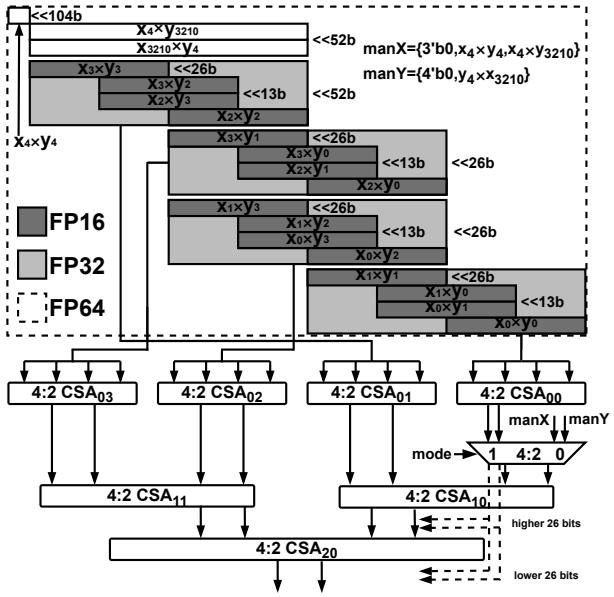

Fig. 4. Accumulation process with 3-stage adder tree within multi-precision PE for FP16, FP32 and FP64 modes.

cessing stage which splits 256-bit *X* and *Y* input into sign bits, exponent bits and mantissa bits according to different modes. Then, exponent bits and mantissa bits will enter the exponential comparator and multiplier array in S1, respectively. The exponential comparator compares 16 exponents to get the maximum one and other 10-bit corresponding exponent differences. Besides, the multiplier array in S1 consists of 16 13-bit multipliers with 26-bit products for vector multiplication. For proposed PE, 16×FP16, 4×FP32 and 1×FP64 operations can be implemented in one clock under different mode selections. In S2, 16 products are sent to the alignment shifter. The shifting number for each product will depend on exponent differences and the sum of the segment numbers. After alignment, adder tree is used for vector accumulation. In Normalization part of S3, the leading zero anticipating and counting (LZAC) unit is used for shifting amount prediction [6], [13]. And the mantissa will be rounded according to RoundtowardPositive. Meanwhile, the processing of abnormal values, 0 and invalid values are also considered.

In S2, the adder tree is designed with three levels of 4:2 CSAs, of which the accumulation process is provided in Fig. 4. For FP16 mode, 16 products in dark gray boxes are shifted according to the exponential differences, which are sent to the adder tree and pass through 3-level CSAs directly. For FP32 mode, 16 products in light gray boxes are shifted according to the exponential differences and segment sums. Then the shifted products are accumulated by adder tree directly. In FP64 mode, the corresponding process is shown in the dashed box with detailed shifting values. 16 products are also shifted at first. Then the products are passed to CSA_{00}-CSA_{03} in 1st level. Then, CSA_{10}-CSA_{11} in 2nd level will process not only the data from the 1st level, but also the products of manA and manB. manA and manB are constructed by the partial products from the splitted MSB 1 bit as shown in Fig. 2. Third level is an 83-bit CSA_{20}. In FP64 mode, 52-bit carry and sum results from CSA_{00} are not processed through CSA_{10}, which are splitted into higher 26-bit parts and lower 26-bit parts as the supplementary parts of CSA_{20} inputs and outputs. The higher 26-bit parts concatenate with CSA_{10} outputs to form CSA_{20} inputs. The lower 26-bit of parts construct the lower 26-bit carry and sum results of final outputs directly. When compared with the conventional FP PE without bit-reconfiguration, the proposed PE with additional circuits has 27.23% area overhead and 25.67% power overhead.

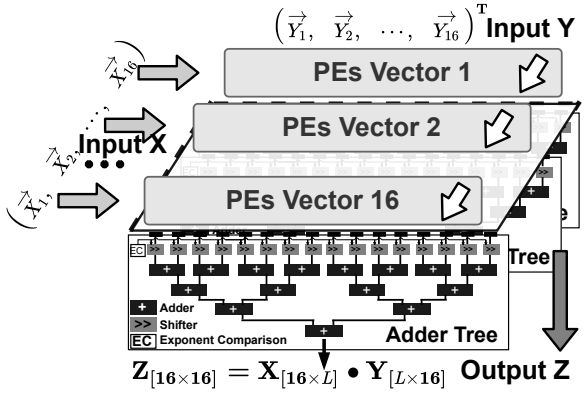

Fig. 5. Structure and dataflow of vector systolic array.

B. Vector Systolic Accelerator

Next, a vector systolic array for matrix multiplication is constituted by the proposed PEs. Both the conventional non-stationary systolic array (NSSA) [17] and the TPU-style weight-stationary systolic array [18] are unsuitable for multi-stage FP PE works. The output flow of the traditional design is coupled to the input systolic flow. Thus, accumulation must be completed to generate a new partial sum (PSum) with large latency. In this work, the output flow is perpendicular to the input flow. The overall latency is reduced. The vector systolic array structure and its dataflow are depicted in Fig. 5. By considering hardware cost and bandwidth limitation, the proposed array consists of 16 PE vectors (PEVecs), each

TABLE III
PERFORMANCE COMPARISON OF THE PROPOSED WORK WITH STATE-OF-THE-ART DESIGNS

Design		Node (nm)	Frequency (MHz)	Latency (ns)	Area (mm^2)	Voltage (V)	Power (mW)	Mul. Utilization Ratio (%)			Throughput (GFLOPS)			Energy Efficiency (GFLOPS/W)		
								FP16	FP32	FP64	FP16	FP32	FP64	FP16	FP32	FP64
ICECS	**2010 [5]**	130	291	10.29	0.287	1.2	35.2	25	50	100	-	0.58	0.29	-	33.13	16.52
ISSCC	**2012 [14]**	32	1450	2.07	0.045	1.05	60.0	50	100	-	2.90	1.45	-	96.67	52.00	-
ARAB	**2016 [8]**	90	357	8.4	0.082	1.0	250.5	100	100	-	5.72	1.43	-	45.64	11.41	-
arXiv	**2016 [15]**	28	1360	4.41	0.018	0.8	18.38	-	25	100	-	1.36	1.36	-	110.00	43.70
TCOM	**2019 [6]**	90	667	4.5	0.795	1.0	43.8	12.5	25	50	2.67	1.33	0.67	121.77	60.88	30.44
TVLSI	**2020 [16]**	22	923	3.25	0.049	0.8	57.41	25	50	100	3.69	1.85	0.92	497.67	199.70	74.83
TVLSI	**2022 [10]**	28	1429	2.8	0.013	1.0	29.3	100	100	100	14.29	3.57	0.71	975.13	243.78	48.76
This Work	**PE Accelerator**	**28**	**1351**	**2.96**	**0.015**	**1.0**	**38.03**	**100**	**100**	**100**	**43.2**	**10.8**	**2.70**	**1136**	**302.2**	**73.6**
		28	**1351**	**2.96**	**5.33**	**1.0**	**9275**	**100**	**100**	**100**	**11067**	**2767**	**691.7**	**1193**	**317**	**77.3**

PEVec contains 16 PEs and one 16-input adder tree. The adder tree is constructed by exponent comparison (EC) part, alignment shifter part and adder part. The PE outputs are sent to these parts for exponent difference generation, mantissa alignment and parallel accumulation sequentially. The systolic dataflow makes it possible to perform 8192 FP16, 2048 FP32 or 512 FP64 MAC operations each clock cycle.

The input-X matrix consists of 16 row vectors $\vec{X_1} - \vec{X_{16}}$ with the shape of $[16 \times L]$, where length L of each vector is 256, 64 and 16 for FP16, FP32 and FP64 data formats, respectively. The input-Y is similar but transposed. The data flow can be described as follows. The input-X matrix data is stationary and it will be preloaded horizontally to the systolic array one-row-per-cycle and then remain for 16 clocks at first. The input-Y dataflow is systolic and will pass through the PE array in a row-by-row style. And the output-Z dataflow is pipelined and will finally output from adder tree.

$$\mathbf{X} = \begin{pmatrix} \vec{X_1} \\ \vec{X_2} \\ \vdots \\ \vec{X_{16}} \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} \vec{Y_1} \\ \vec{Y_2} \\ \vdots \\ \vec{Y_{16}} \end{pmatrix}^{\mathbf{T}}. \tag{8}$$

$$Where\ \vec{X_m} = \begin{pmatrix} x_{m,1} & x_{m,2} & \cdots & x_{m,L} \end{pmatrix},$$
$$and\ \vec{Y_n} = \begin{pmatrix} y_{n,1} & y_{n,2} & \cdots & y_{n,L} \end{pmatrix}.$$

$$\mathbf{Z} = \mathbf{X} \bullet \mathbf{Y} = \begin{pmatrix} \vec{Z_1} \\ \vec{Z_2} \\ \vdots \\ \vec{Z_{16}} \end{pmatrix} = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,16} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,16} \\ \vdots & \vdots & \ddots & \vdots \\ z_{16,1} & z_{16,2} & \cdots & z_{16,16} \end{pmatrix} \tag{9}$$

$$z_{m,n} = \vec{X_m} \cdot \vec{Y_n}^T = \sum_{k=1}^{L} x_{m,k} \cdot y_{n,k} \tag{10}$$

To get output of (9), $\vec{X_1} - \vec{X_{16}}$ are sent to PEVec1-PEVec16 while $\vec{Y_1} - \vec{Y_{16}}$ are sent to PEVec1 and pass to other PEVecs in the systolic flow at clk_1-clk_{16}. Full 16×16 outputs are obtained after 16+16+7 clock cycles. In (10), $\vec{Z_m}$ is $PEVec_m$ output, and $z_{m,n}$ is obtained at $clk = m + n - 1 + t$, where $t = 7$ is the number of pipeline stages. In each computation cycle, input X and Y are reused 16 times, which reduces the loads of I/O drive and energy consumption in terms of data access.

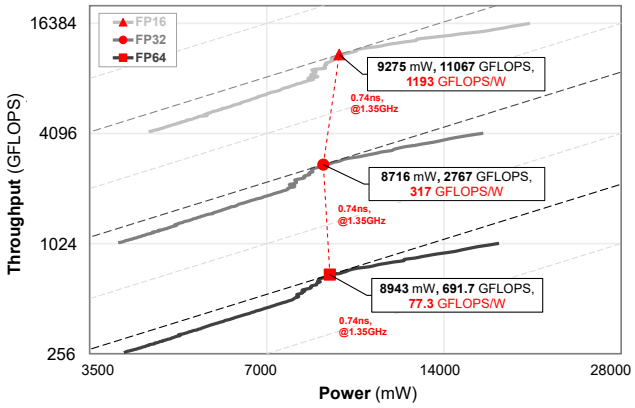

Fig. 6. Speed versus power at different timing constraints.

IV. EXPERIMENT RESULTS

A. Methodology

The performance evaluation is implemented on 28-nm process with 1.0 V. Digital synthesis, dynamic timing evaluation and power performance simulation are done by Synopsys Design Compiler, VCS and PrimeTime PX.

B. Performance Evaluation

To obtain the suitable timing constraint for energy-efficiency of PE and use it in future higher-level evaluation, we set a timing constraint range [0.5:0.01:2.0] in ns and get the Performance, Power and Area (PPA) of each timing constraint.

Fig. 6 represents the throughput and power performance with different timing constraints. The three lines in light grey, dark grey and black correspond to the simulation results of FP16, FP32, and FP64, respectively. The coordinate is LOG4-LOG2 plot, and different dashed-lines are iso-energy-efficiency lines. Experiment result shows that our design obtains best performance under the condition that timing-constraint of 0.74ns and frequency of 1351MHz. And the best energy efficiency is 1193, 317, and 77.3 in GFLOPS/W for FP16, FP32 and FP64 modes, respectively.

C. Performance Comparison

Table III summarizes multi-precision operation performance in different works, including process, frequency, latency, area,

TABLE IV
FPGA PERFORMANCE SUMMARY

Designs		[19]	[20]		[10]	Prop. PE
Xilinx Virtex Ver.		5	5	7	US+	US+
Frequency (MHz)		336.0	348.2	427.4	179.9	164.5
Power (W)		0.25	0.248	0.189	0.378	0.483
Mul. Oper. Per Clock	FP16	-	-	-	10	16
	FP32	2/7	1/3	1/3	5/2	4
	FP64	1/12	1/9	1/9	1/2	1
Energy Effi. (GFLOPS/W)	FP16	-	-	-	9.516	10.89
	FP32	0.768	0.936	1.507	2.379	2.72
	FP64	0.224	0.312	0.502	0.476	0.68

voltage, power, throughput and energy efficiency. As shown, [5] splits high precision multiplier to support one FP64 operation and two FP32 operations, leading to long latency and large area overhead because of its large bitwidth multiplication. With similar splitting method to support FP32 and FP16 operations, [15] also performs not very well in low precision. [8] which can perform FP32 operation and truncate the mantissa to lower bitwidth to support FP16 operation also lead to low hardware utilization ratio and poor energy efficiency.

In [6], 15-bit multipliers are used to optimize FP128 operation. The implementation for FP16, FP32 and FP64 operations have large bit redundancy. As shown in Table III, the area of this design is very large. In [16], best energy efficiency is achieved for FP64 operation with full hardware utilization ratio by flexible bitwidth processing ability. However, for FP16 operation, its energy efficiency is only half of the proposed design. [10] has considered the redundancy bits achieved improved throughput and energy efficiency compared with previous work. But it still has not reached the minimum redundancy due to fused multiplier. Besides, [10] requires different cycles for different FP modes and the ratios of computation times between different precisions are inconsistent, which is not compatible for geometric dataflow. The proposed vector systolic accelerator is designed for the multi-precision FP matrix multiplication. By using redundancy-minimized bit-partitioning method, the proposed PE can achieve 1136 GFLOPS/W at FP16 mode, 302.2 GFLOPS/W at FP32 mode and 73.6 GFLOPS/W at FP64 mode. Thanks to the data reuse strategy from systolic dataflow in the proposed accelerator, average number of used registers per PE is reduced, which improves the performance further with 1193 GFLOPS/W at FP16 mode, 317 GFLOPS/W at FP32 mode and 77.3 GFLOPS/W at FP64 mode.

D. FPGA Implementation

The proposed multi-precision PE is verified on Xilinx FPGA platform with Virtex UltraScale+ (US+). Due to hardware recourse constraints, the accelerator is not implemented. The performance of different multi-precision FP PEs on Xilinx platform are also summarized for comparison in Table IV. Results show that proposed PE can achieve best energy efficiency performance with at least 14.4% improvement.

V. CONCLUSION

In this work, a redundancy-minimized bit-partitioning method is proposed to develop a multi-precision FP PE. To further improve throughput, the vector systolic accelerator is designed for matrix multiplication. High energy efficiencies of 1193 GFLOPS/W at FP16 mode, 317 GFLOPS/W at FP32 mode, and 77.3 GFLOPS/W at FP64 mode is achieved. Compared with the existing multi-precision designs, the proposed work improves 22.3%, 30% and 3.3% performance for FP16, FP32 and FP64 operations, respectively.

REFERENCES

[1] J. A. Kahle *et al.*, "2.1 summit and sierra: designing AI/HPC supercomputers," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, 2019, pp. 42–43.
[2] Y. Tanimura *et al.*, "Building and evaluation of cloud storage and datasets services on AI and HPC converged infrastructure," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 1992–2001.
[3] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug 2008.
[4] I. Sourdis *et al.*, "Resilient chip multiprocessors with mixed-grained reconfigurability," in *IEEE Micro*, Jan. 2015, pp. 35–45.
[5] K. Manolopoulos *et al.*, "An efficient dual-mode floating-point multiply-add fused unit," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2010, pp. 5–8.
[6] H. Zhang *et al.*, "Efficient multiple-precision floating-point fused multiply-add with mixed-precision support," *IEEE Transactions on Computers*, vol. 68, no. 7, pp. 1035–1048, July 2019.
[7] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.
[8] S.-R. Kuang *et al.*, "A multi-functional multi-precision 4D dot product unit with SIMD architecture," *Arabian Journal for Science and Engineering*, vol. 41, no. 8, pp. 3139–3151, 2016.
[9] W. Mao *et al.*, "A reconfigurable multiple-precision floating-point dot product unit for high-performance computing," in *2021 IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1793–1798.
[10] W. Mao *et al.*, "A configurable floating-point multiple-precision processing element for HPC and AI converged computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2021.
[11] H. Zhang and S.-B. Ko, "Design of power efficient posit multiplier," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 861–865, 2020.
[12] L. Crespo, P. Tomás, N. Roma, and N. Neves, "Unified posit/IEEE-754 vector mac unit for transprecision computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2478–2482, 2022.
[13] M. Schmookler and K. Nowka, "Leading zero anticipation and detection-a comparison of methods," in *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, 2001, pp. 7–12.
[14] H. Kaul *et al.*, "A 1.45 GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS," in *2012 IEEE International Solid-State Circuits Conference*, 2012, pp. 182–184.
[15] J. Pu *et al.*, "FPMax: A 106GFLOPS/W at 217GFLOPS/mm^2 single-precision FPU, and a 43.7 GFLOPS/W at 74.6GFLOPS/mm^2 double-precision FPU, in 28nm UTBB FDSOI," *arXiv preprint arXiv:1606.07852*, 2016.
[16] S. Mach *et al.*, "FPnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 774–787, 2020.
[17] H. T. Kung *et al.*, "Systolic arrays for (VLSI)," in *Sympos. Sparse Matrix Comput.*, 1978, pp. 256–282.
[18] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.
[19] M. K. Jaiswal and R. Cheung, "Area-efficient architectures for double precision multiplier on FPGA, with run-time-reconfigurable dual single precision support," *Microelectronics Journal*, vol. 44, no. 5, pp. 421–430, 2013.
[20] H. Zhang, D. Chen, and S.-B. Ko, "Area- and power-efficient iterative single/double-precision merged floating-point multiplier on FPGA," *IET Computers & Digital Techniques*, vol. 11, no. 4, pp. 149–158, July 2017.