Received 31 December 2022, accepted 9 January 2023, date of publication 16 January 2023, date of current version 20 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3237079

RESEARCH ARTICLE

Multi-Mode SpMV Accelerator for Transprecision PageRank With Real-World Graphs

WHIJIN KIM, (Graduate Student Member, IEEE),
HANA KIM, (Graduate Student Member, IEEE),
JIHYE LEE, (Graduate Student Member, IEEE), HYUNJI KIM, (Graduate Student Member, IEEE),
AND JI-HOON KIM, (Senior Member, IEEE)
Department of Electronic and Electrical Engineering, Ewha Womans University, Seoul 03760, South Korea
Smart Factory Multidisciplinary Program, Ewha Womans University, Seoul 03760, South Korea

Corresponding author: Ji-Hoon Kim (jihoonkim@ewha.ac.kr)

This work was supported by IITP grant funded by the Korea Government (MSIT) (No. 2022-0-00441, Memory-Centric Architecture Using the Reconfigurable PIM Devices). This work was supported by the Ewha Womans University Research Grant of 2020. The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

ABSTRACT With the development of Internet networks, the PageRank algorithm, which was initially developed to recommend important pages in Google's web search systems, is widely used as the basis of various ranking systems in graph processing fields. However, PageRank algorithm requires Sparse Matrix-Vector Multiplication (SpMV) repeatedly which becomes main bottleneck for the calculation. In this study, we present the multi-mode SpMV accelerator for half-to-single transprecision PageRank with real-world graphs. To support transprecision, where the operation performs in half-precision (FP16) initially and changes its precision to single-precision (FP32), the proposed multi-mode SpMV accelerator can perform both dual FP16 mode and single FP32 mode. In dual FP16 mode, the proposed accelerator performs two FP16 SpMV in parallel, and in single FP32 mode, it performs one FP32 SpMV with the same hardware resources. Also, for the reduction of memory footprint, the proposed accelerator supports the Compressed Sparse Row (CSR) format. In addition, dual-issue accumulator and multi-mode transprecision multiplier are presented to support both FP16 and FP32 modes. Validation of the proposed transprecision PageRank algorithm is performed with four real-world graph datasets, resulting in a low 0-4% error rate and 1.3×-1.9× speedup compared to single-precision PageRank computation without transprecision.

INDEX TERMS Transprecision, sparse matrix vector multiplication (SpMV), PageRank (PR), floating-point multiplier, single-precision (FP32), half-precision (FP16), graph processing.

I. INTRODUCTION

With the development of Internet communication and social networks, large amounts of data are pouring into users. To deal with this big data, many algorithms have been developed to analyze and provide useful information to users. Particularly, web search engines should quickly extract the information that users want and display it in order of importance. The most basic algorithm for web search, named PageRank, was developed by Larry Page in 1998 for Google's search engine and is still widely used [1]. It was initially developed for search engines working with web-scale datasets but is now being utilized in various fields to rank target data from different types of datasets. PageRank algorithm uses a dataset that displays web pages and their relationships with connection information. With a graph theory notation, each page is represented as a node or vertex, and the connection of pages is represented as an edge or link. In PageRank, nodes with many links to other nodes would have high PageRank values. In addition, if the node is connected to the other node with a high PageRank value, the PageRank value of this node also would be high. The PageRank vector is generated as the output of PageRank calculation, representing the importance of each page. With this output vector, the search engine can recommend the most weighted result to the user. To this day, several algorithms such as personalized PageRank are being developed using the basic concept of PageRank [2], [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Paolo Crippa.

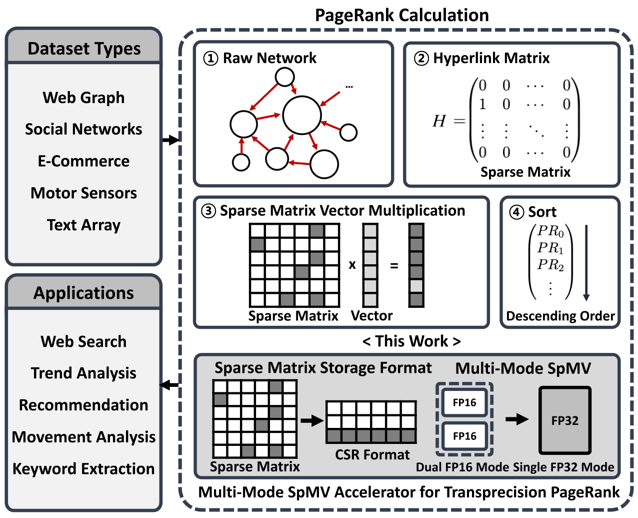


FIGURE 1. Overall PageRank calculation flow with the proposed multi-mode SpMV accelerator for transprecision PageRank.

TABLE 1. Details of datasets from SNAP used in the evaluation.

Dataset	Type	# of Nodes	# of Edges
Amazon	E-Commerce	262,111	1,234,877
Twitter	Social Network	81,306	1,768,149
Stanford	Web Graph	281,903	2,312,497
Pokec	Web Graph	1,632,803	30,622,564

Fig. 1 shows the overall calculation flow of PageRank which can be utilized for various applications. Most representatively, PageRank can be used for web search using large web-scale graphs [4], [5]. Social networks are used to analyze trends by extracting the key resources or calculating a ranking of users [6], [7], [8]. PageRank in e-commerce system allows users to get the appropriate recommendation of products [9]. To analyze human movement, motor sensors are used to identify the movement path based on the importance ranking of sensors calculated by PageRank [10]. TextRank algorithm was developed to extract keywords by obtaining ranking for a text array based on PageRank algorithm [11].

In PageRank calculation, hyperlink matrix H is generated to indicate the connection of each node in the network datasets. In hyperlink matrix, a nonzero element shows that there is a connection between the two nodes, while a zero element indicates that there is no connection. Real-world graphs are represented as a sparse hyperlink matrix mostly filled with zero elements because it has very few connections compared to a large number of nodes. During the calculation, the sparse matrix is repeatedly multiplied with the vector, and this process is known as Sparse Matrix-Vector Multiplication (SpMV). Since SpMV is the main bottleneck of the PageRank operation, it becomes an important challenge to accelerate this operation without losing accuracy.

Recent studies suggest dedicated accelerators to optimize SpMV operation. To reduce the memory overhead, [12] divided the SpMV algorithm into two steps and used parallel floating-point multipliers, focusing on high-performance computing for large and hyper-sparse datasets. Matrix partitioning scheme was also proposed to accelerate SpMV operation [13]. Partitioning of the sparse matrix using Compressed Sparse Row (CSR) format enabled parallel computation. Changing the general precision scheme of PageRank was effective to implement the SpMV accelerator on FPGA platform [14]. For high accuracy, PageRank is generally calculated in a 32-bit single-precision floating-point (FP32). To reduce the hardware overhead incurred by FP32 operation, 20- to 26-bit fixed-point calculation scheme was proposed, showing its efficiency on top-10 to top-50 rankings.

Also, there have been several software approaches suggesting a new precision scheme for PageRank calculations. In general, dedicated hardware accelerators are superior to software approaches on programmable computing units, such as CPUs and GPUs, both in terms of performance and power consumption which leads to their wide adoption in many digital signal processing (DSP) applications and recent Neural Networks (NNs) [15], [16], [17]. The customized 16-bit half-precision type with the requirement of additional encoding and decoding process was presented and simulated in [18]. In [19] and [20], the double-precision number system with mantissa segmentation was proposed. The transprecision scheme with the double-precision was applied to PageRank for the CUDA GPU environment, which requires more iterations to reach the target threshold. In this work, we present FP16 to FP32 transprecision PageRank algorithm and also provide its optimal hardware architecture based on multi-mode SpMV accelerator with FPU resource sharing and memory system architecture.

In our previous work, the transprecision SpMV engine with Coordinate (COO) format was presented, without any analysis of speedup and hardware implementation results [21]. In this work, we present the low-complexity transprecision scheme for PageRank to support the sparse matrix efficiently with a compressed storage format. As illustrated in Fig. 1, the proposed transprecision scheme starts its operation with a 16-bit half-precision floating-point (FP16) and changes to a 32-bit single-precision floating-point (FP32) at the optimal point. For reduced memory requirement in main memory, the multi-mode SpMV accelerator supports executing the transprecision algorithm with data in CSR format, with comparison to COO format. Also, the proposed SpMV accelerator supports two operation modes where it can compute two FP16 SpMVs in dual FP16 mode and one FP32 SpMV in single FP32 mode. Since the proposed accelerator can execute two FP16 multiplications in parallel, we can achieve the speedup compared to the baseline approach in single-precision operations saving hardware resources.

As denoted in Table 1, we use four real-world graph datasets with various node and edge sizes from Stanford Large Network Datasets Collections (SNAP) [22] which is widely used to simulate graph processing algorithms [14], [23].

The rest of this paper is organized as follows. Section II explains the backgrounds of PageRank equations and sparse matrix storage format, with the concept of transprecision. In Section III, we present the data storage requirements for transprecision PageRank. The hardware architecture of the multi-mode SpMV accelerator is described in Section IV. The experimental results are explained in Section V. Finally, Section VI contains the conclusions from this work.

II. BACKGROUNDS

This section provides the fundamentals of PageRank algorithm, sparse matrix storage format, and transprecision scheme which should be considered together for the optimized transprecision PageRank operation.

A. OVERVIEW OF PageRank ALGORITHM

Since the development of PageRank in 1998, it has been used as an early basis for Google and is still applied in various ranking algorithms. The basic concept of PageRank is that the importance of each node is determined by the other nodes which have an association with it. PageRank value can be calculated as the sum of the linked nodes [24]. Based on this concept, PageRank can be expressed in vector and matrix notation with the power iteration method. Consider a directed graph $G = (V, E)$ which has $n = |V|$ number of vertices and $|E|$ number of edges. Hyperlink matrix, or weighted adjacency matrix $H \in \mathbb{R}^{n\times n}$ can be expressed as follows:

$$H[i,j] = \begin{cases} 1/O_i & \textit{if } (v_i, v_j) \in E \\ 0 & \textit{otherwise} \end{cases} \tag{1}$$

where O_i indicates the number of outgoing links from the node i, v_i to node j, v_j. Since hyperlink matrix, H only has the nonzero value when the node is linked to the other nodes, it becomes the matrix with high sparsity. The PageRank equation in each iteration can be written as follows:

$$PR^{\{k+1\}} = \frac{(1-\alpha)}{n}e + \frac{\alpha}{n}ew^T PR^{\{k\}} + \overbrace{\alpha H^T PR^{\{k\}}}^{\text{SpMV}} \tag{2}$$

where $PR^{\{k\}}$ denotes the PageRank column vector in k-th iteration [25]. At each iteration, the entire graph data is read and calculated through (2), and these iterations are repeated until the whole PageRank calculation is finished. α is the damping factor, which models the situation where the user stops clicking, generally set to 0.85 experimentally [26]. e is a column vector whose element is all 1, while dangling vector w is a column vector that has an element of 0 if the node has an outgoing link and 1 if not. In (2), the first and second terms indicate constant additions with simple computation. However, the last term consists of multiplication with hyperlink matrix H and PageRank vector PR. This multiplication becomes SpMV operation and is widely known as the biggest challenge for PageRank calculation [27].

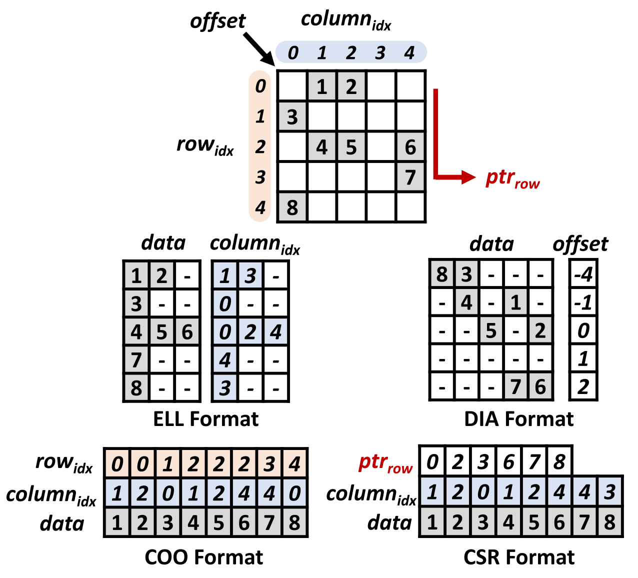
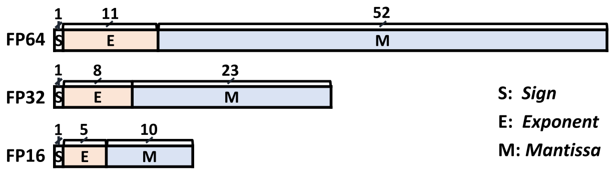


FIGURE 2. Examples of ELL, DIA, COO, and CSR formats.

B. SPARSE MATRIX STORAGE FORMATS

Sparse matrix is a matrix that mostly has 0 as its element. When the sparse matrix becomes very large, it requires a significant amount of memory to store whole data although only a portion is non-zero values. In PageRank calculation, if the total number of nodes in the graph is given as $|V|$, a memory with a size of $|V|^2$ is required to store all data. Memory waste due to a huge amount of sparse memory is a fatal problem in big data applications, which generally deal with large datasets.

To overcome this problem, various compression formats for the sparse matrix, such as Coordinate (COO), Compressed Sparse Row (CSR), Compressed Sparse Column (CSC), Ellpack (ELL), and Diagonal (DIA), are suggested [28]. Fig. 2 shows examples of four sparse matrix storage formats. ELL format compresses the matrix in data and column indices. It is suitable for matrices obtained from semi-structured meshes and unstructured meshes, while the maximum number of non-zero elements in the row needs to be known. DIA format stores data with the offset of each diagonal from the main diagonal, only applied to matrices with diagonal structure usually from the application of stencils to regular grids. COO is the most intuitive format, storing row and column indices for all elements with nonzero values. This format has a reduced memory requirement that can store three times of the number of nonzero values. CSR format contains column indices and data like COO format, but the row indices are replaced with the pointer (ptr). The first element of ptr contains the index of the start row. The following elements indicate the number of data accumulated up to the corresponding row. CSC format is the column-major



Type	Range	Significant Figures
FP64	2^{-1022} to 2^{1023}	15
FP32	2^{-126} to 2^{127}	7
FP16	2^{-14} to 2^{15}	3

FIGURE 3. Floating-point format and precision level of FP64, FP32, and FP16.

version of CSR format. COO and CSR formats are widely used since they can be applied to various types of datasets.

C. TRANSPRECISION PARADIGM

Transpresicion is a paradigm that selects an appropriate precision type depending on the situation to reduce hardware resources and energy consumption [29]. Even if the precision level required as a result of a specific operation is determined already, it can be advantageous to change precision appropriately in the middle of the operation without accuracy degradation. Fig. 3 shows the most widely used floating-point types, double-precision (FP64), single-precision (FP32), and half-precision (FP16) defined in IEEE 754 standard [30].

The precision level of a floating-point value is a very important issue for hardware complexity and accuracy where a higher precision level can achieve better accuracy at the cost of hardware complexity. Accordingly, it is necessary to choose the appropriate precision level for energy-efficient operation without resource waste. With transprecision paradigm, since the precision changes depending on the target operation at run-time, PageRank is a good target to apply transprecision where the relative ranking is more important than each exact value and we can start the calculation with low precision and change to high precision at the optimal point.

III. DATA STORAGE REQUIREMENTS FOR TRANSPRECISION PageRank

In the proposed multi-mode SpMV accelerator for transprecision PageRank, the sparse matrix for SpMV operation is stored in the main memory, usually off-chip DRAM, and the portion of matrix data is loaded into the local buffer, usually implemented as an on-chip SRAM. This section compares sparse matrix storage formats for reduced memory requirements of the main memory. Also, we analyze the length of row index x, stored in the local buffer for the proposed accelerator.

A. SPARSE MATRIX STORAGE FORMAT IN MAIN MEMORY

For the sparse matrix, a compressed storage format can dramatically reduce memory requirements. Since memory-bounded algorithms such as PageRank have the bottleneck on memory bandwidth, a well-defined compressed storage format is an effective solution to decrease total clock cycles for PageRank calculations as well. Among sparse matrix storage formats introduced in Section II-B, COO and CSR formats are widely used due to their flexibility compared to ELL and DIA formats for specifically structured matrices. Based on the real-world graph datasets, we analyze the memory requirements of COO and CSR for the proposed transprecision PageRank operation.

Fig. 4 shows the process of compressing an example raw network dataset to COO and CSR format for transprecision PageRank. The raw network can be expressed in hyperlink matrix H, indicating a link between a node to node with information of outdegrees, as explained in Section II-A.

TABLE 2. Memory requirement comparison between COO and CSR format.

Dataset	Sparsity	Memory Size (bits)		Memory Reduction (%)
		COO	CSR	
Amazon	1.80E-05	98,790,160	67,611,680	31.5
Twitter	2.67E-04	141,451,920	87,472,976	38.2
Stanford	2.91E-05	184,999,760	120,020,784	35.1
Pokec	1.15E-05	2,449,805,120	1,522,132,800	37.9

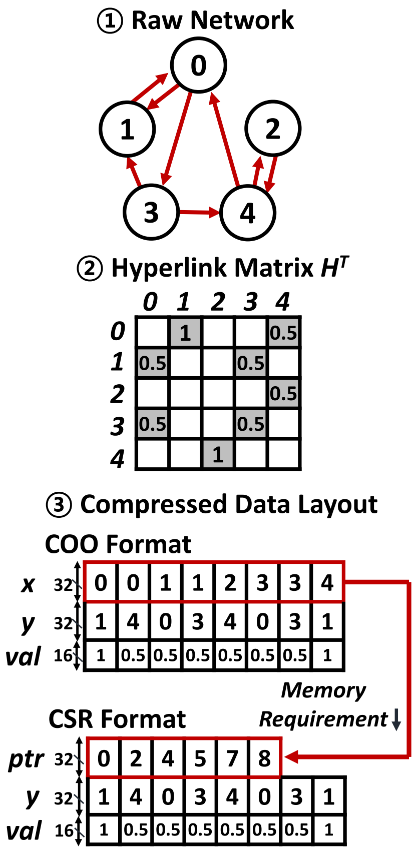


FIGURE 4. Comparison of COO and CSR format for transprecision PageRank.

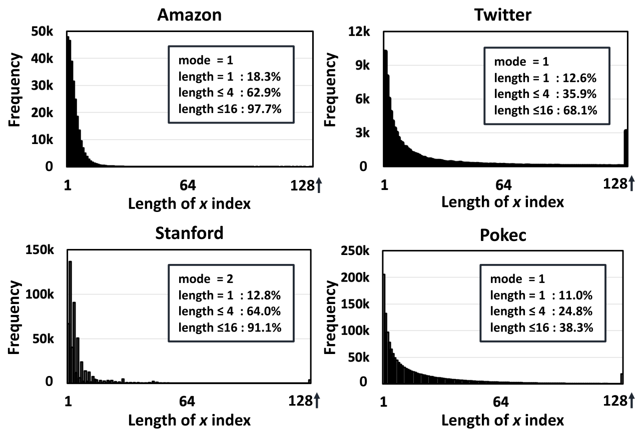


FIGURE 5. **Histogram for the length of *x* index on Amazon, Twitter, Stanford, and Pokec datasets.**

H^T can be compressed through COO and CSR formats as illustrated in Fig. 4 where x, y, and val each represent the row, column, and data of the matrix. x and y are 32-bit integer values that indicate each index number of nodes, while val is stored in memory with a 16-bit half-precision floating-point number.

Table 2 shows information about the required memory size of COO and CSR format for real-world graph datasets. COO format requires three times the number of nonzero data entries, known as the number of edges, for a list of three elements: x, y, and val. In CSR format, it has the same y and val entries as COO format, while x is replaced by ptr. The maximum number of entries in ptr is the one more entry to indicate the start row in addition to the number of nodes of the graph, which leads to a reduced memory requirement compared to COO format [31]. COO format is only advantageous for datasets with hyper-sparsity when the number of nodes is larger than the number of edges [32]. Since general real-world datasets have more edges than nodes, CSR format can reduce up to 38.2% of memory size compared to COO format. Accordingly, in this work, CSR format is selected to reduce memory efficiently for the data layout in the main memory.

B. ANALYSIS OF DATA IN LOCAL BUFFER

For the PageRank calculation, the portion of matrix data is loaded from the main memory into the local buffer where the double-buffering scheme is applied. Although the size of the local buffer affects the overall hardware complexity, too small-sized local buffer leads to frequent access to the main memory, usually off-chip DRAM, incurring increased latency and power consumption. In this work, the local buffer is determined to have 64 entries with 32-bit width and can store up to 128 entries for dual FP16 mode, two FP16 numbers per single 32-bit entry. After multiplication, the results with the same x index should be accumulated together and written to write buffer, which can be identified from ptr values in CSR format.

Algorithm 1 Operation Flow of Multi-Mode SpMV With CSR Format

```
1  Function SpMV ($ITER_{FP16}$, $ITER_{FP32}$,
     $newPR, oldPR, CSR\_graph$):
2  $ptr \leftarrow CSR\_graph.ptr$; $y \leftarrow CSR\_graph.y$; $val \leftarrow CSR\_graph.val$
3  $iter = 0$
4  do
5      for $i = 0, 2, 4, \ldots$ to $|V| - 1$ do
6          $newPR[ptr.x[i]] += val[i] \cdot oldPR_y[i]$
7          $newPR[ptr.x[i+1]] += val[i+1] \cdot oldPR_y[i+1]$
8      end
9      $iter++$
10 while $iter < ITER_{FP16}$;
11 $iter = 0$
12 do
13     for $j = 0, 1, \ldots$ to $|V| - 1$ do
14         $newPR[ptr.x[j]] += val[j] \cdot oldPR_y[j]$
15     end
16     $iter++$
17 while $iter < ITER_{FP32}$;
```

Fig. 5 shows the distribution of the length of x index where the length of x index indicates the number of data to be accumulated together with the same x value. As illustrated in Fig. 5, the histogram shows a left-leaning distribution and almost data have short x index lengths nearby 1 or 2. Accordingly, the local buffer with 64 entries has the data values with the different x values, that is from different rows, for both dual FP16 mode and single FP32 mode. The multi-mode SpMV accelerator should carefully handle the accumulation of multiplied data with different x indices while processing the data in the local buffer.

IV. PROPOSED MULTI-MODE SpMV ACCELERATOR

The multi-mode SpMV accelerator is proposed for transprecision PageRank in that it has a huge impact on the overall latency and hardware complexity. To operate both in dual FP16 mode and single FP32 mode, transprecision multiplier and dual-issue accumulator are proposed. In this section, we explain the architecture of the proposed accelerator and its operation for multi-mode support.

A. MULTI-MODE SpMV ARCHITECTURE FOR TRANSPRECISION PageRank

For the transprecision PageRank calculation, both half-precision (FP16) and single-precision (FP32) are used as described in Algorithm 1. The last term of (2) in Section II-A indicates SpMV operation, which aims to be accelerated through hardware. The graph H^T is stored in CSR format as denoted in Fig. 4. The optimal number of iterations of FP16 and FP32 that minimize the number of errors and operation cycles for each dataset, $ITER_{FP16}$ and $ITER_{FP32}$, can be determined through software profiling. At first, the proposed SpMV accelerator operates in dual FP16 mode, calculating two FP16 multiplications at the same time. When the number of iterations reaches $ITER_{FP16}$, the operation mode changes to single FP32 mode where one FP32 multiplication is executed at a time during the following $ITER_{FP32}$ iterations.

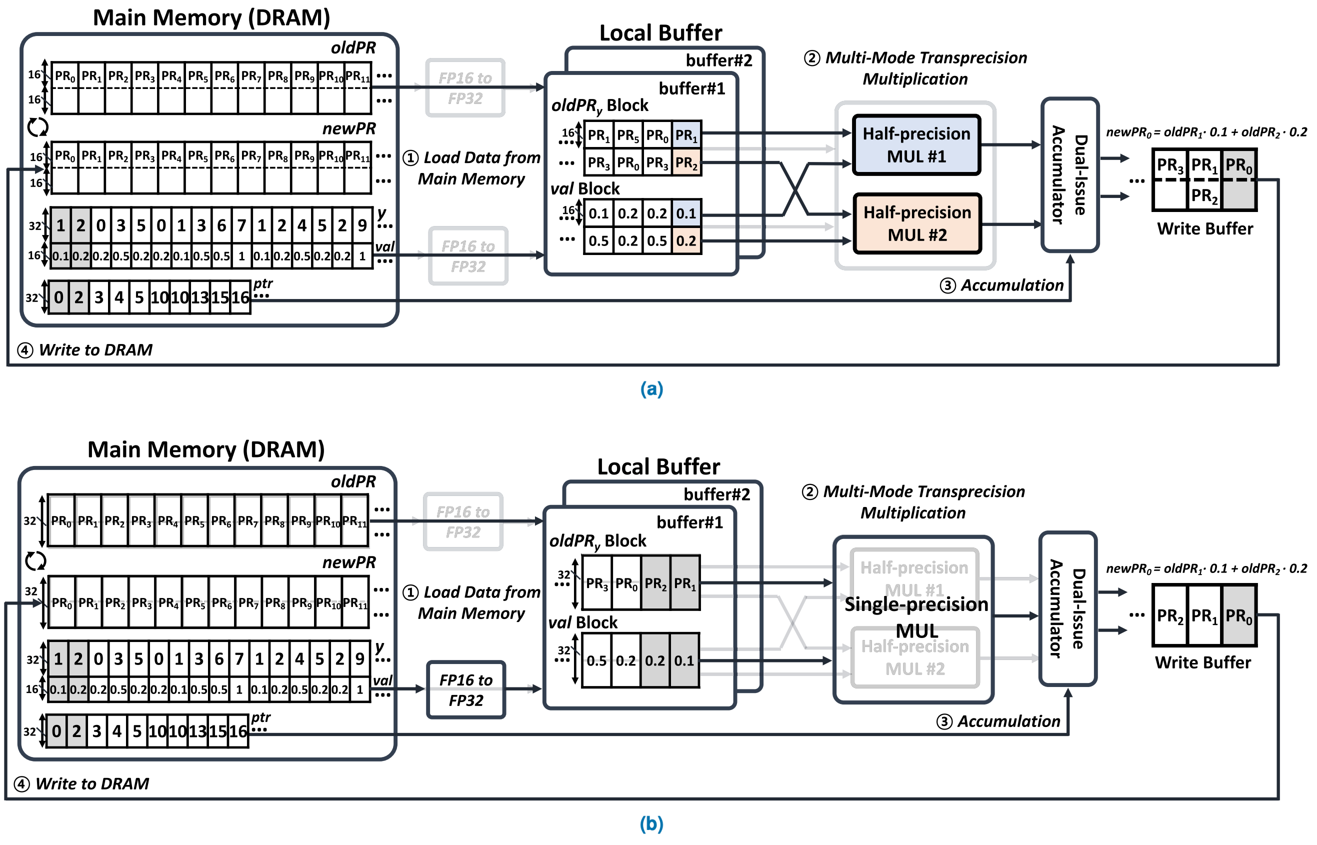


FIGURE 6. Architecture of multi-mode SpMV accelerator operating (a) in dual FP16 mode and (b) in single FP32 mode.

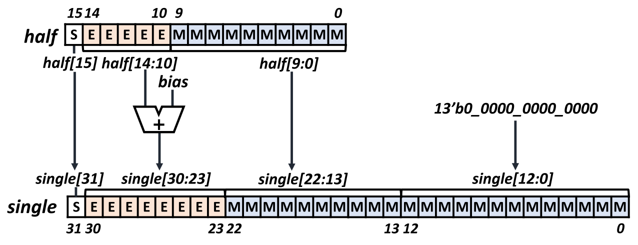


FIGURE 7. Datapath of FP16 to FP32 converter.

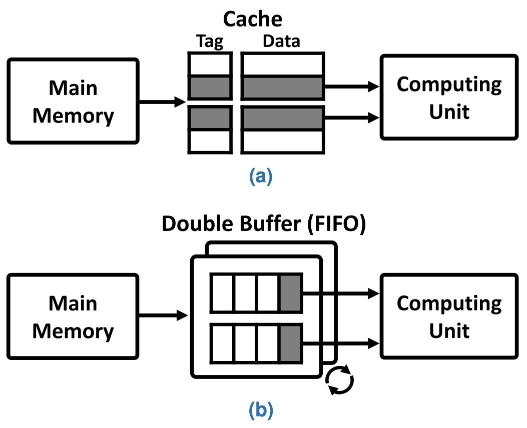


FIGURE 8. Local buffer types. (a) Cache. (b) Double buffer (FIFO).

Fig. 6 shows the overall architecture of the proposed multi-mode SpMV accelerator which supports two operation modes, dual FP16 mode and single FP32 mode. Initially, in the main memory, data is stored in FP16 based on CSR format. Local buffer is prepared to load data from main memory for the next computation steps, considering the general System-on-Chip environments with various DRAMs such as HBM, DDR2/3/4 with different specifications and bandwidths. With the graph data, *val* and previous PageRank values, *oldPR* from the local buffer, the multi-mode multiplier operates in dual FP16 or single FP32. The multiplication results are accumulated according to the *x* values obtained from *ptr* values, then updated as *newPR* to the main memory with the write buffer.

In dual FP16 mode, as illustrated in Fig. 6(a), data used for the multiplier is represented in 16-bit half-precision. Since the local buffer has 32-bit bitwidth, two 16-bit numbers in half-precision can be stored in one write buffer entry for multiplication. After multiplication, up to two results can be stored in one 32-bit entry of the write buffer. When the update of *newPR* ends for the given iteration, two regions for *oldPR* and *newPR* change their roles for the next iteration. Also, when the proposed SpMV accelerator changes its operation mode from dual FP16 mode to single FP32 mode, 16-bit half-precision *val* and *oldPR* from the main memory should be converted into 32-bit single-precision with the simple conversion unit as shown in Fig. 7. In single FP32 mode as illustrated in Fig. 6(b), one result from multiplication is stored into write buffer.

Local buffer can be implemented both in cache and double buffer in First-In, First-Out (FIFO) types as illustrated in Fig. 8. Cache is suitable when the same data is repeatedly requested, so-called temporal/spatial localities, at the cost of additional memory requirement for tag. In the proposed architecture, data in main memory is arranged in the order of *x* index of *ptr* and PageRank values in local buffer corresponding to *y* index have little temporal/spatial localities with the practical cache size. In the proposed architecture, since data is loaded in local buffer and sequentially transmitted to the computing unit, FIFO is preferred over cache and implemented in the form of double buffer to hide the long latency of data transfer from main memory while providing the data to computing unit at the same time.

B. MULTI-MODE TRANSPRECISION MULTIPLIER

In the proposed multi-mode SpMV accelerator, the multiplier which can execute two FP16 multiplications in dual FP16 mode and one FP32 multiplication in single FP32 mode is required. In a floating-point multiplier, the sign bit is gained with XOR operation, and the exponent bit can be easily calculated with simple addition or subtraction. However, since the mantissa bit should be multiplied, this multiplier becomes the biggest overhead of the floating-point multiplier. To minimize hardware resources, we present the dual-mode multiplier which can share the hardware resource for FP16 and FP32 as indicated in (3).

$$\begin{aligned}
xSig &= xSig_H \cdot 2^{\frac{n}{2}} + xSig_L \\
ySig &= ySig_H \cdot 2^{\frac{n}{2}} + ySig_L \\
xSig \cdot ySig &= (xSig_H \cdot 2^{\frac{n}{2}} + xSig_L) \cdot (ySig_H \cdot 2^{\frac{n}{2}} + ySig_L) \\
&= xSig_H \cdot ySig_H \cdot 2^n + xSig_H \cdot ySig_L \cdot 2^{\frac{n}{2}} \\
&\quad + xSig_L \cdot \, ySig_H \cdot 2^{\frac{n}{2}} + xSig_L \cdot ySig_L \qquad (3)
\end{aligned}$$

where *xSig* and *ySig* are significands from 32-bit single-precision floating-point numbers and the method to multiply *xSig* and *ySig* by dividing them into higher and lower parts is presented [33]. Significand is the extended form of mantissa bits with 1-bit 1. In floating-point multiplication, the significand number is multiplied instead of mantissa bits to control overflow and underflow. Since *xSig* and *ySig* can be expressed as the sum of higher parts and lower parts, the multiplication of significands can be rewritten by adding the products of divided parts.

Fig. 9 shows the datapath of the Type I and Type II multiplier. Type I multiplier consists of one FP32 multiplier and two FP16 multipliers. The multipliers contain status flag logic although not indicated in Fig. 9 for simplicity. Since the operands of the multipliers, *oldPR* and *val* are always

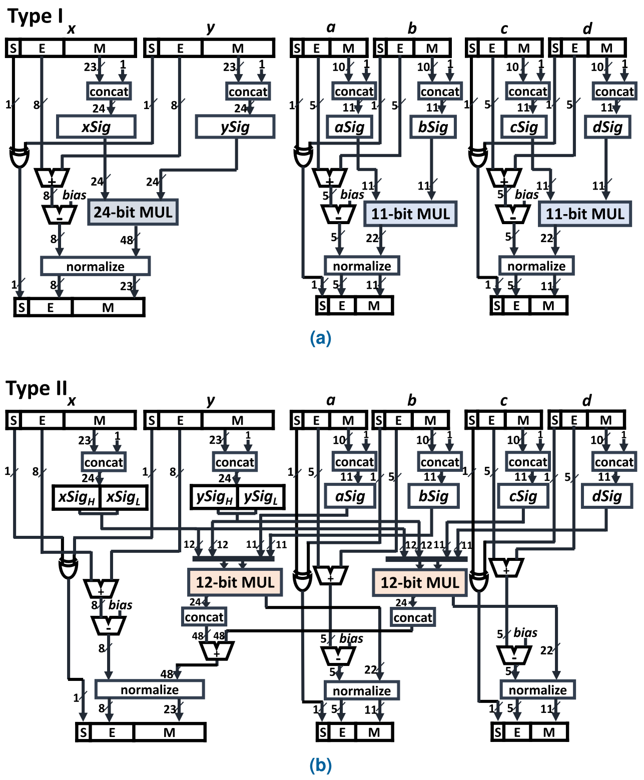


FIGURE 9. Datapath of Type I and Type II multiplier. (a) Type I multiplier: two FP16 multipliers and one FP32 multiplier. (b) Type II multiplier: multi-mode transprecision multiplier for dual FP16 mode and single FP32 mode by sharing two 12-bit multipliers.

within 0 to 1, overflow does not occur [34]. In the case of underflow, the underflowed value is treated as 0. Since the proposed multi-mode SpMV accelerator should be capable of performing two FP16 multiplications at the same time, or one FP32 multiplication, this Type I multiplier proposes the simple method using one FP32 multiplier and two FP16 multipliers without considering hardware overhead. Type II is a multi-mode transprecision multiplier designed according to (3), sharing multipliers between two modes. Since the significand of FP32 has 24-bit and FP16 has 11-bit, Type I requires one 24-bit multiplier and two 11-bit multipliers. In Type II, the 24-bit multiplier can be replaced by two 12-bit multipliers, while these are enough to calculate 11-bit significands of FP16. Added multiplexers enable control of operation modes of the multiplier, and multiplication of 2^n can be easily implemented with concatenation. Type II multiplier is finally applied in multi-mode SpMV accelerator for transprecision multiplication, reducing hardware resource compared to general Type I multiplier.

C. DUAL-ISSUE ACCUMULATOR

Since dual FP16 mode executes two multiplications in parallel, the accumulator should treat results from multi-mode transprecision multiplier at the same time. Dual results of transprecision multiplier are named *A* and *B*, from FP16 multipliers *MUL A* and *MUL B*. The *x* indices of *A* and *B*

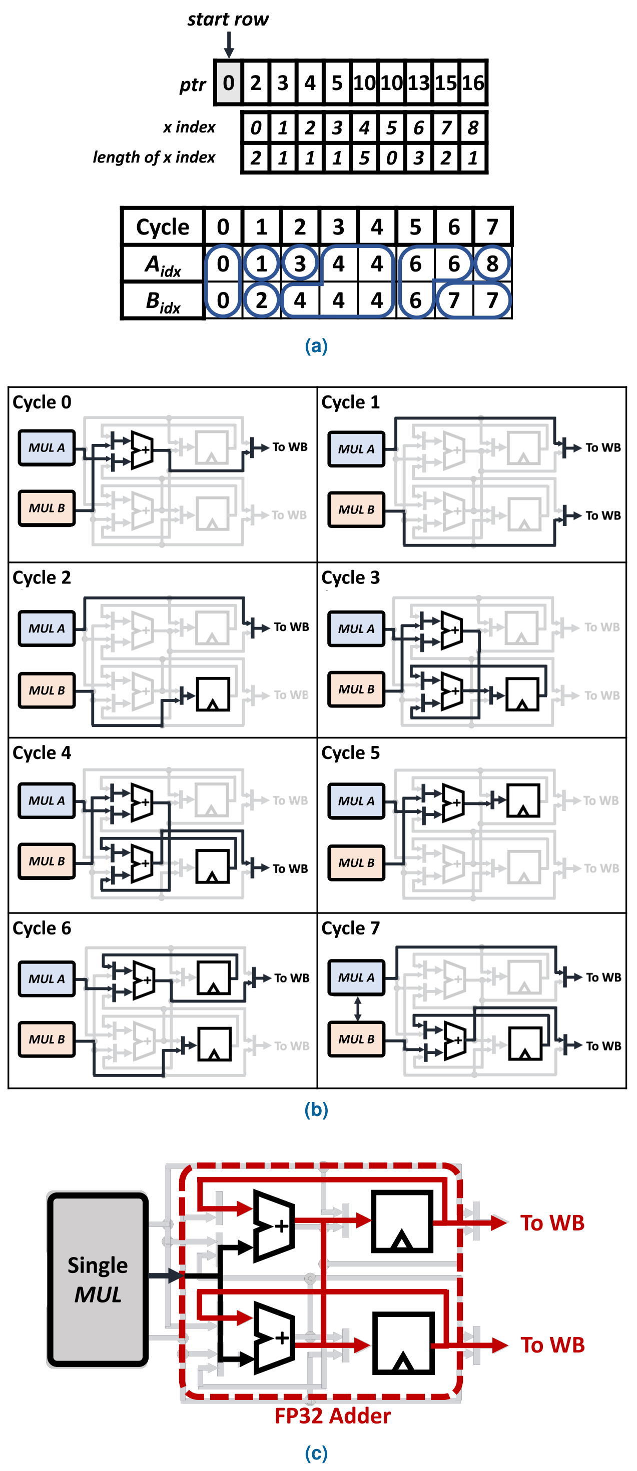


FIGURE 10. Architecture of dual-issue accumulator. (a) *x* index and length of *x* index of multiplier A, B output decoded from *ptr* in dual FP16 mode. (b) Datapath of dual-issue accumulator per cycle in example of (a). (c) Datapath of dual-issue accumulator in single FP32 mode.

should be different according to the analysis of data in the local buffer as shown in Section III-B. To treat data with different indices, dual-issue accumulator is designed to cover all possible situations.

Fig. 10 explains the architecture of dual-issue accumulator with some examples. Fig. 10(a) shows the example of information prepared in CSR format. Since CSR format consists of *ptr* instead of storing entire x indices, the length of each x index can be known with subtraction of nearby *ptr* value. In dual FP16 mode, two multiplier outputs, A and B enter into the accumulator per one cycle. With this information, x indices of A and B can be decoded in A_{idx} and B_{idx}. In cycle 7, A and B change their value to accumulate data with index 7 at the same position as the last cycle 6.

Fig. 10(b) shows the datapath of the dual-issue accumulator using the example of Fig. 10(a). The accumulator consists of two FP16 adders and two 16-bit registers. Inputs of adders and registers can be selected by multiplexers. In cycles 0, 3, 4, and 5, since x indices of A and B, A_{idx} and B_{idx} are the same, they should be added to each other. After adding A and B, the result will be stored in the register or passed to write buffer according to the length of the x index. If the x index has length 2 like in cycle 0, the added result can be updated to write buffer immediately. However, when the length is more than 2, the accumulator should check the end of the same x index, then store the result in the register or update to write buffer.

In cycles 1, 2, 6, and 7, since A_{idx} and B_{idx} are different, they should be treated separately. When the length of A_{idx} and B_{idx} are all 1, they can directly dual-issued to write buffer like in cycle 1. If the length is more than 1, the value stored in the register should be added to the input value. Added results can be updated to write buffer when the calculation of the same index is finished. In cycle 7, values of A and B are exchanged to accumulate value with a lower register. Example of Fig. 10(b) covers all possible modes of accumulator in dual FP16 mode.

Fig. 10(c) depicts the datapath of dual-issue accumulator in single FP32 mode, reusing datapath of dual FP16 mode. Since multiplier output of single FP32 mode has one result per cycle, there is no need to consider various operation modes as in FP16 mode. Two 16-bit registers can be reused to store 32-bit single-precision numbers, while two FP16 adders can also be reused for the FP32 adder. Since the floating-point adder consists of exponent logic and mantissa adder with shifter, the adder and shifter can easily work in dual mode [35]. The results of the FP32 adder can be stored in two registers separately or dual-issued in two 16-bit values to write buffer.

V. EXPERIMENTAL RESULTS

For real-world graph datasets selected from SNAP as denoted in Table 1, the proposed transprecision PageRank algorithm is analyzed with in-house simulator in C++ language. To support transprecision, transpoint is newly defined which determines the time to change the precision level of operation from FP16 to FP32. Then, threshold determines when the iterative PageRank calculation can stop. Euclidean distance between *oldPR* and *newPR* is used as the difference of PageRank operation, while transpoint and threshold work

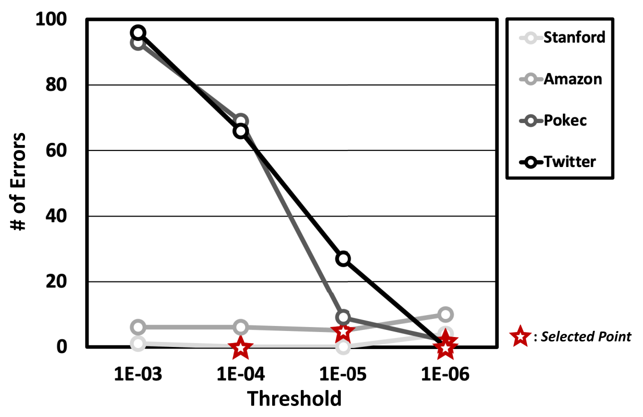


FIGURE 11. The number of errors of top-100 ranking in *single* calculation.

as the control point of operation. This FP16 to FP32 transprecision PageRank is verified with a simulator in the aspects of accuracy and speed, comparing the results of transprecision (*trans*) with FP16 half-precision (*half*) and FP32 single-precision (*single*).

Fig. 11 shows the graph of the number of errors in the top-100 ranking in *single* calculation according to the change of threshold from 1E-3 to 1E-6. The number of errors is determined by counting the number of different rankings compared to the top-100 rank result of calculated in double, which is the most precise calculation result. Then, we selected the threshold point which has the lowest number of errors for each dataset to analyze transpoint. For Twitter and Pokec datasets, since threshold has a large effect on the number of errors, the lowest threshold 1E-6 point is selected. Stanford and Amazon datasets have similar numbers of errors according to the threshold, with 1E-04 and 1E-05 showing the lowest error for each.

To evaluate the throughput in PageRank calculation, operation cycles are newly defined as the sum of the number of iterations in *single* and halved value of the number of iterations in *half* since two FP16 operations can be processed at once in the proposed multi-mode SpMV accelerator. The graphs in Fig. 12 show the number of errors, the number of iterations in FP16 and FP32, and operation cycles with the change of transpoint in the selected threshold point from Fig. 11. The number of iterations is depicted in the bar graph, indicating the proportion of *half* and *single* according to transpoint. The optimal point is indicated with the number of *half* and *single* iterations in $ITER_{FP16}$ and $ITER_{FP32}$. When the transpoint is set nearby the threshold, the proportion of iterations calculated in *half* increases. In contrast, when the transpoint is set to 1E+0, the entire PageRank is calculated in *single*.

Table 3 summarizes the number of errors in *half* and *trans*. *trans* shows an extremely low number of errors compared to *half*. When the entire PageRank computation is executed in *half*, it can lead to convergence failure with a large number of errors due to the low precision level of FP16. However, when the appropriate transpoint is set near the threshold without any

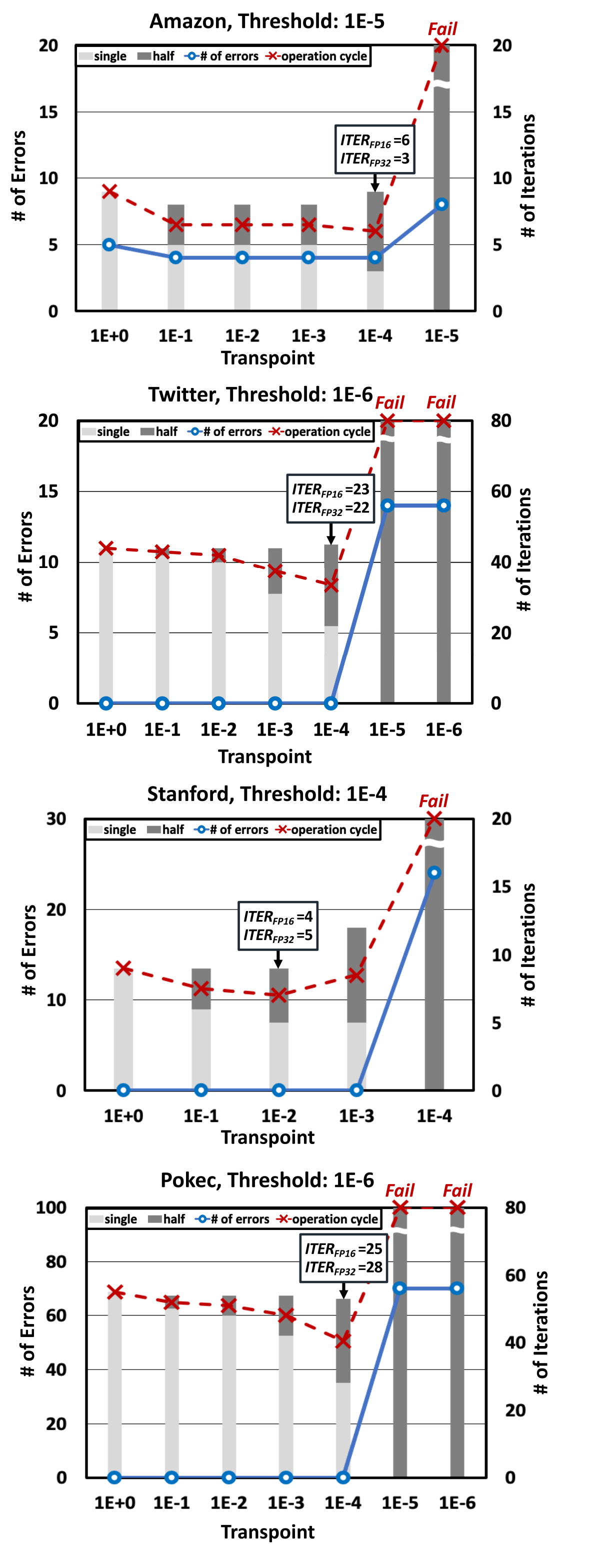


FIGURE 12. The number of errors, operation cycles, and *half*, *single* portion in the number of iterations according to transpoint.

convergence failure, it maintains a similar number of errors in *single*.

In PageRank calculation, the required number of floating-point operations (FLOPs) is given by the product of the

TABLE 3. The number of errors in *half* and *trans*.

Amazon		Twitter		Stanford		Pokec	
half	*trans*	*half*	*trans*	*half*	*trans*	*half*	*trans*
8	4	23	0	24	0	70	0

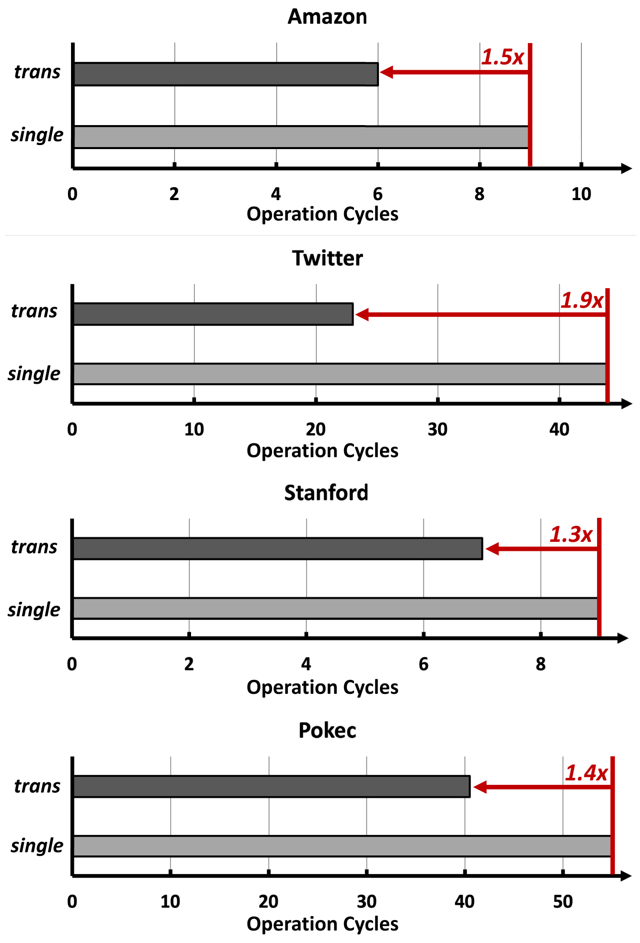


FIGURE 13. Operation cycles comparison of *trans* and *single*.

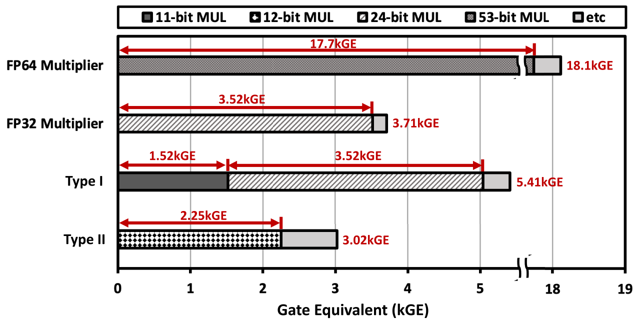


FIGURE 14. Gate count comparison of FP64 multiplier, FP32 multiplier, Type I and Type II multiplier.

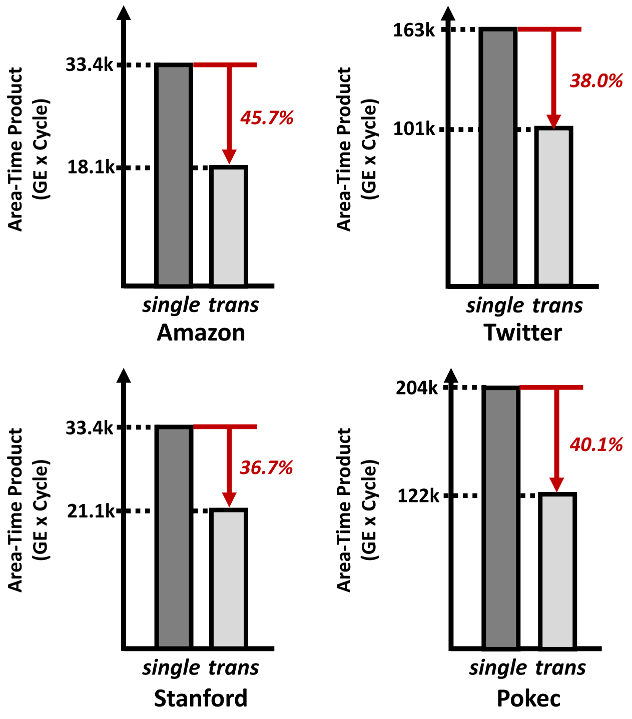


FIGURE 15. Area-time product comparison of *single* and *trans*.

number of iterations and twice the number of edges in each dataset since it requires one multiplication and one accumulation for an edge. For general-purpose CPUs and GPUs, overall throughput for PageRank calculation depends on the floating-point calculation capability of the computing units. Due to the multi-mode SpMV architecture where two FP16 operations can be processed at once, the proposed transprecision architecture can improve the throughput of the PageRank calculation. Fig. 13 shows the comparison between operation cycles of *trans* and *single*. In *trans*, since the operation cycles in *half* iterations can be halved, it shows the result of speedup up to 1.9× compared to *single*. The proposed multi-mode SpMV accelerator enables the speedup of *trans*, supporting two FP16 multiplications at the same time in dual FP16 mode.

Multipliers introduced in Section IV-B are synthesized in 28nm CMOS process at 1GHz clock frequency with FP64 double-precision multiplier and FP32 single-precision multiplier for comparison. As illustrated in Fig. 14, FP64 multiplier has the largest gate count of 18.1kGE (NAND2 Gate Equivalent) mainly due to a 53-bit significand number to be multiplied. Type I multiplier consists of one FP32 multiplier and two FP16 multipliers, while Type II is multi-mode transprecision multiplier used for the proposed SpMV accelerator sharing 12-bit multipliers. Type II multiplier has the lowest gate count of 3.02kGE compared to others, even lower than the FP32 multiplier. Splitting the multiplier by dividing the mantissa part of the floating-point number is effective to reduce hardware resources.

The area-time product of *single* and *trans* is illustrated in Fig. 15 where the proposed transprecision PageRank shows the area-time product reduction of 36.7-45.7% compared to

TABLE 4. Comparison with prior works.

Type	MICRO' 19 [12]	ASAP' 19 [13]	ASP-DAC' 21 [14]	**This Work**
Max. # of Nodes	2E+09	6E+04	2E+05	**2E+05**
Min. Sparsity	5.5E-03	8.3E-05	5E-05	**2.91E-05**
Sparse Matrix Format	COO	CSR	COO	**CSR**
Precision	FP32	FP32/FP64	20/22/24/26-bit Fixed-point	**FP16 to FP32 Transprecision**
Error	-	-	20-bit: 25-90% 26-bit: 1-25%	**0-4%**

the uniform single-precision system. For combinational logic, since the power consumption is proportional to the number of logic gates, the area-time product is a good measure of the energy savings of the proposed architecture. Additionally, regarding the number of memory accesses, which is one of the major contributions to energy dissipation, a reduced amount of memory accesses is required in the proposed architecture since FP16 operations require half of the data amount from main memory compared to FP32.

Table 4 summarizes the comparison of prior works which designed SpMV accelerator. COO format is applied to [12] and [14], which has an intuitive format to utilize. CSR format is selected in this work and [13] to reduce memory size compared to COO format for common datasets which are not hyper-sparse. The SpMV accelerator in [14] selects 20-bit to 26-bit fixed-point precision for PageRank acceleration, while working on the newly proposed FP16 to FP32 transprecision scheme has shown higher accuracy.

VI. CONCLUSION

The multi-mode SpMV accelerator for transprecision PageRank is proposed, operating in dual FP16 mode and single FP32 mode. Four real-world datasets are used in CSR sparse matrix compression format, with up to 38% memory reduction compared to COO format. To perform two FP16 SpMV at the same time, a dual-issue accumulator and multi-mode transprecision multiplier are designed. Hardware resources can be reused between two operation modes to reduce overhead. The proposed FP16 to FP32 transprecision PageRank algorithm is verified for aspects of accuracy and speed. Transprecision shows an extremely low number of errors compared to FP16 half-precision, with an error rate of 0-4%. Compared to single-precision PageRank without transprecision, the proposed transprecision PageRank shows 1.3×-1.9× speedup.

REFERENCES

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep. 1999-66, Nov. 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[2] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized PageRank on MapReduce," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2011, pp. 973–984.

[3] W. Lin, "Distributed algorithms for fully personalized PageRank on large graphs," in *Proc. World Wide Web Conf.*, May 2019, pp. 1084–1094.

[4] H. Eedi, S. Peri, N. Ranabothu, and R. Utkoor, "An efficient practical non-blocking PageRank algorithm for large scale graphs," in *Proc. 29th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Mar. 2021, pp. 35–43.

[5] T. H. Haveliwala, "Topic-sensitive PageRank," in *Proc. 11th Int. Conf. World Wide Web*, May 2002, pp. 517–526.

[6] I. M. Kamal, H. Bae, L. Liu, and Y. Choi, "Identifying key resources in a social network using f-PageRank," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 397–403.

[7] H. Zhao, X. Xu, Y. Song, D. L. Lee, Z. Chen, and H. Gao, "Ranking users in social networks with motif-based PageRank," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2179–2192, May 2021.

[8] S. Bo, G. Wenfeng, and L. Yan, "Discovery of key nodes in social networks premised on PageRank algorithm," in *Proc. Asia–Pacific Conf. Image Process., Electron. Comput. (IPEC)*, Apr. 2020, pp. 26–28.

[9] H. S. Sint and K. K. Oo, "Consumer trust recommendation in online social commerce," in *Proc. IEEE 8th Global Conf. Consum. Electron. (GCCE)*, Oct. 2019, pp. 436–438.

[10] S. Payandeh and E. Chiu, "Application of modified PageRank algorithm for anomaly detection in movements of older adults," *Int. J. Telemed. Appl.*, vol. 2019, pp. 1–9, Mar. 2019.

[11] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2004, pp. 404–411.

[12] F. Sadi, J. Sweeney, T. M. Low, J. C. Hoe, L. Pileggi, and F. Franchetti, "Efficient SpMV operation for large and highly sparse matrices using scalable multi-way merge parallelization," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 347–358.

[13] B. Sigurbergsson, T. Hogervorst, T. D. Qiu, and R. Nane, "Sparstition: A partitioning scheme for large-scale sparse matrix vector multiplication on FPGA," in *Proc. IEEE 30th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2019, pp. 51–58.

[14] A. Parravicini, F. Sgherzi, and M. D. Santambrogio, "A reduced-precision streaming SpMV architecture for personalized PageRank on FPGA," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 378–383.

[15] J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile Wimax and 3GPP-LTE," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2009, pp. 487–490.

[16] D. Kim, W. Byun, Y. Ku, and J.-H. Kim, "High-speed visual target identification for low-cost wearable brain-computer interfaces," *IEEE Access*, vol. 7, pp. 55169–55179, 2019.

[17] J. Song, Y. Cho, J.-S. Park, J.-W. Jang, S. Lee, J.-H. Song, J.-G. Lee, and I. Kang, "7.1 An 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 130–132.

[18] A. S. Molahosseini and H. Vandierendonck, "Half-precision floating-point formats for PageRank: Opportunities and challenges," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2020, pp. 1–7.

[19] T. Grützmacher and H. Anzt, "A modular precision format for decoupling arithmetic format and storage format," in *Proc. Eur. Conf. Parallel Process.* Cham, Switzerland: Springer, 2018, pp. 434–443.

[20] T. Grutzmacher, H. Anzt, F. Scheidegger, and E. S. Quintana-Orti, "High-performance GPU implementation of PageRank with reduced precision based on mantissa segmentation," in *Proc. IEEE/ACM 8th Workshop Irregular Appl., Archit. Algorithms (IA3)*, Nov. 2018, pp. 61–68.

[21] W. Kim, J. Lee, S. Kim, and J.-H. Kim, "Multi-mode transprecision sparse matrix-vector multiplication engine for PageRank," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Feb. 2022, pp. 1–3.

[22] J. Leskovec and A. Krevl, "SNap datasets: Stanford large network dataset collection," Jun. 2014. [Online]. Available: http://snap.stanford.edu/data

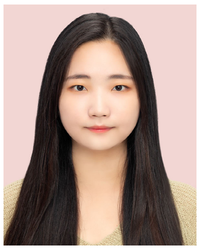
[23] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate club: An API oriented open-source Python framework for unsupervised learning on graphs," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 3125–3132.

[24] N. Perra, V. Zlatić, A. Chessa, C. Conti, D. Donato, and G. Caldarelli, "PageRank equation and localization in the WWW," *EPL Europhys. Lett.*, vol. 88, no. 4, Nov. 2009, Art. no. 48002.

[25] M. Franceschet, "PageRank: Standing on the shoulders of giants," *Commun. ACM*, vol. 54, no. 6, pp. 92–101, Jun. 2011.

[26] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, nos. 1–7, pp. 107–117, Apr. 1998.
[27] T. Wu, B. Wang, Y. Shan, F. Yan, Y. Wang, and N. Xu, "Efficient PageRank and SpMV computation on AMD GPUs," in *Proc. 39th Int. Conf. Parallel Process.*, Sep. 2010, pp. 81–89.
[28] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA Corp., Tech. Rep. NVR-2008-004, Dec. 2008.
[29] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomas, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1105–1110.
[30] W. Kahan, "IEEE standard 754 for binary floating-point arithmetic," *Lect. Notes Status IEEE*, vol. 754, no. 1776, p. 11, May 1996.
[31] T. Kelly, "Programming workbench: Compressed sparse row format for representing graphs," *Login Usenix Mag.*, vol. 45, no. 4, pp. 77–79, 2020.
[32] A. Buluc and J. R. Gilbert, "On the representation and multiplication of hypersparse matrices," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–11.
[33] K. Manolopoulos, D. Reisis, and V. A. Chouliaras, "An efficient multiple precision floating-point multiplier," in *Proc. 18th IEEE Int. Conf. Electron., Circuits, Syst.*, Dec. 2011, pp. 153–156.
[34] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, "Exploiting the block structure of the web for computing pagerank," Stanford InfoLab, Tech. Rep. 2003-17, 2003. [Online]. Available: http://ilpubs.stanford.edu:8090/579/
[35] M. K. Jaiswal, B. S. C. Varma, and H. K. H. So, "Architecture for dual-mode quadruple precision floating point adder," in *Proc. IEEE Comput. Soc. Annu. Symp. (VLSI)*, Jul. 2015, pp. 249–254.

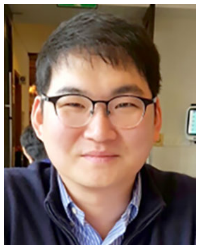
WHIJIN KIM (Graduate Student Member, IEEE) received the B.S. degree in electronic and electrical engineering from Ewha Womans University, Seoul, South Korea, in 2021, where she is currently pursuing the M.S. degree with the Digital System Architecture Laboratory. Her current research interests include graph processing, high-performance computing, neural network processing, processing-in-memory, and near-memory processing.



HANA KIM (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering and the M.S. degree in electronic and electrical engineering from Ewha Womans University, South Korea, in 2020 and 2022, respectively, where she is currently pursuing the Ph.D. degree with the Digital System Architecture Laboratory. Her current research interests include data type, deep neural network (DNN) accelerator, system-on-chip, and digital system architecture design.

JIHYE LEE (Graduate Student Member, IEEE) received the B.S. degree in electronic and electrical engineering from Ewha Womans University, Seoul, South Korea, in 2021, where she is currently pursuing the M.S. degree with the Digital System Architecture Laboratory. Her current research interests include domain-specific processor, system-on-chip, secure microarchitecture, near-memory processing, and processing-in-memory.

HYUNJI KIM (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering and the M.S. degree in electronic and electrical engineering from Ewha Womans University, South Korea, in 2019 and 2021, respectively, where she is currently pursuing the Ph.D. degree with the Digital System Architecture Laboratory. Her current research interests include programmable scalable deep neural processing unit.



JI-HOON KIM (Senior Member, IEEE) received the B.S. (summa cum laude) and Ph.D. degrees in electrical engineering and computer science from KAIST, Daejeon, South Korea, in 2004 and 2009, respectively.

In 2009, he joined Samsung Electronics, Suwon, South Korea, as a Senior Engineer, and worked on next-generation architecture for 4G communication modem system-on-chip (SoC). He was an Associate Professor with the Department of Electronics Engineering, Chungnam National University, Daejeon, from 2010 to 2016. In 2018, he joined the Department of Electronic and Electrical Engineering, Ewha Womans University, Seoul, South Korea, as a Faculty Member, where he is currently a Professor. His current research interests include CPU microarchitecture, domain-specific SoC, and deep neural network accelerators.

Dr. Kim has served on the Technical Program Committee and Organizing Committee for various international conferences, including the IEEE International Conference on Computer Design (ICCD), IEEE Asian Solid-State Circuits Conference (A-SSCC), and IEEE International Solid-State Circuits Conference (ISSCC). He was a recipient of the Best Design Award at Dongbu HiTek IP Design Contest, in 2007, and the First Place Award at the International SoC Design Conference Chip Design Contest, in 2008.