Scalable Low-Cost Sorting Network with Weighted Bit-Streams

Brady Prince†, M.Hassan Najafi*, and Bingzhe Li†
†Oklahoma State University
*University of Louisiana at Lafayette
{brady.prince, bingzhe.li}@okstate.edu; najafi@louisiana.edu

2023 24th International Symposium on Quality Electronic Design (ISQED) | 979-8-3503-3475-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISQED57927.2023.10129357

***Abstract*—Sorting is a fundamental function in many applications from data processing to database systems. For high performance, sorting-hardware based sorting designs are implemented by conventional binary or emerging stochastic computing (SC) approaches. Binary designs are fast and energy-efficient but costly to implement. SC-based designs, on the other hand, are area and power-efficient but slow and energy-hungry. So, the previous studies of the hardware-based sorting further faced scalability issues. In this work, we propose a novel scalable low-cost design for implementing sorting networks. We borrow the concept of SC for the area- and power efficiency but use weighted stochastic bit-streams to address the high latency and energy consumption issue of SC designs. A new lock and swap (LAS) unit is proposed to sort weighted bit-streams. The LAS-based sorting network can determine the result of comparing different input values early and then map the inputs to the corresponding outputs based on shorter weighted bit-streams. Experimental results show that the proposed design approach achieves much better hardware scalability than prior work. Especially, as increasing the number of inputs, the proposed scheme can reduce the energy consumption by about 3.8% - 93% compared to prior binary and SC-based designs.**

***Index Terms*—Sorting, Stochastic computing, Low cost**

I. Introduction

Sorting [1], [2] is one of the fundamental tasks in computer science. Sorting operations can be found in many applications including database [3], data mining [4], image processing [5], video encoding [6], etc. While most sorting algorithms are realized in software, there is a growing number of real-time applications that require a higher speed than what is offered by generalized computer circuitry. For these instances specialized circuitry is used. Hardware-based sorting networks are realized with field-programmable gate arrays (FPGAs) or Application-specific integrated circuit (ASIC) as a high-performance and energy-efficient alternative to software-based sorting solutions.

While hardware-based sorting networks can provide significant speed boosts over software solutions, they come with the drawbacks of being large and power hungry. This is a major problem for small embedded systems where signal and image processing are often used. These systems have hard limits on size and power making the implementation of sorting networks impractical. Reducing the area and power consumption of sorting networks is an important goal as it makes them viable for a much wider range of applications.

A recent study [7], [8] reduced the area and power consumption of the sorting networks by employing stochastic computing (SC) [9], [10]. Stochastic computing is a technology by using approximation to reduce the hardware cost in terms of area and power consumption. SC has been used in many fields including but not limited to artificial neural networks and image processing [11]–[17]. SC-based sorting networks are much more hardware-efficient than the sorting networks implemented based on the traditional binary computing [2], [18]. The hardware area and power cost reduces by more than 92% for a 256-input sorting network circuit [7]. However, SC-based implementations suffer from a major drawback: significant increase in the latency and energy consumption. This drawback comes from how inputs are represented and compared. In traditional sorting networks, inputs are represented and processed in positional binary format. However, in SC-based networks inputs are represented using long stochastic bit-streams. These bit-streams scale exponentially in length taking many cycles to process. The long processing time leads to a high total energy consumption even though the network uses a lower continuous power than the traditional binary-based sorting networks.

In this work, we propose a novel scheme called *lock-and-swap sorting network* (LAS) by using weighted bit-streams. The proposed technique takes advantage of the bit-stream functionality of SC but keeps the binary weighting of traditional binary computing. LAS uses swap and lock units to perform comparison between weighted bit-streams inputs. LAS using weighted bit-streams can decrease the latency from 2^N cycles required by SC-based sorting network to N cycles (where N is the number of bit precision of the inputs). Finally, the LAS scheme achieves much better scalability and significantly lower hardware cost compared to the SC based and traditional binary implementations.

The contributions of this paper are fourfold: **I.** We propose a novel scalable low-cost design for implementing sorting networks based on the weighted stochastic bit-streams. **II.** A new lock and swap (LAS) unit is proposed to sort weighted bit-streams. **III.** A weighted bit-stream generator is proposed to generate SC based bit-streams for the sorting networks. **IV.** The experimental results indicate that the proposed scheme has much better hardware scalability than prior conventional binary and stochastic computing based designs.

The reminder of this paper is organized as follows: Section II discusses the background on sorting networks and SC. Section III demonstrates the proposed sorting network design. The synthesis results of different sorting network are reported and compared in Section IV. Finally, conclusions are drawn in

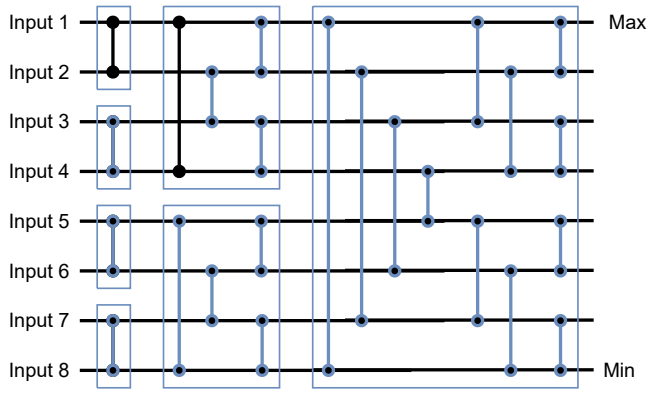

Fig. 1. 8-input Bitonic sorting network [2].

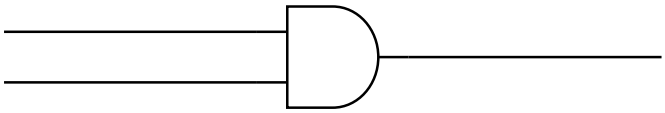

Fig. 2. Example of a stochastic multiplication circuit.

Section V.

II. Background

A. Stochastic Computing

Stochastic Computing (SC) [9], [10] is an unconventional method of computing that operates over uniform random bit-streams where data is represented by the ratio of 1's to the length of the bit-stream. This allows SC to represent real numbers between 0 and 1. For example, 11101010 is a representation for $5/8$ in the stochastic domain. Representing numbers with higher precision than that of the previous example simply requires increasing the length of the bit-stream. Implementing complex arithmetic circuits in the stochastic domain can be done with very simple hardware. Taking for example a multiplication circuit implementation, in the traditional binary domain, this circuit would be very complex and large, but in the stochastic domain, it can be implemented with just an AND gate as shown in Figure 2. SC-based implementations have achieved low-cost designs in many areas including but not limited to artificial neural networks and image processing [5], [11]–[13], [19]–[22]. The area- and power-efficiency of SC designs comes from reduction in the hardware complexity. To generate bit streams, stochastic number generators (SNGs) [23]–[25] are needed to convert binary data to bit streams with a uniform weight. However, stochastic computing has a major drawback. Stochastic circuits often have a long processing latency because long stochastic bit-streams must be processed for acceptable accuracy. The length of the bit-streams is exponentially proportional to the data precision. Long latency directly translates to high energy consumption. Even though SC-based implementations have a lower power consumption than that of their conventional binary counterparts, they take many more processing cycles such that they end up consuming more energy to complete the computation or task. Moreover, improving the data precision increases the latency and energy consumption exponentially in SC-based implementations. In summary, SC-based designs benefit from the low area and low power consumption due to the simplicity of arithmetic operations but face extremely large energy consumption, especially when the precision requirement is high.

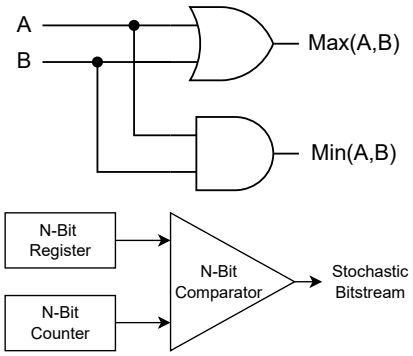

Fig. 3. (a) Stochastic-based Compare and Swap (CAS). (b) Binary to stochastic bit-stream converter.

B. Bitonic Sorting Networks

Bitonic sorting is suggested for hardware-based sorting due to its great scalability in term of number of comparators, $M * log_2(M)$, with M being the number of inputs, as well as how easy it is to parallelize in hardware. A Bitonic sorting network is constructed recursively, as indicated in Figure 1. An M-bit Bitonic sorting network is made up of two M/2-input Bitonic sorting networks and each of those are made up of two M/4-input networks and so on down to 2-input networks. Each non-recursive block of the network is made of two different sub-blocks referred to a merge block and a shift block. The merge block compares the top input with the bottom input then the 2^{nd} from the top input with the 2^{nd} from the bottom input and so on until all inputs are compared. The outputs of the merge block are connected to the inputs of two shift blocks. The shift block compares the top half of the block's inputs with the bottom half of the block's inputs. The outputs of each of the shift blocks are connected to two half sized shift blocks. This recurses down until the shift blocks are two-input wide. For both types of sub-blocks each input is only compared once. This allows each sub-block to be parallelized into a single step.

Traditionally, sorting networks [2] are large parallel circuits composed entirely of Compare And Swap (CAS) blocks. These blocks take in two numbers and return the minimum and maximum of the two inputs. A traditional binary CAS block consists of an N-bit comparator and two N-bit multiplexers, making the complexity of the block dependent on the precision of the input numbers. The number and arrangement of CAS blocks in a network is dependent on the sorting algorithm. Therefore, the complexity of the network scales with the number of inputs times the precision of the input numbers.

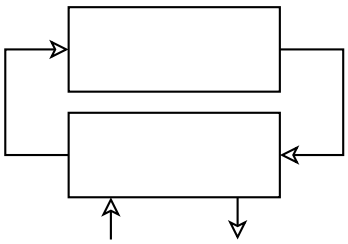

Fig. 4. A overall view of sorting network implementation for the proposed method.

To reduce the hardware cost, stochastic computing-based sorting network implementations were proposed [7], [8]. In a stochastic sorting network, the minimum and maximum functions of the CAS block can be implemented by an OR gate and an AND gate, respectively, as indicated in Figure 3(a). This makes the complexity of the block extremely low and independent of the data precision. However, SC design comes with an additional overhead cost of converting between binary and bit-stream formats. As shown in Figure 3(b), the binary to stochastic converter consists of an N-bit counter, an N-bit register and an N-bit comparator. The complexity of the conversion circuitry is proportional to the number of inputs and the precision of those inputs. For some SC designs, the overhead of the bit-stream generators take more than 80% of the total circuit design cost [26]. Another drawback of the stochastic designs is that the bit streams are often processed sequentially. So the number of cycles needed to complete the sorting task must also be taken into account. For the stochastic design, the number of cycles needed to sort N-bit precision data is 2^N. This limits the scalability of the SC design when it comes to high precision numbers. For example, to sort 32-bit numbers the SC-based sorting design takes 10^{32} cycles which is not acceptable in all applications.

III. Algorithm Design

This section describes the implementation of the proposed design. As discussed in the previous section, sorting two N-bit data with the stochastic approach takes 2^N clock cycles. This exponential increase in the processing time will significantly outweigh any benefit from the decreased hardware complexity for even a moderately high precision. The fundamental idea of this work is to exploit the simple implementation of the SC design while shortening the bit-stream by compressing it down to a weighted binary format. By doing so, we can achieve both a low hardware cost and a low latency for the sorting network design. The length of the weighted bit-streams scales linearly with the data precision making it feasible to process higher precision inputs. As indicated in Figure 4, there are two major parts in the proposed design. One is the bit-stream converter which generates weighted bit-streams (Figure 5). The other is the sorting design based on the proposed lock-and-swap (LAS) unit as shown in Figure 6.

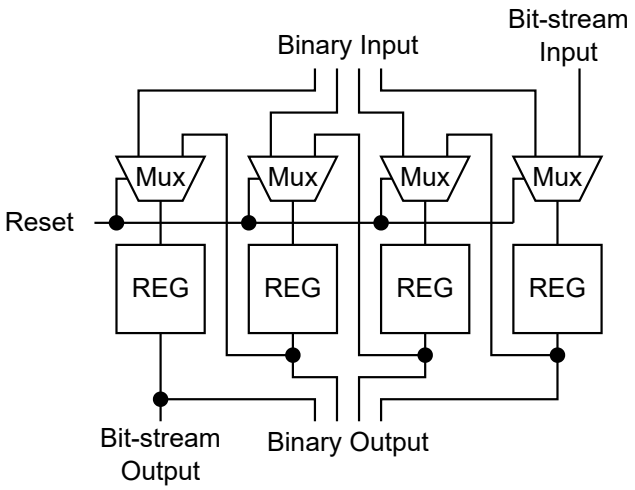

Fig. 5. An example of a 4-bit weighted bit-stream converter.

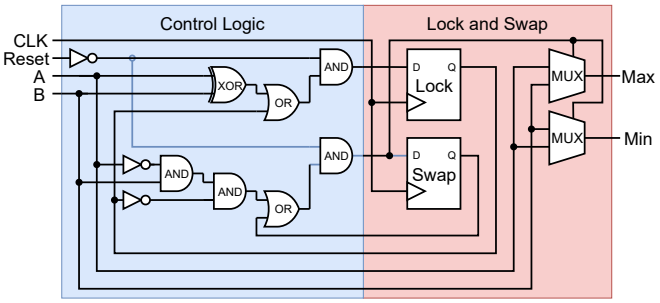

Fig. 6. Proposed Lock and Swap (LAS) block.

A. Weighted Bit-stream Converter

The weighted bit-stream converter is used to generate weighted bit-streams for the scalable sorting network implementation. Compared to the conventional bit-streams in SC, each bit in the weighted bit-streams remains their weight as the conventional binary value. By doing so, the number of bits in SC can be shrinked from 2^N to N for N-bit precision representation. Therefore, the conversion from exponential to linear can significantly reduce the energy consumption.

In the weighted bit-stream converter, both the binary input and the output bit-stream are weighted the same. This allows us to use much less complex hardware as indicated in Figure 4. To convert between binary and bit-stream formats, a specialized shift register is used. An example of a 4-bit weighted bit-stream converter is shown in Figure 5. Both the conversions from binary to weighted bit-streams and from weighted bit-streams to binary are integrated in this converter. By doing so, we eliminate the need for additional hardware to convert the bit-streams back into binary format by looping the output of the sorting network back into the same shift register. Reducing the complexity of the bit-stream converter is important as it is the largest unit of the design and the only component that its complexity scales with the data precision.

TABLE I
HARDWARE SYNTHESIS RESULT COMPARISON BETWEEN THE PROPOSED LAS AND PRIOR WORKS AT 100MHZ WITH 8-BIT PRECISION.

Number	Area (um^2)			Power (mW)			Energy (nJ)		
of Inputs	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic
8	2351	5353	4457	0.19	0.44	0.34	0.015	0.004	0.88
16	6527	15419	9094	0.50	1.23	0.69	0.040	0.012	1.76
32	16826	39291	18690	1.26	3.34	1.39	0.101	0.033	3.56
64	41416	104235	36545	3.17	8.84	2.83	0.25	0.088	7.26
128	104520	282744	73103	9.95	22.5	5.49	0.79	0.22	14.07

B. LAS-based Sorting Network Implementation

In the proposed design, the bits towards the front of the bit-stream will be more heavily weighted than the bits farther down the stream. The comparison between two bit-streams is decided when the first difference between two input streams is detected. All bits after this point are irrelevant to the comparison as the decision has already been made. This necessitates the use of a CAS block that can lock its comparison state once the comparison decision is made. This new CAS block will be referred to as a *Lock And Swap (LAS)* block. Figure 6 shows the proposed LAS block.

The LAS block has the same functionality as a CAS block. It gets two inputs and returns the minimum and maximum value between the two inputs. The block contains two registers: a *Lock* register and a *Swap* register. The *Lock* register remembers if the comparison decision is made earlier and prevents the LAS from changing the swap states. The *Swap* register, on the other hand, stores the swap state of the LAS when the comparison decision is made and the LAS is locked. The next state of each register is driven by its own set of combinational logic. The *Lock* register is set when the inputs to the LAS differ. The comparison is done at that point. *Lock* remains in this state until a reset signal is received. The *Swap* register is set when the inputs to the LAS block are in the swap condition and the comparison decision is not made yet. In the LAS block of Figure 6, the swap condition is when $A < B$. This is because we want the top output to have the maximum value and the bottom output the minimum value. If the outputs need to be flipped, either the swap condition or the inputs to the LAS' MUXs need to be changed. Since the circuit works with single-bit inputs, $A < B$ can be simply implemented by the Boolen function $\bar{A}\&B$ (i.e., one inverter plus one AND gate). The *Swap* register remains set until a reset signal is received. The MUXs at the output swap the input values when either the comparison is completed with a swap state or the comparison is not yet done but the inputs are at a swap condition. This is the same condition as the next state signal of the *Swap* register, so the same driver logic can be used.

In implementing the Bitonic sorting network, the LAS will be the basic component for sorting each two inputs. In summary, in our proposed LAS unit, the input A and B will be swap based on their weighted bits. The larger value will be routed to the top output and the smaller value will be routed to the bottom output. The unit accomplishes the sorting task with a low hardware cost and a short latency.

IV. EXPERIMENTAL RESULTS

In this section, we compare our proposed design with two baseline implementations: the SC-based design [7] and the traditional binary design [27]. All designs are structured as a Bitonic sorting network and have parallel binary inputs and outputs. We use the Synopsys design compiler with the FreePDK 45nm library [28] for hardware synthesis. We synthesize the designs based on three scanrios: a fixed clock frequency of 100 MHz, a fixed clock frequency of 1 GHz for the pipelined designs and, at each design's maximum working frequency. Unless stated otherwise, the number of inputs and the data precision are 8 and 8-bit, respectively.

A. Overall Comparison

We compare the hardware cost of the three implemented designs when varying the sorting networks size from 8 to 128. Table I shows the synthesis result for different sorting network designs. The input values to all three design approaches have 8-bit precision. From the results we can see that for 8-input sorting network the proposed LAS-based design can achieve up to 60% area reduction and 64% power reduction compared to the binary and stochastic designs. This trend continues for comparison between the proposed and the binary designs as the conventional binary design is complex and power hungry. However, this is not the case when comparing the proposed and the stochastic design. There is a cross over point between the sorting networks with 32 and 64 inputs where the stochastic design becomes more space and power efficient than the proposed design. This is due to the number of CAS / LAS blocks in the network scaling faster than the number of bit-stream converters. So any benefit gained from our design's simplified bit-stream converters would be surpassed by the downside of the more costly LAS blocks.

The stochastic CAS block is simple. Hence, the increase in the number of inputs has less impact on the power and area compared to the binary design. However, from the energy perspective, the stochastic-based implementation has a significantly higher energy consumption compared to our LAS design as in the stochastic approach the energy is proportional to the bit-stream length. As it can be seen, the proposed design consumes up to 30× less energy than the stochastic design. Compared to the binary implementation, even though the proposed design consumes slightly higher energy, the power and area costs are much lower. The proposed design approach addresses the massive area and power issue of the binary design as well as the excessive energy consumption of the

TABLE II
HARDWARE SYNTHESIS RESULT COMPARISON BETWEEN THE PROPOSED LAS AND PRIOR WORKS AT THEIR MAX FREQUENCY WITH 8-BIT PRECISION.

Number of Inputs	Area (um^2)			Power (mW)			Energy (nJ)			MAX Frequency (MHz)		
	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic
8	2429	5846	4548	2.5227	4.1666	3.2652	0.0190	0.0061	0.7887	1,060	680	1,060
16	6291	16203	9307	6.2933	11.7363	6.7006	0.0475	0.0311	1.6198	1,059	378	1,059
32	17037	44141	19197	16.3046	32.0107	13.4619	0.1360	0.1242	3.3484	959	258	1,029
64	45021	114000	41069	41.2285	84.0915	26.9366	0.4617	0.4593	8.3046	714	183	830
128	112494	297265	85550	102.4888	216.8701	53.6658	1.5077	1.5646	21.4196	544	139	641

TABLE III
PIPELINED HARDWARE SYNTHESIS RESULT COMPARISON BETWEEN THE PROPOSED LAS AND PRIOR WORKS AT 1 GHZ WITH 8-BIT PRECISION.

Number of Inputs	Area (um^2)			Power (mW)			Energy (nJ)		
	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic	Proposed	Binary	Stochastic
8	2819	9098	6239	2.9954	8.4840	4.5415	0.0240	0.0085	1.1626
16	8401	28002	15309	8.0021	26.0618	10.9000	0.0640	0.0261	2.7904
32	22345	77925	33864	20.4106	73.9101	26.2431	0.1633	0.0739	6.7182
64	55135	212679	78327	51.5176	194.5134	64.1867	0.4121	0.1945	16.4318
128	138761	560570	184277	128.7045	498.9233	151.9539	1.0296	0.4989	38.9002

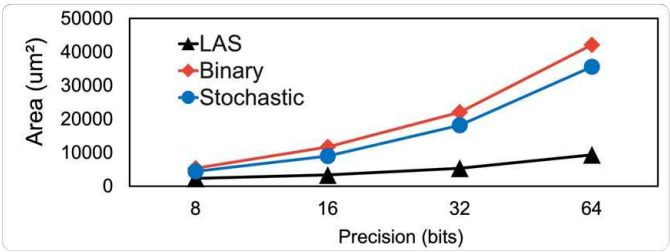

Fig. 7. Area results as scaling input precision.

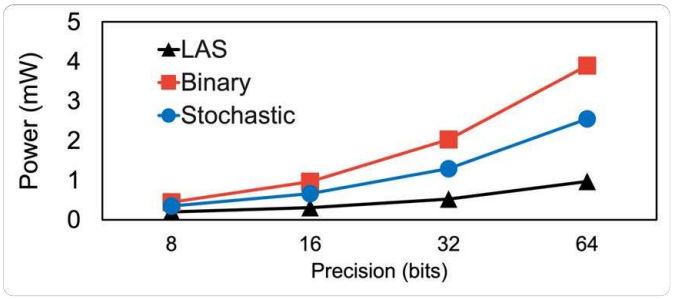

Fig. 8. Power results as scaling input precision.

stochastic design. This makes the proposed design optimal for embedded applications such as edge computing devices with renewable power resources like solar and wind, which have limited hardware resources but are supplied with continuous (though limited) power.

B. Scaling Data Precision

We also investigate the impact of changing data precision on the hardware cost of the sorting network in terms of area, power and energy consumption. The results are shown in Figures 7 and 8 and Table IV. The number of inputs is 8 in all three design approaches. As shown, when increasing the data precision, the proposed LAS scheme reduces the area by 28% - 77%, and the power by 26% - 75% compared to the other two baseline designs. The LAS is implemented with simple logic gates, achieving less hardware cost than the binary implementations. Also, compared to the stochastic implementation, LAS uses much cheaper bit-stream converters. In term of energy, both the LAS and the binary implementations consume significantly lower energy than the stochastic implementation as the stochastic design takes much more cycles to finish the sorting task. The LAS consumes slightly higher energy than the binary implementation as it needs N cycles to finish the sorting task where N is the number of bit precision of the data.

TABLE IV
ENERGY CONSUMPTION COMPARISON OF THE SORTING DESIGNS FOR 8-INPUT SORTING.

Precision (data-bits)	Energy (J)		
	LAS	Binary	Stochastic
8	1.58E-11	4.44E-12	8.82E-10
16	4.89E-11	9.61E-12	4.31E-07
32	1.68E-10	2.03E-11	5.54E-02
64	6.21E-10	3.90E-11	4.70E+08

C. Max Frequency Results

Table II reports the maximum working frequency of each sorting design and the corresponding hardware costs. From the reported results we can see that the proposed design can achieve up to 118.6% higher maximum frequency than the binary design and have comparable maximum frequency to the stochastic design. For the 8 and 16 input design, the proposed design has up to 60.7% and 25.7% lower area and power consumption, respectively, compared to the stochastic design. However, for the 32-input sorting the proposed design gets beat out by the stochastic design in terms of area and power but remains much more efficient than the binary design. More notably is the energy results. As can be seen in Table II, for 128-input sorting the proposed design achieves a 3.7% decrease in total energy consumption compared to the binary design. For larger sorting network sizes, the proposed design similarly

provides a significantly lower area and power consumption and also a lower energy consumption compared to the binary counterpart.

D. Pipelined Results

From the reported results in Table III, we can see that the proposed design can achieve 69.0% area and 64.7% power reduction when compared to the binary design. When compared to the stochastic design, the proposed design reports a reduction in area and power up to 54.8% and 34.0% respectively. These results remain comparatively similar with an increasing number of inputs.

V. Conclusion

In this work, we propose a novel scalable, low-cost sorting network design. We borrow the concept of stochastic computing but use weighted bit-streams to significantly reduce the number of processing cycles. A new lock and swap (LAS) unit is proposed to sort weighted bit-stream inputs. The weighted bit streams can significantly decrease the latency compared to the stochastic implementation. Also, the bit-stream operations use simple logic gates which reduce the hardware cost in terms of area and power compared to the binary implementation. Experimental results show that the proposed design approach achieves much better hardware scalability than prior work. Especially, as increasing the number of inputs, the proposed scheme can reduce the energy consumption by about 3.8% - 93% compared to prior binary and SC based designs.

VI. Acknowledgement

This work was partially supported by the following NSF awards 2208317, and 2204657. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Miklós Ajtai, János Komlós, and Endre Szemerédi. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9, 1983.

[2] Amin Farmahini-Farahani, Henry J Duwe III, Michael J Schulte, and Katherine Compton. Modular design of high-throughput, low-latency sorting units. *IEEE Transactions on Computers*, 62(7):1389–1402, 2012.

[3] Goetz Graefe. Implementing sorting in database systems. *ACM Computing Surveys (CSUR)*, 38(3):10–es, 2006.

[4] Naga Govindaraju, Jim Gray, Ritesh Kumar, and Dinesh Manocha. Gputerasort: high performance graphics co-processor sorting for large database management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 325–336, 2006.

[5] Peng Li, David J Lilja, Weikang Qian, Kia Bazargan, and Marc D Riedel. Computation on stochastic bit streams digital image processing case studies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(3):449–462, 2013.

[6] Kumara Ratnayake and Aishy Amer. An fpga architecture of stable-sorting on a large data volume: Application to video signals. In *2007 41st Annual Conference on Information Sciences and Systems*, pages 431–436. IEEE, 2007.

[7] M Hassan Najafi, David J Lilja, Marc D Riedel, and Kia Bazargan. Low-cost sorting network circuits using unary processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(8):1471–1480, 2018.

[8] M Hassan Najafi, David J Lilja, Marc Riedel, and Kia Bazargan. Power and area efficient sorting networks using unary processing. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 125–128. IEEE, 2017.

[9] Brian R Gaines. Stochastic computing systems. In *Advances in information systems science*, pages 37–172. Springer, 1969.

[10] Weikang Qian and Marc D Riedel. The synthesis of robust polynomial arithmetic with stochastic logic. In *2008 45th ACM/IEEE Design Automation Conference*, pages 648–653. IEEE, 2008.

[11] Bingzhe Li, M. H. Najafi, Bo Yuan, and David J. Lilja. Quantized neural networks with new stochastic multipliers. In *Proc. IEEE International Symposium on Quality Electronic Design*, pages 376–382, 2018.

[12] Ao Ren, Zhe Li, Caiwen Ding, Qinru Qiu, Yanzhi Wang, Ji Li, Xuehai Qian, and Bo Yuan. Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing. *ACM SIGPLAN Notices*, 52(4):405–418, 2017.

[13] Bingzhe Li, M. Hassan Najafi, and David J. Lilja. An FPGA implementation of a restricted boltzmann machine classifier using stochastic bit streams. In *Proc. IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pages 68–69. IEEE, 2015.

[14] M Hassan Najafi, S. Rasoul Faraji, Bingzhe Li, David J Lilja, and Kia Bazargan. Using resolution splitting to enhance performance of deterministic bit-stream computing. In *Proc. International Workshop on Logic and Synthesis*, 2018.

[15] Bingzhe Li, Yaobin Qin, Bo Yuan, and David J. Lilja. Neural network classifiers using a hardware-based approximate activation function with a hybrid stochastic multiplier. *Journal on Emerging Technologies in Computing Systems*, 15(1):12, 2019.

[16] Joonsang Yu, Kyounghoon Kim, Jongeun Lee, and Kiyoung Choi. Accurate and efficient stochastic computing hardware for convolutional neural networks. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 105–112. IEEE, 2017.

[17] Bingzhe Li, M.Hassan Najafi, and David J. Lilja. Low-cost stochastic hybrid multiplier for quantized neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 15(2):18, 2019.

[18] Ren Chen, Sruja Siriyal, and Viktor Prasanna. Energy and memory efficient mapping of bitonic sorting on fpga. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 240–249, 2015.

[19] Bingzhe Li, M Hassan Najafi, and David J Lilja. Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 36–41, 2016.

[20] Armin Alaghi, Cheng Li, and John P Hayes. Stochastic circuits for real-time image-processing applications. In *Proceedings of the 50th Annual Design Automation Conference*, pages 1–6, 2013.

[21] S Rasoul Faraji, M Hassan Najafi, Bingzhe Li, David J Lilja, and Kia Bazargan. Energy-efficient convolutional neural networks with deterministic bit-stream processing. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1757–1762. IEEE, 2019.

[22] Hyeonuk Sim and Jongeun Lee. A new stochastic computing multiplier with application to deep convolutional neural networks. In *Proceedings of the 54th Annual Design Automation Conference 2017*, pages 1–6, 2017.

[23] Florian Neugebauer, Ilia Polian, and John P Hayes. Building a better random number generator for stochastic computing. In *2017 Euromicro Conference on Digital System Design (DSD)*, pages 1–8. IEEE, 2017.

[24] Meng Yang, Bingzhe Li, David J. Lilja, Bo Yuan, and Weikang Qian. Towards theoretical cost limit of stochastic number generators for stochastic computing. In *Proc. IEEE Computer Society Annual Symposium on VLSI*, pages 154–159, 2018.

[25] Kyounghoon Kim, Jongeun Lee, and Kiyoung Choi. An energy-efficient random number generator for stochastic circuits. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 256–261. IEEE, 2016.

[26] Devon Jenson and Marc Riedel. A deterministic approach to stochastic computation. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2016.

[27] Chaitali Chakrabarti. Sorting network based architectures for median filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(11):723–727, 1993.

[28] James E Stine, Ivan Castellanos, Michael Wood, Jeff Henson, Fred Love, W Rhett Davis, Paul D Franzon, Michael Bucher, Sunil Basavarajaiah, Julie Oh, et al. Freepdk: An open-source variation-aware design kit. In *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*, pages 173–174. IEEE, 2007.