IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Unary Processing Based Odd-Even Merge Sorting Network

[1]Nandini.H.P, [2]Dr.Jalaja.S
[1]PG student, [2]Assistant Professor
[1]VLSI and Embedded Systems,
[1]Bangalore Institute of Technology, Bengaluru, INDIA

Abstract: Sorting is a core function in numerous applications. Sorting is performed in hardware with FPGA or ASIC for application which demands greater performance. The critical issues are consumption of power and area. This paper has an efficient method for sorting networks by utilizing odd-even merge sort by Batcher and "unary processing". In unary processing numbers are represented as a sequence of one value (like 0's) preceded by a sequence of other value (like 1's) in a stream, with the value described depending on number of 1's in the stream. The proposed approach is expected to show reduction in hardware.

***Index Terms* - CAR (compare and rearrange), SCE (stochastic computation element), Unary processing, sorting networks.**

I. INTRODUCTION

The job done by sorting has a massive significance in diverse applications. Cost of the hardware and consumed power are the two key challenges in hardware development of sorting. Out of all applications plenty of them will have constrained chip space and fabrication technologies will be scaling. Since leakage current grows exponentially with temperature, design should be in such a way that it consumes lowest possible power. So it is really essential to have hardware and power efficient methods for sorting.

The sensible remedy is to construct a network by using CAR blocks in a structure recognized as Batcher's [2] networks. Effortless implementation of Pipelining is possible in these networks. The hardware methods with parallel characteristics overtakes the software methods with serial characteristics. Cost of individual CAR unit combined with total CAR units determines the power consumption and hardware cost.

The paper aims to design a better sorting network by applying unary processing [1] to Batcher's odd-even merge sorting network. Because it has fewer CAR blocks when contrasted with bitonic sorting network [3]. But this approach has long latency because, unary representation of a number is exponentially longer than binary representation.

II. LITERATURE SURVEY

M. Hassan Najafi *et al.* [1] proposed a completely unique area efficient and power efficient technique for sorting network, which depends on "unary processing." In this approach the fundamental logic which is essential for sorting network will have simple gates and it does not depends on the width of the data. Over 90% reduction has been observed in power and area when compared with conventional approach but there is an increase in latency.

K.E. Batcher *et al*.[2] describes two popular hardware sorting networks Bitonic and odd-even merge sorting network and their application in switching networks and multi access memories.

Amin Farmahini-Farahani *et al.* [3] designed and implemented P-to-Q sorting and units for selecting Maximum set, having lower latency and increased throughput. Performance, configuration and resource necessities of these components are addressed by author.

Bradley D.Brown *et al.* [4] analyzes different SCEs, a few of which are presented for the first time, along with the overview of their working. Which involves squaring, division, addition, multiplication and subtraction. An effective method has defined for production and transformation among binary and stochastic signals.

Peng Li *et al.* [5] presents new SCEs based on finite state machines for digital image processing. As an example of practical applications of the method five digital image processing algorithms has been introduced and comparison of stochastic and conventional implementation has been made based on latency, hardware area, error tolerance.

III.SORTING NETWORKS AND UNARY PROCESSING

A. *Sorting networks*

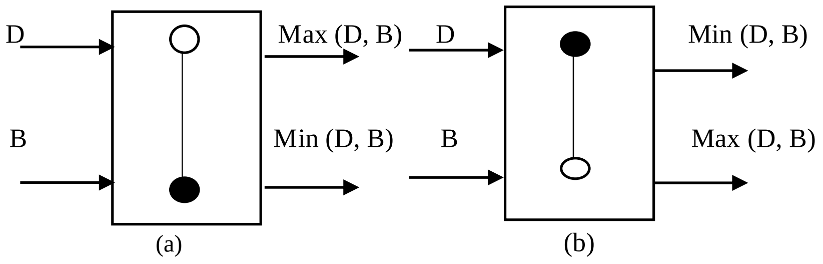


Fig. 1. Symbol of CAR (a) Ascending (b) Descending [1].

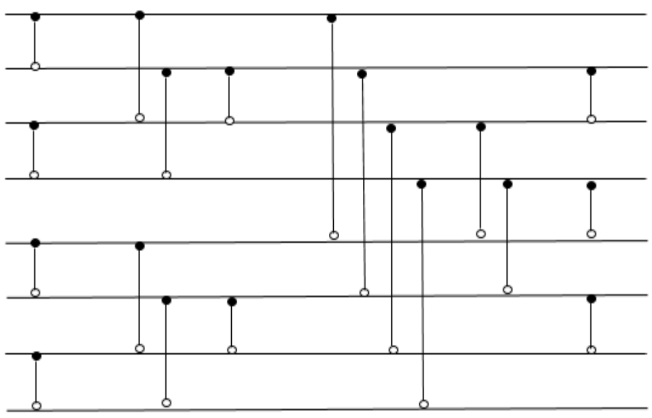


Fig. 2. 8-input odd-even merge sorting network.

Sorting network is a collection of CAR blocks in a particular order, which arranges the input data in a required sequence. Each compare-and-rearrange (CAR) block will have two input and two output. It makes the comparison of the applied input and upon necessity, it rearranges the values at output. There are two types of CAR blocks: 1) “ascending” 2) “descending” as shown in fig 1.

Two well-known sorting networks are bitonic and odd-even sorting network from Batcher [2]. In bitonic sorting repetitive merging of an ascending and descending batch of length J/2 gives a ordered batch of length J. A sorted set of length J is formed by repetitively merging two ascending set of length J/2 in odd–even merge sorting.

An odd-even merge sorting network is built from a set of odd-even merging modules. A J-input odd-even merging module (OEM-J) consists of $\log_2$ (J) CAR stages. Where the number of CAR blocks in each stage lies between J/4 to J/2.

B. *Processing in unary*

Weighted binary is the leading scheme for depicting the number in various fields. Weighted binary has a compact representation, but performing computations on this scheme is pretty complicated because a weight is assigned to every bit relative to the location of the bit. This depiction is not immune to noise, an inverted bit causes an error and the error may be huge if significant bit is inverted.

Unary processing is a progressive work on stochastic processing [1]. Equal weights are assigned to every digit in stochastic representation. The chances of getting 1 versus 0 in a sequence gives the number and it is bounded to the interval [0, 1]. In stochastic representation, for portraying real number having a resolution which is equal to 2^{-R}, a sequence with 2^{R} bits is needed [4]. Stochastic representation is less dense when compared to the representation in weighted binary. But it makes use of notably simple logic to compute the complex function, also it can tolerate bit flips.

The characteristics of unary processing is a mixture of characteristics of both stochastic processing and conventional binary. In unary processing the numbers are represented as sequence of one value (like 1’s) leading the sequence of other value (like 0’s) called as unary stream. Since all the bits are having equal weight unary stream exhibits immunity to noise. It is possible to portray just real numbers in stochastic processing. It is having two schemes: unipolar scheme for portraying the values within the range [0, 1] and bipolar scheme for portraying the values within the range [-1, 1]. But in case of unary processing both integers and real numbers can be portrayed. In unary, for real values the ratio of number of ones to the stream length gives the value. For portraying integer values, number of ones in the sequence defines the value directly [1]. Let’s take an instance, in the real domain while making use of unary, the sequences 1110 and 11111100 are both represents the value 0.75 but while representing integers it represents 3 and 6.

The Min (minimum of two numbers) and Max (maximum of two numbers) operations are two essential operations in sorting which have easy unary implementation. In weighted binary method multiplexers and a comparator which are data dependent will be used for implementing Min and Max operations. An AND gate can be used to implement Min operation and an OR gate can be used to implement Max operation in case of unary method, when two unary streams of equal-length are its inputs [1]. This is a valuable strength of unary method. An example is provided in Fig.3 to find the Max and Min values in unary processing.

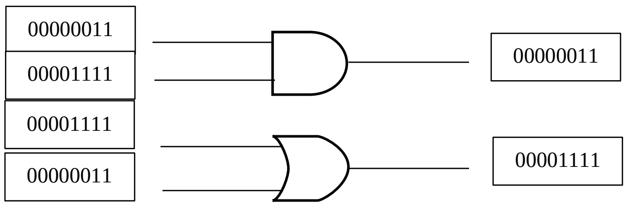


Fig. 3. Maximum and Minimum value operation [1].

C. *Design of sorting system*

The sorting networks are built from CAR units. As already discussed every CAR block is made up of an R-bit comparator and two R-bit multiplexer in conventional design where R is the length of data. Design complexity increases with the increase in length of the data and design complexity has a direct influence on power and area[1].

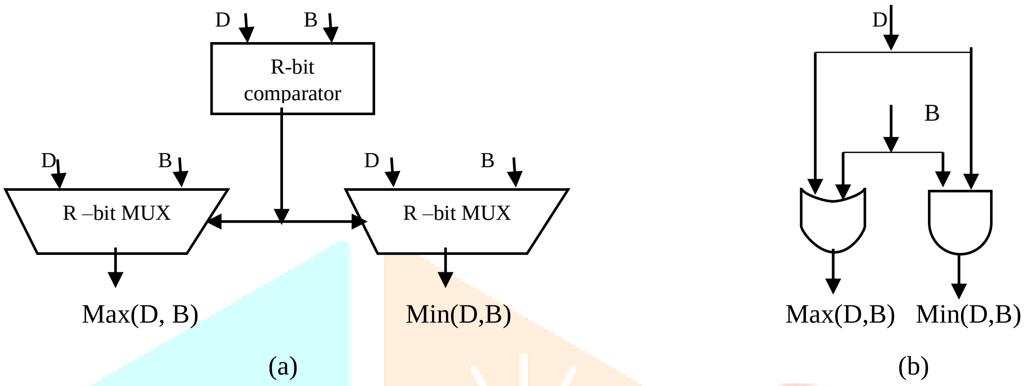


Fig. 4. CAR blocks hardware (a) Weighted binary (b) Unary approach [1].

As shown in the Fig.4 in unary domain instead of complex logic which depends on the length of the data, AND gate and OR gate are adequate for the construction of CAR unit. So circuit in unary method will consume less power and area when compared to the circuit in the binary method. Extra expenses will be experienced because of converting data from binary to unary and vice-versa. Also a lengthy processing time is required because the process is carried out on a stream having length 2^R [1].

IV.PROPOSED METHODOLOGY

Fig.5 shows the flow of complete sorting network which has three different stages

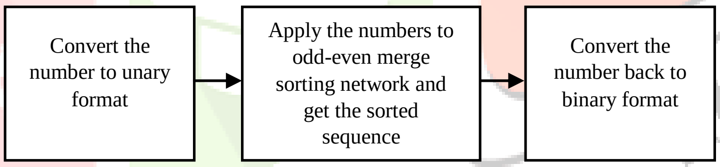

Convert the number to unary format
Apply the numbers to odd-even merge sorting network and get the sorted sequence
Convert the number back to binary format

Fig. 5. Flow of complete sorting network using unary processing.

Stage1: Binary to unary conversion

For the transformation of data to unary from binary unary stream generator is needed. Fig.6 shows a unary stream generator. Register consists of binary data whose unary form is required and it is needed to run the counter for number of clock cycles which is equal to 2^R. The value of the counter will be compared with the register value for every clock cycle if it is greater, then the comparator produces 0 else 1.

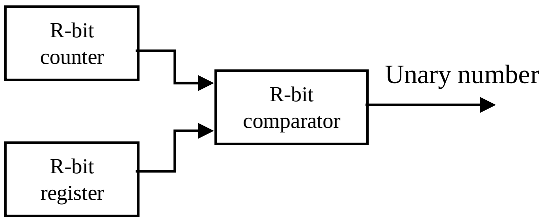


Fig. 6.Binary to Unary converter [1].

Stage2: Sorting of numbers

In comparison with the bitonic sort, odd-even merge sort makes use of less number of CAR blocks [3] as given by the TABLE I. So this paper makes use of odd-even merge sorting network to sort the numbers.

TABLE I

Number of CAR blocks required for 2^R - input odd-even merge and bitonic sorting units [3]

# of input	# of CAR blocks		
	Bitonic	Odd-even	Difference
8	24	19	5
16	80	63	17
32	240	191	49
64	672	543	129
128	1,792	1,471	321
256	4,608	3,839	769

Network for odd-even merge sort is as shown in Fig. 2.

Stage 3: unary to binary conversion

As shown in Fig. 7. A counter can be used as a unary to binary converter by counting the numbers of 1s in the given unary number.

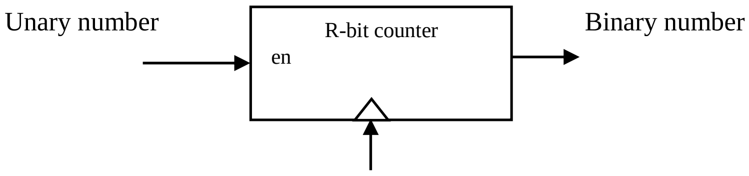
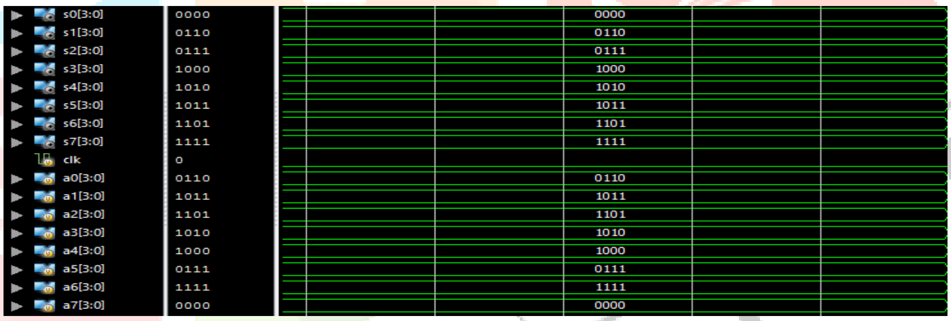


Fig. 7. Unary to Binary converter [9].

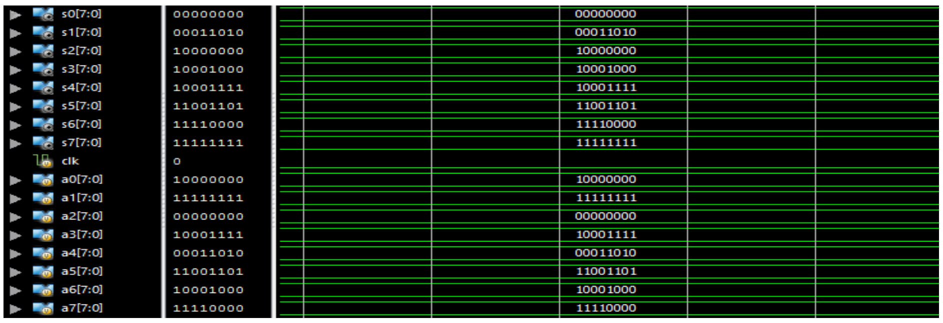
V.RESULTS AND DISCUSSION

For evaluating the advantages of proposed approach, a Verilog hardware description has been developed for odd-even merge and bitonic sorting network based on unary approach for 8 input with data widths of 4, 8, 16 and 32 bits. To get to know the hardware utilization of these designs Xilinx 3s250eft256 -4 has been used. Fig.8, Fig.9, Fig.10, Fig.11 show the simulation result of sorter for different data widths. TABLE II and TABLE III shows the results that are based on the consumption of hardware resources like number of slices, flip-flops and LUTS.



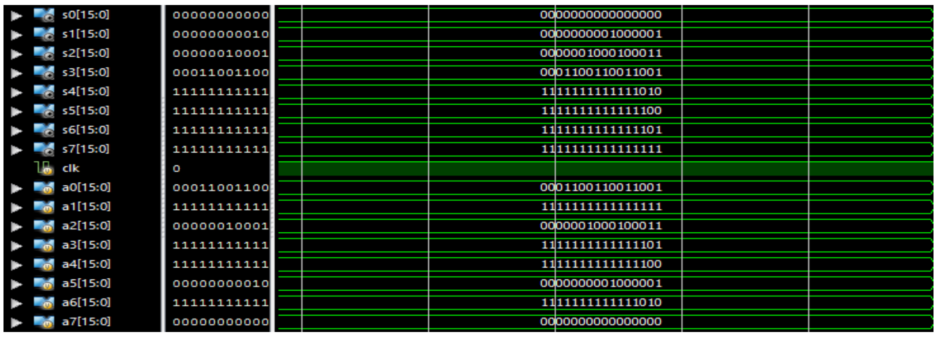
```
s0[3:0]   0000   0000
s1[3:0]   0110   0110
s2[3:0]   0111   0111
s3[3:0]   1000   1000
s4[3:0]   1010   1010
s5[3:0]   1011   1011
s6[3:0]   1101   1101
s7[3:0]   1111   1111
clk       0
a0[3:0]   0110   0110
a1[3:0]   1011   1011
a2[3:0]   1101   1101
a3[3:0]   1010   1010
a4[3:0]   1000   1000
a5[3:0]   0111   0111
a6[3:0]   1111   1111
a7[3:0]   0000   0000
```

Fig. 8. Simulation result of sorter for a data width of 4-bit.



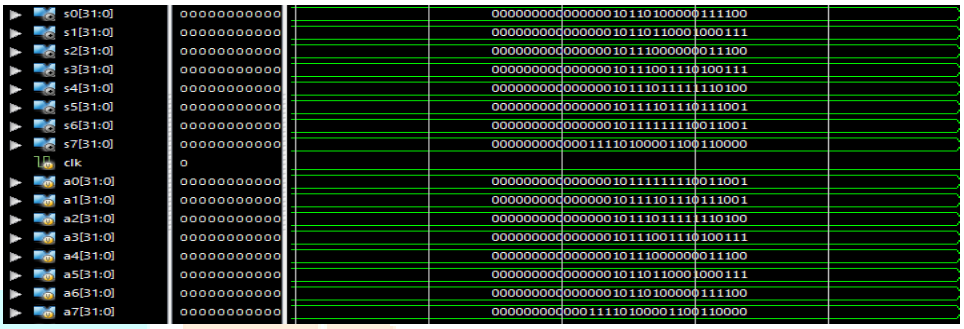
```
s0[7:0]   00000000   00000000
s1[7:0]   00011010   00011010
s2[7:0]   10000000   10000000
s3[7:0]   10001000   10001000
s4[7:0]   10001111   10001111
s5[7:0]   11001101   11001101
s6[7:0]   11110000   11110000
s7[7:0]   11111111   11111111
clk       0
a0[7:0]   10000000   10000000
a1[7:0]   11111111   11111111
a2[7:0]   00000000   00000000
a3[7:0]   10001111   10001111
a4[7:0]   00011010   00011010
a5[7:0]   11001101   11001101
a6[7:0]   10001000   10001000
a7[7:0]   11110000   11110000
```

Fig. 9. Simulation result of sorter for a data width of 8-bit.



```
s0[15:0]  00000000000  0000000000000000
s1[15:0]  00000000010  0000000001000001
s2[15:0]  00000010001  0000001000100011
s3[15:0]  00011001100  0001100110011001
s4[15:0]  11111111111  1111111111111010
s5[15:0]  11111111111  1111111111111100
s6[15:0]  11111111111  1111111111111101
s7[15:0]  11111111111  1111111111111111
clk       0
a0[15:0]  00011001100  0001100110011001
a1[15:0]  11111111111  1111111111111111
a2[15:0]  00000010001  0000001000100011
a3[15:0]  11111111111  1111111111111101
a4[15:0]  11111111111  1111111111111100
a5[15:0]  00000000010  0000000001000001
a6[15:0]  11111111111  1111111111111010
a7[15:0]  00000000000  0000000000000000
```

Fig. 10. Simulation result of sorter for a data width of 16-bit.



```
s0[31:0]  00000000000  00000000000000010110100000111100
s1[31:0]  00000000000  00000000000000010110110001000111
s2[31:0]  00000000000  00000000000000010111000000011100
s3[31:0]  00000000000  00000000000000010111001110100111
s4[31:0]  00000000000  00000000000000010111011111110100
s5[31:0]  00000000000  00000000000000010111101110111001
s6[31:0]  00000000000  00000000000000010111111110011001
s7[31:0]  00000000000  00000000000011110100001100110000
clk       0
a0[31:0]  00000000000  00000000000000010111111110011001
a1[31:0]  00000000000  00000000000000010111101110111001
a2[31:0]  00000000000  00000000000000010111011111110100
a3[31:0]  00000000000  00000000000000010111001110100111
a4[31:0]  00000000000  00000000000000010111000000011100
a5[31:0]  00000000000  00000000000000010110110001000111
a6[31:0]  00000000000  00000000000000010110100000111100
a7[31:0]  00000000000  00000000000011110100001100110000
```

Fig. 11. Simulation result of sorter for a data width of 32-bit.

TABLE II

Comparison of hardware utilization of odd-even merge sorting network using unary approach with the hardware utilization of odd-even merge sorting network using conventional approach in [7]

Data width	Odd even merge sorting [7]					Odd even merge sorting(unary approach)				
	# of IOBs	# of occupied slices	# of four-input LUTs	# of slice Flip flop	Delay (ns)	# of IOBs	# of occupied slices	# of four-input LUTs	# of slice Flip flop	Delay (ns)
4-bit	66	126	221	80	7.36	66	**42**	**75**	**36**	8.632
8-bit	128	302	549	160	9.89	130	**81**	**159**	**72**	**8.887**
16-bit	258	501	912	320	11.24	258	**149**	**295**	**144**	**9.636**
32-bit	514	996	1024	640	13.37	514	**285**	**567**	**288**	**10.80**

The results in TABLE II shows that there is a good amount of reduction in hardware utilization and delay in proposed odd-even merge sorting when compared to conventional odd-even merge sorting in [7].

TABLE III

Comparison of hardware utilization of bitonic sorting and odd-even merge sorting using unary approach

Data width	Bitonic sorting (unary approach)					Odd even merge sorting(unary approach)				
	# of IOBs	# of occupied slices	# of four-input LUTs	# of slice Flip flop	Delay (ns)	# of IOBs	# of occupied slices	# of four-input LUTs	# of slice Flip flop	Delay (ns)
4-bit	66	49	87	36	8.772	66	**42**	**75**	36	**8.632**
8-bit	130	89	177	72	9.049	130	**81**	**159**	72	**8.887**
16-bit	258	157	313	144	9.978	258	**149**	**295**	144	**9.636**
32-bit	514	293	585	288	10.970	514	**285**	**567**	288	**10.80**

The results in TABLE III shows that there is a considerable amount of reduction in hardware utilization and delay in odd-even merge sorting using unary approach when compared to bitonic merge sorting using unary approach.

VI. CONCLUSION

In variety of applications like switching networks and multi-access memory, Batcher's sorting network is used. They are more popular because of their regular structure. In the proposed method hardware is lowered because of less number of CAR blocks in odd-even sorting network when compared to bitonic sorting network and also because of unary processing. But this method has long latency because when compared to conventional representation serial bit representation is exponentially longer.

REFERENCES

[1] M. H. Najafi, D. J. Lilja, M. D. Riedel, and K. Bazargan, "Low-cost sorting network circuits using unary processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 8, pp. 1471–1480, Aug. 2018

[2] K. E. Batcher, "Sorting networks and their applications," in Proc. Apr. 30–May 2, 1968, Spring Joint Comput. Conf. (AFIPS), New York, NY, USA, 1968, pp. 307–314

[3] A. Farmahini-Farahani, H. J. Duwe, III, M. J. Schulte, and K. Compton, "Modular design of high-throughput, low-latency sorting units," IEEE Trans. Comput., vol. 62, no. 7, pp. 1389–1402, Jul. 2013.

[4] B. D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," IEEE Trans. Comput., vol. 50, no. 9, pp. 891–905, Sep. 2001.

[5] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," IEEErans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

[6] B. Gaines, "Stochastic computing systems," in Advances in Information Systems Science. New York, NY, USA: Springer-Verlag, 1969, pp. 37–172.

[7] V. S. Harshini, K. K. Senthil Kumar, "Design of Hybrid Sorting Unit", in IEEE 6th International Conference on smart structures and systems ICSSS, Chennai, India, 2019,pp. 1-6.

[8] D. Jenson and M. A. Riedel, "A deterministic approach to stochastic computation," in Proc. 35th Int. Conf. Comput.-Aided Design (ICCAD), New York, NY, USA, 2016, p. 102:1–102:8.

[9] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," ACM Trans. Embedded Comput. Syst., vol. 12, no. 2s, pp. 92:1–92:19, 2013.