Power and Area Efficient Sorting Networks using Unary Processing

M. Hassan Najafi, David J. Lilja, Marc Riedel and Kia Bazargan
Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA
{najaf011, lilja, mriedel, kia}@umn.edu

***Abstract*—Sorting is a common task in a wide range of applications from signal and image processing to switching systems. For applications that require high performance, sorting is often performed in hardware. Hardware cost and power consumption are the dominant concerns. The usual approach is to wire up a network of compare-and-swap units in a configuration called a Batcher (or Bitonic) network. This paper proposes a novel area- and power-efficient approach to sorting networks based on "unary processing." Data is encoded as serial bit-streams, with values represented by the fraction of 1's in a stream of 0's and 1's. (This is an evolution of prior work on stochastic logic. Unlike stochastic logic, the unary approach is deterministic and completely accurate.) Synthesis results of complete sorting networks show up to 87% area and power saving compared to the conventional binary implementations. However, the latency increases. To mitigate the increased latency, the paper uses a novel time-encoding of data. The approach is validated with implementation of an important application of sorting: median filtering. The result is a low-cost, energy-efficient implementation of median filtering with only a slight accuracy loss.**

***Keywords*-Sorting networks; unary processing; time-encoding data; stochastic computing; median filtering; low-cost design.**

I. Introduction

Sorting is an important task in applications ranging from data mining and databases to image and signal processing. For applications that require high performance, sorting is often performed in hardware. The total chip area is limited in many applications. As fabrication technologies continue to scale, keeping chip temperatures low is an important goal since leakage current increases exponentially with temperature. Power consumption must be kept as low as possible. Developing low-cost, power-efficient hardware-based solutions to sorting is an important goal. The usual approach for hardware implementation of sorting is to wire up a network of compare-and-swap (CAS) units in a configuration called a Batcher (or Bitonic) network. The hardware cost and the power consumption depend on the number of CAS blocks and the cost of each CAS block.

This paper proposes a novel area- and power efficient approach to sorting networks based on "unary processing." Data is encoded as serial bit-streams, with values represented by the fraction of 1's in a stream of 0's and 1's. This is an evolution of prior work on stochastic processing [15][7][16]. Our designs inherit the fault tolerance and low-cost design advantages of stochastic processing while producing completely accurate results. As with stochastic processing, however, the approach is handicapped in terms latency. A serial representation is exponentially longer than a binary positional representation. To mitigate the long latency issue of unary processing, this paper adopts a time-encoding approach recently proposed in [11]. The proposed approach is validated with implementation of an important application of sorting networks: median filtering. Synthesis results show up to 87% area and power savings compared to conventional weighted binary implementations. Time-encoding the data provides a significant improvement in the latency and energy consumption with only a slight loss in accuracy.

*This work was supported in part by National Science Foundation grant no. CCF-1408123. 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.

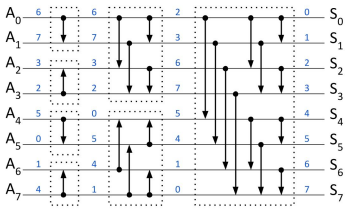

Figure 1: The CAS network for an 8-input bitonic sorting [6].

II. Background

A. Sorting Networks

A sorting network is a combination of CAS blocks that sorts a set of input data. Each CAS block compares two input values and swaps the values at the output, if required. The bitonic sorting network proposed by Batcher [4] is a popular configuration for a sorting network that has the lowest known latency for hardware-based sorting. Bitonic Sort uses a key procedure called Bitonic Merge (BM). Given two equal size sets of input data, sorted in opposing directions, the BM procedure will create a combined set of sorted data. It recursively merges an ascending and a descending set of size N/2 to make a sorted set of size N [8]. Figure 1 shows the CAS network for an 8-input bitonic sorting network consisting of 24 CAS blocks. 16-input, 32-input, and 256-input bitonic sorting networks can be similarly constructed using 80, 240, and 4,608 CAS blocks, respectively.

B. Unary processing

Weighted binary radix has been the dominant format for representing numbers in the field of computer engineering since its inception. The representation is compact; however, computing on this representation is relatively complex, since each bit must be weighted according to its position. Also, the representation is very susceptible to noise: a flipped bit can introduce a large error (if it is a significant bit in the

representation). Poppelbaum [15] and Gaines [7] introduced stochastic processing based on uniformly distributed random bit-streams. All digits have the same weight in this computing paradigm. Numbers are limited to the [0, 1] interval and encoded by the probability of obtaining a one versus a zero in the stream. To represent a real number with a resolution of 2^{-M}, a stream of 2^M bits is required. Beginning in 2008, Qian et al. reintroduced the concept of stochastic processing to the computer engineering community [16] [19]. Clearly, a stochastic representation is much less compact than weighted binary; this translates to high latency. However, complex functions can be computed with remarkably simple logic, e.g. multiplication can be performed using a single AND gate. Also, the representation can tolerate high clock skew [10], timing errors [1], and soft logic errors (i.e., bit flips) [19][3].

A recently evolution of the idea of stochastic computing (SC) has been to perform the processing deterministically [9][11][12]. If properly structured, computation on deterministic bit-streams can be performed with the same circuits as are used in SC. The results are completely accurate with no random variations; furthermore, the latency is greatly reduced. The idea of unary processing was first introduced in the 1980s [14] as a hybrid information processing technique that has characteristics common to both conventional binary and to SC. It is deterministic, but borrows the concept of averaging from stochastic methods. In this paper we apply unary processing to problem of desiging low-cost, power-efficient sorting networks.

Unary streams. In unary processing, numbers are encoded uniformly by a sequence of one value (say, 1) followed by a sequence of the other value (say, 0) (See Figure 2). This uniform sequence of bits is called a unary stream. As with stochastic streams, all the bits have equal weight. This property provides the immunity to noise. Multiple bit flips in a long unary stream produce small and uniform deviations from the nominal value.

Unary Operations. The maximum (Max) and minimum (Min) value functions are two useful functions with simple and low cost unary implementation: an AND (OR) gate gives the Min (Max) of two unary streams when two equal-length unary streams are connected to its inputs. These gates showed a similar functionality when fed with correlated stochastic bit-streams [2]. Figure 3 shows an example of finding the Max and Min values in unary processing. Recent work has shown absolute-value subtraction (using an XOR gate) [2], comparison (using a D-type flip-flop) [12], and multiplication (using an AND gate) [9], [11] of unary streams.

Time-based unary streams. The representation of numbers in unary processing is not limited to purely digital bit-streams. A time-based interpretation of numbers is also possible using pulse modulation of data [11]. Figure 2 shows both approaches. While both approaches can operate on the same unary logic, the time-based representation offers a seamless solution to the increasing number of time-based sensors and, as we will show, can be exploited in addressing the long latency problem of unary circuits.

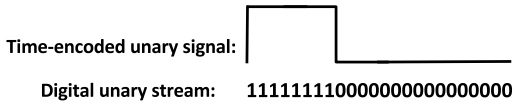

Figure 2: Time-based vs. digital-stream unary representation.

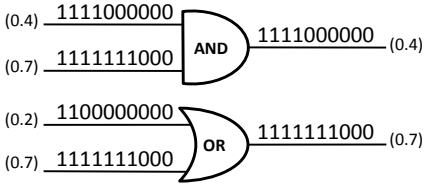

Figure 3: Example of performing maximum and minimum operations on unary streams.

III. Complete Sort System

In this section we first discuss the conventional binary and the unary design of complete sorting networks and then compare the designs by presenting synthesis results.

A. Conventional Design vs Unary Design

Sorting networks are made of CAS blocks. The hardware cost of a sorting network is therefore a direct function of the number of CAS blocks and the cost of each block. As shown in Figure 4a, in a weighted binary design with a data-width of M bits, each CAS block consists of one M-bit comparator and two M-bit multiplexers. In the unary domain, however, one AND and one OR gate is sufficient to synthesize a CAS block. The sorting networks can therefore be synthesized regardless of the resolution of the input data. While the synthesized circuit will be much less costly than the circuit synthesized in the binary approach, additional overhead must be incurred for conversion units which are required to convert the data between the binary and the unary formats, and a longer time is required to perform the operation on 2^M-bit long streams.

Assuming that the input data is given in binary format and the result must again be in binary, a unary stream generator is required to convert the data from binary to unary and a counter is required to count the number of ones in the final unary stream to convert the result back into binary. Figure 5 shows the design of a unary stream generator. Note that while the converters are data-width dependent, the CAS blocks synthesized with the unary approach are independent of data resolution.

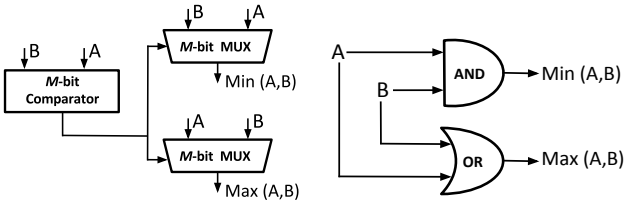

Figure 4: Hardware implementation of a CAS block a) Conventional binary design b) Unary design.

Table I: Synthesis results for complete bitonic sort networks

# of inputs and outputs	# of CAS units	Area (μm^2)		Critical Path (ns)		Power (@max f) — (@50MHz) (mW)			
		Conven.	Unary	Conven.	Unary	Conven.	Unary	Conven.	Unary
8	24	3,086	2,194	1.85	0.74	1.30	3.26	0.12	0.13
16	80	10,534	4,511	2.73	0.87	3.66	5.30	0.49	0.25
32	240	32,508	9,235	4.06	1.07	8.86	8.40	1.75	0.49
64	672	90,691	19,028	5.71	1.33	19.8	13.4	5.48	0.96
128	1,792	242,049	33,916	7.49	1.62	44.4	21.4	15.7	1.80
256	4,608	586,456	74,719	9.71	1.91	88.75	36.5	42.2	3.64

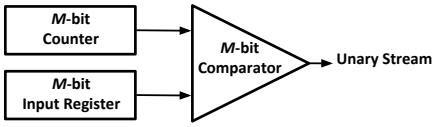

Figure 5: Unary stream generator.

B. Design Evaluation

We developed Verilog hardware descriptions of complete bitonic sorting networks for 8, 16, 32, 64, 128, and 256 data inputs, for both the conventional binary and for the proposed unary approach. For the unary approach, the architectures include the required conversion units from/to binary. The designs are synthesized using the Synopsys Design Compiler vH2013.12 and a 45nm standard-cell library. We report synthesis results for a data-width of 8 bits. In order to find the minimum hardware cost, we implement a non-pipelined version of each architecture.

Table I shows the synthesis results. For small networks like the 8-input sort networks, the cost overhead of converters was comparable to the saving due to using a low-cost CAS implementation and so lower savings are achieved. By increasing the number of inputs, and so the number of CAS blocks, the savings dominate the overheads and a hardware area saving of around 87% is achieved when implementing the 256-input sorting network with the unary approach.

The total power consumption at the maximum feasible working frequency of each architecture, and also at 50 Mhz, are presented in Table I. Although the unary designs would have a much lower power consumption at low speeds, due to a lower critical path latency and so higher maximum working frequency, the power numbers reported for the unary implementations of the 8 and 16 input sorting networks are greater than the power numbers reported for their corresponding binary implementations. For larger sorting networks (32-input and above), the simplicity of the unary design has led to even a lower power consumption at the maximum working frequency than that of the binary implementation.

Due to a simpler architecture, the critical path (CP) latency of the designs synthesized with the unary approach is lower than that of the conventional binary designs. However, the total latency of the unary approach, which is the product of the CP latency and the number of clock cycles that the system processes the unary stream, is much more than the latency of the conventional design. Although the longer latency of the unary approach is still acceptable for many applications, a more important issue is the energy consumption. Energy consumption is evaluated by the product of the processing time and the total power consumption. A very long processing time of unary design would lead to a higher energy consumption than their binary counterparts. We will address the long latency and high energy consumption problem of unary designs in the next section.

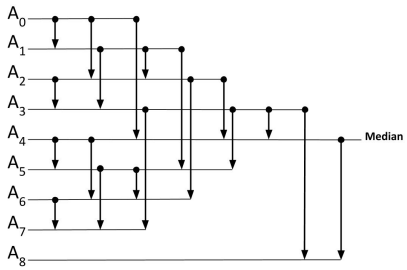

Figure 6: The CAS network for a 3x3 Median Filter [13].

IV. Highly Efficient Median Filters

A median filter is a popular non-linear filter widely used in image and signal processing applications. It replaces each input data with the median of all the data in a local neighborhood. The high computational complexity of median filters makes their hardware implementation expensive and inefficient for many applications. In this section we first propose a low-cost implementation of median filters similar to the unary sorting networks introduced in Section III. We then exploit a time-based representation of input data to address the long latency problem of the unary circuits.

A. Circuit Design

Sorting network-based architectures [5] consisting of a network of CAS blocks are one of the most common approaches for hardware implementation of median filters. Figures 6 shows the sorting network for a 3x3 median filter. We developed a non-pipelined structure of this median filter with both the conventional binary and the proposed unary design approach with 8-bit input data resolution. The CAS blocks of Figure 4 were used in these architectures.

Table II shows the synthesis results for these architectures. The overhead in the bit-stream based unary design includes the required converters from/to binary. As can be seen, the unary implementation significantly improves the hardware cost. For applications in which hardware cost and power consumption are the main priorities, the proposed unary design outperforms the conventional binary design. However, for high-performance, low-energy applications the binary design can be a better choice. In the following section we exploit the concept of time-based representation of data to improve the latency and energy consumption of the unary-based median filtering at the cost of a slight accuracy loss.

Table II: Synthesis results of the sorting network-based median filters for data-width=8.

Median Filter	Design Approach	Area (μm^2)			Latency (ns)		Power (mW) (@max freq)	Energy (pJ)
		CAS Logic	Overheads	Total	CP	Total		
3x3	Conventional Binary	2,167	-	2,167	2.10	2.10	1.03	2.1
	Unary-Bit-Stream-based	79	917	**996**	0.70	179.2	0.95	170.2
	Unary-Time-based	79	776	**855**	0.39	**0.39**	1.78	**0.69**

B. Time-based unary design

1) Overview: Image sensors convert the light intensity to an analog voltage or current. With more and more sensors providing time-encoded outputs and ways to convert voltage or current to time signals [17], the sensed data in the form of time-encoded signals can directly be fed to unary circuits. Based on the idea of time-encoding data introduced in [11], we time-encode the input data to address the long latency of processing using unary circuits. An analog-to-time converter (ATC) (e.g., a PWM signal generator) is used to convert the sensed data to a time-encoded pulse signal. The converted signal is processed using the unary circuit and the output is converted back to a desired analog format using a time-to-analog converter (TAC) (e.g., a voltage integrator).

2) Evaluation: Table II shows the area, latency, power, and energy consumption of the implemented median filtering circuits synthesized with the conventional binary, digital bit-stream based unary, and the proposed time-based unary approach. The low-cost pulse-width modulator proposed in [11] was used as the ATC and a Gm-C active integrator [18] was used as the TAC in the time-based unary design. While a pulse-width modulator generates a periodic signal with a specific duty cycle and frequency, only one period of the generated signal is needed for processing the data. The duty cycle is determined by the DC level of the sensed data. We extracted the area and energy numbers from [11] and report them as the overhead of the time-based unary design. A separate ATC is used for time-encoding each input data. The reported overhead numbers are for a working frequency equal to the inverse of the CP latency of the circuit. Assuming that the clock signal that drives the ATC is available in the system, a lower working frequency translates to a lower area and energy overhead. As can be seen in Table II, the total area, latency, and energy consumption of the time-based unary design are better than those of the bit-stream based unary and also those of the binary design.

The down-side of the time-based unary design, however, is a slight accuracy loss. The working frequency of the ATC affects the effective number of bits in representing data, hence the accuracy of computation. To evaluate the performance of the time-based design, we developed a SPICE netlist of the circuit and simulated its operation on a 128×128 noisy soldier image [12]. Simulations were carried out using a 45-nm standard cell library in HSPICE. Table III shows the mean absolute error rates (MAE) for the images produced using the time-based unary design. Image pixel intensities were converted to pulse signals using the ATC of [11] and also using the HSPICE built-in pulse generator (an ideal ATC). As can be seen in Table III, a lower working frequency leads to a higher accuracy in the time-based approach. A MAE of less than 1 percent is achieved with 1ns processing time. The inherent inaccuracy in converting the values with the ATC of [11] resulted in a slightly higher error rate when compared to the error rate where using the ideal ATC.

Table III: MAE of the time-based unary circuit.

Median Filter Time-based Unary		Length of input signals (1/freq.)			
		CP	1ns	2ns	5ns
3x3	Ideal ATC	2.09%	0.84%	0.45%	0.19%
	ATC of [11]	2.65%	1.05%	0.56%	0.21%

V. Conclusion

This work proposed an area and power efficient implementation of sorting networks based on unary processing. The core processing logic consists of simple gates. The only overhead in the approach, the cost of converting data from/to binary, is small. More than 80% area and power savings are observed when compared to a conventional binary implementation. The penalty is latency. Processing digital unary streams requires a relatively long running time (e.g., more than 100ns). To mitigate the latency, we further developed a time-based unary design approach in which the input data is encoded in time and represented with pulse signals. The result is a significant improvement in the latency and energy consumption, at the cost of a slight loss in accuracy.

References

[1] A. Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng, and J. Li. Trading accuracy for energy in stochastic circuit design. *ACM JETC*, 2017.

[2] A. Alaghi and J. Hayes. Exploiting correlation in stochastic circuit design. In *Proc. ICCD*, pages 39–46, Oct. 2013.

[3] B. Li et al. Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier. In *Proc. FPGA*, pages 36–41, New York, NY, USA, 2016.

[4] K. E. Batcher. Sorting networks and their applications. In *AFIPS '68*, pages 307–314, New York, NY, USA, 1968. ACM.

[5] C. Chakrabarti. Sorting network based architectures for median filters. *IEEE Trans. on Circuits Syst. II: Analog Digit. Signal Process.*, 1993.

[6] A. Farmahini-Farahani et al. Modular design of high-throughput, low-latency sorting units. *IEEE Transactions on Computers*, July 2013.

[7] B. Gaines. Stochastic computing systems. In *Advances in Information Systems Science*. Springer US, 1969.

[8] B. Gedik, R. R. Bordawekar, and P. S. Yu. Cellsort: High performance sorting on the cell processor. In *VLDB'07*, 2007.

[9] D. Jenson and M. Riedel. A Deterministic Approach to Stochastic Computation. In *Proc. ICCAD*, New York, NY, USA, 2016.

[10] M. H. Najafi et al. Polysynchronous Clocking: Exploiting the Skew Tolerance of Stochastic Circuits. *IEEE Trans. on Computers*, 2017.

[11] M. H. Najafi et al. Time-Encoded Values for Highly Efficient Stochastic Circuits. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2017.

[12] M. H. Najafi and D. J. Lilja. High-Speed Stochastic Circuits Using Synchronous Analog Pulses. In *Proc. ASP-DAC*, pages 481–487, 2017.

[13] P. Li et al. Computation on Stochastic Bit Streams Digital Image Processing Case Studies. *IEEE TVLSI*, pages 449–462, 2014.

[14] W. Poppelbaum, A. Dollas, J. Glickman, and C. O'Toole. Unary processing. In *Advances in Computers*. Elsevier, 1987.

[15] W. J. Poppelbaum, C. Afuso, and J. W. Esch. Stochastic computing elements and systems. In *AFIPS '67*, New York, NY, USA, 1967. ACM.

[16] W. Qian and M. Riedel. The synthesis of robust polynomial arithmetic with stochastic logic. In *Proc. DAC*, pages 648–653, 2008.

[17] V. Ravinuthula, V. Garg, J. G. Harris, and J. A. B. Fortes. Time-mode circuits for analog computation. *IJCTA*, 37(5):631–659, 2009.

[18] W. Sansen. Analog design essentials, ser. the international series in engineering and computer science, 2006.

[19] W. Qian et al. An Architecture for Fault-Tolerant Computation with Stochastic Logic. *IEEE Transactions on Computers*, 2011.