LOW COST SORTING NETWORKS USING UNARY PROCESSING

[1]G. Pavani, [2]K. Anusha, [3]G. Pranay kumar, [4]A.Aravind, [5]E.Sanjeeva, [6]B.Santhoshi
[1,2,3,4,5]UG Student, [6]Assistant Professor, Dept. of Electronics and Communication Engineering, Visvesvaraya College of Engineering and Technology, Mangalpalle, Telangana, India

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 with application specific integrated circuits or field-programmable gate arrays. 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 the Batcher (or bitonic) network. Such networks can readily be pipelined. This paper proposes a novel area-efficient and power-efficient approach to sorting networks, based on "unary processing." 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) in a stream of 0's and 1's with the value defined by the fraction of 1's in the stream. Synthesis results of complete sorting networks show up to 92% area and power saving compared to the conventional binary implementations. However, the latency increases. To mitigate the increased latency, this paper uses a novel time-encoding of data. The approach is validated with two implementations 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, compared to conventional implementations.

INTRODUCTION:
This paper proposes a novel area-efficient and power efficient approach to sorting networks based on "unary processing." " 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) in a stream of 0's and 1's with the value defined by the fraction of 1's in the stream. Synthesis results of complete sorting networks show up to 92% area and power saving compared to the conventional binary implementations. However, the latency increases. To mitigate the increased latency, this paper uses a novel time-encoding of data. The approach is validated with two implementations 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, compared to conventional implementations.

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. . However, complex functions can be computed with remarkably simple logic, e.g., multiplication can be performed using a single AND gate. The maximum (Max) and minimum (Min) value functions are two useful functions with simple and low-cost unary implementation. In a weighted binary design, data-width-dependent comparator and multiplexer units must be used to implement these functions. In unary processing, individual gates can synthesize these functions: an AND gate gives the minimum of two unary streams when two equal-length unary streams are connected to its inputs; an OR gate gives the maximum value when its inputs are fed with two equal-length unary streams.

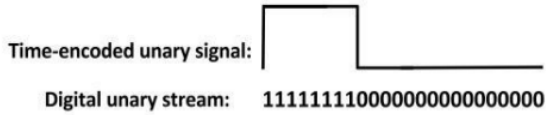

Fig 1:Time-based versus digital-stream unary representation.

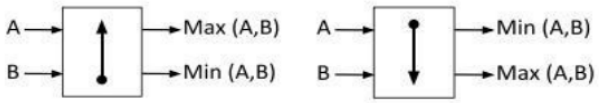

Fig 2: Schematic symbols of CAS block. i)Ascending ii)Descending

PROPOSED ARCHITECTURE

- This paper proposes a novel area-efficient and power efficient approach to sorting networks based on unary processing. This is an evaluation of prior work on stochastic processing.
- Our designs inherit the fault tolerance and low cost design advantages of stochastic processing while producing completely accurate and deterministic results.
- As with stochastic processing, however, the approach is handicapped in term of latency. A serial representation is exponentially longer than conventional binary positional representations.

Block Diagram:

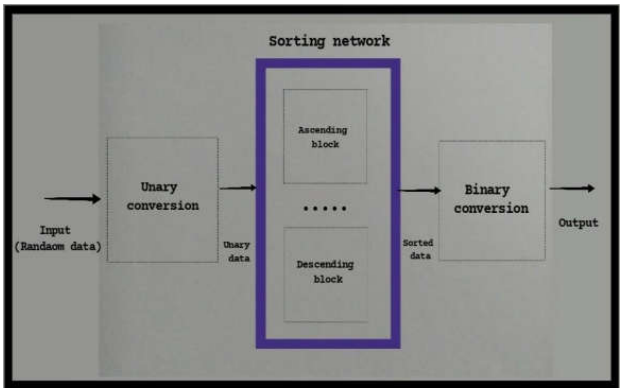

Fig 3: Block diagram of sorting network using unary processing

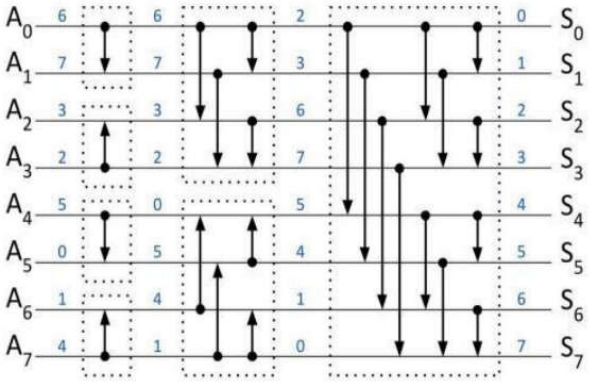

Fig 4: The CAS network for an 8- input bitonic sorting

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. There are two variants: 1) an “ascending” type and 2) a “descending” type. Fig. 1 shows their schematic symbols. In a conventional design, each CAS block consists of an M-bit comparator and two M-bit multiplexers, where M is the data width of the inputs.

Sorting networks are fundamentally different from software algorithms for sorting such as quick sort, merge sort, bubble sort, etc. since the order of comparisons is fixed in advance; the order is not data dependent as is the case with software algorithms. The bitonic and odd–even merge sorting networks proposed by Batcher are the two popular configurations 0f sorting networks . They have 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 . Fig. 2 shows the CAS network for an 8-input bitonic sorting network made up of ascending and descending BM units. The total number of CAS blocks in an N-input bitonic sorting is $N \times \log2(N) \times (\log2(N) + 1)/4$. Thus, 8-input, 16-input, 32-input, and 256-input bitonic sorting networks require 24, 80, 240, and 4608 CAS blocks, respectively.

An odd–even merge sorting network recursively merges two ascending sequences of length N/2 to make a sorted sequence of length N. Odd–even merge sorting units require fewer CAS blocks than bitonic sorting units, but often have more complex wiring . Due to their simpler structure, in this paper, we will present designs based on bitonic sort networks. The proposed design approach, however, is applicable to any sorting network topology, including odd–even sorting networks; it will accrue the same advantages.

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 et al. and Gaines 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 2M bits is required. reintroduced the concept of stochastic processing to the computer engineering community.

Clearly, a stochastic representation is much less compact than conventional 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.

If properly structured, computation on deterministic bit streams can be performed with same circuits as are used in stochastic computing. The results are completely accurate with no random variations; furthermore, the latency is greatly reduced. The idea of unary (or burst) processing was first introduced in 1980s [37], [38] as a hybrid information processing technique that has characteristics common to both conventional binary and to stochastic processing. It is deter-ministic, 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.

1) 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 Fig. 3). This uniform sequence of bits is called a unary stream. In the literature, this method of encoding is also called pulsewidth encoding [15]. As with stochastic streams, all the bits have equal weight.

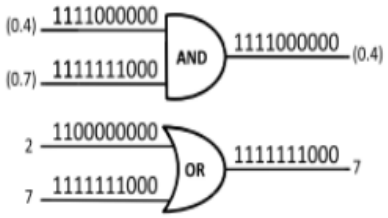

Fig 5: Example of performing the maximum and minimum operations on unary streams

This property provides the immunity to noise. Multiple bit flips in a long unary stream produce small and uniform deviations from the nominal value. In stochastic processing, only real-valued numbers can be represented: numbers in the [0, 1] interval with the unipolar format and numbers in the [−1, 1] interval with the bipolar format. In contrast, with unary streams both real-valued and integer numbers can be represented. In representing real-value numbers, the number of ones divided by the length of stream determines the value.

In representing integer values, the number of ones directly determines the value. For example, when using unary streams in the real domain, the streams 1000 and 11000000 are both representations of the value 0.25. In the integer domain, on the other hand, these streams represent one and two, respectively. Similar to the bipolar format for stochastic streams, negative numbers can also be represented with unary streams using a simple linear transformation .

2) Unary Operations:

The maximum (Max) and minimum (Min) value functions are two useful functions with simple and low-cost unary implementation. In a weighted binary design, data-width-dependent comparator and multiplexer units must be used to implement these functions. In unary processing, individual gates can synthesize these functions: an AND gate gives the minimum of two unary streams when two equal-length unary streams are connected to its inputs; an OR gate gives the maximum value when its inputs are fed with two equal-length unary streams. These gates showed a similar functionality when fed with correlated stochastic bit streams.An important advantage of unary processing is that synthesizing a function is independent of the resolution of data (length of streams). The same core logic is used for processing 128-bit unary streams that is used for processing 256-bit unary streams. While developing a general method for synthesizing all operations with unary processing is still a work in progress, recent work has shown absolute value subtraction (using an XOR gate), comparison (using a D-type flip-flop) [31], and multiplication (using an AND gate) [23], [30] of unary streams.

3) 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.

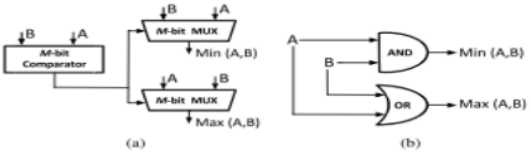

Fig 6: shows both approaches.

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.

SIMULATION & SYNTHESIS RESULTS

RTL SCHEMATIC (SYNTHESIS)

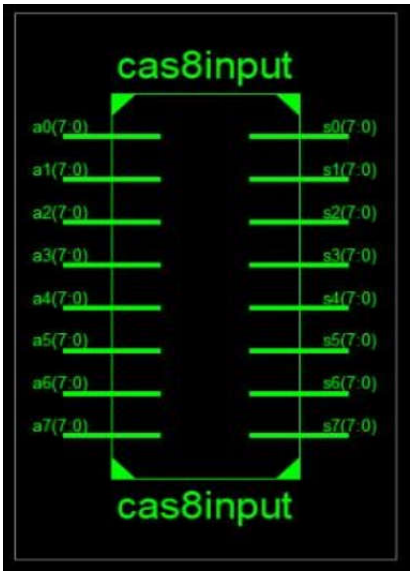

Fig 7: RTL schematic (synthesis)

RTL SCHEMATIC

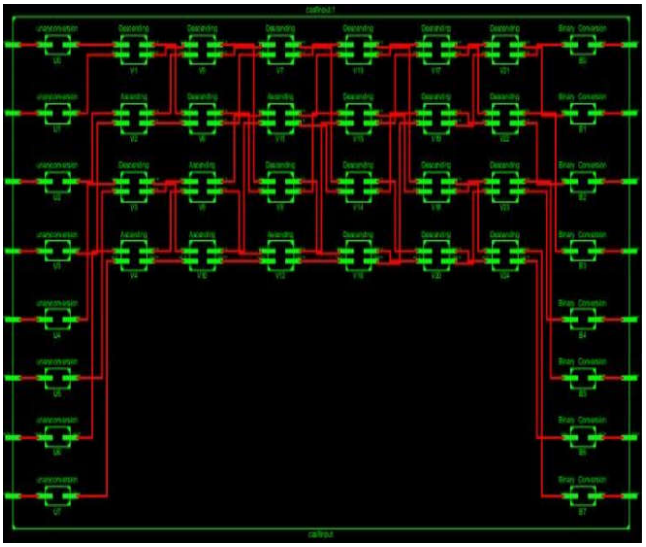

Fig 8: RTL schematic

SIMULATION RESULT

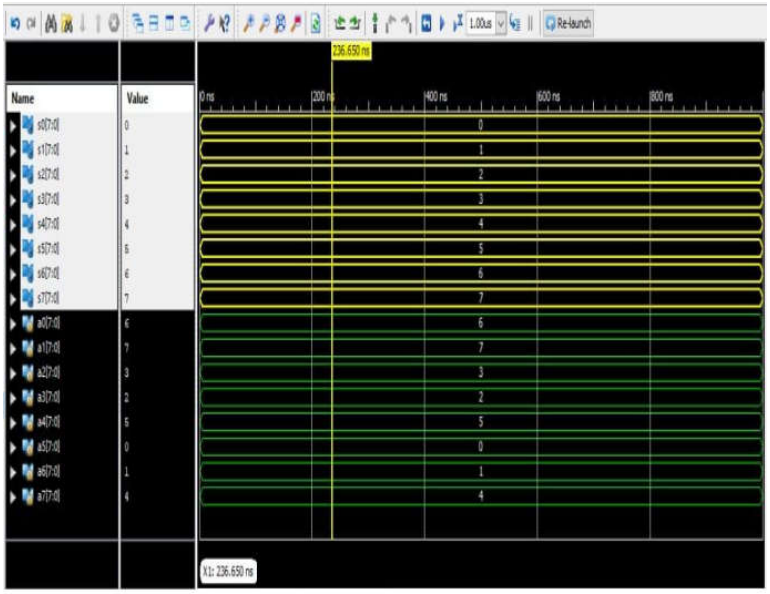

Fig 9: simulation results

CONCLUSION

Batcher sorting networks have been widely used in different applications. Their regular structure makes them popular for signal processing systems and communication switching networks. However, a conventional weighted binary-based implementation of a large sorting network is costly considering the large number of CAS units that such a network entails. The VLSI cost increases significantly with increasing resolution of the input data. The high hardware cost and the high power consumption of such networks restrict their application. This paper proposes an area and power-efficient implementation of sorting networks based on unary processing. The core processing logic consists of simple gates and is independent of the resolution of data. The only overhead in the approach, the cost of converting data from/to binary, is small. More than 90% area and power savings are observed when compared to the costs of a conventional weighted binary implementation. The penalty is latency. Processing digital unary streams requires a relatively long running time, e.g., more than 100 ns to process each set of input data. Although this is a 100× increase in latency over conventional weighted binary, this increase may be tolerable for many applications. For example, ten gray-scale high-definition (HD) (1280 × 720) images or four gray-scale full HD (1920 × 1080) images can be processed per second with the proposed scheme for a task such as median filtering, when operating on 256-bit long unary streams. In spite of the latency, a 90% decrease in power consumption might often make this a winning proposition. To mitigate the latency of the approach, 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. For example, more than 1000 gray-scale HD images or 400 grayscale full HD images can be processed per second with the proposed time-based unary implementation of the 3×3 median filtering at the cost of only 1% loss in accuracy.

FUTURE SCOPE

In the future work, we will explore other applications of sorting based on unary processing, for instance, in hardware implementations of weighted and adaptive median filters. We will also explore applications in communications and coding.

REFERENCES

1) "Arbitrary size bitonic (ASB) sorters and their applications in broadband ATM switching," in Proc. IEEE 15th Annu. Int. Phoenix Conf. Comput. Commun., Mar. 1996, pp. 454–458. 1. J. P. Agrawal.
2) S. W. A.-H. Baddar and B. A. Mahafzah, "Bitonic sort on a chainedcubic tree interconnection network," J. Parallel Distrib. Comput., vol. 74, no. 1, pp. 1744–1761, Jan. 2014.
3) Alaghi, W.-T. J. Chan, J. P. Hayes, A. B. Kahng, and J. Li, "Trading accuracy for energy in stochastic circuit design," J. Emerg. Technol. Comput. Syst., vol. 13, no. 3, p. 47, May 2017.
4) Alaghi and J. P. Hayes, "Survey of stochastic computing," ACM Trans. Embedded Comput. Syst., vol. 12, no. 2s, pp. 92:1–92:19, 2013.
5) Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in Proc. 50th ACM/EDAC/IEEE DAC, May 2013, pp. 1–6.
6) V. Brajovic and T. Kanade, "A VLSI sorting image sensor: Global massively parallel intensity-to-time processing for low-latency adaptive vision," IEEE Trans. Robot. Autom., vol. 15, no. 1, pp. 67–75, Feb. 1999.
7) D. Brown and H. C. Card, "Stochastic neural computation. I. Computational elements," IEEE Trans. Comput., vol. 50, no. 9, pp. 891–905, Sep. 2001.
8) D. Brown and H. C. Card, "Stochastic neural computation. II. Soft competitive learning," IEEE Trans. Comput., vol. 50, no. 9, pp. 906–920, Sep. 2001