International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

Using Stochastic Computing To Implement Logical Bit Arithmetic Operations and Design of an Architecture for Synthesizing Polynomial Functions

GATTU ASA PRASANTHI [1], *CH.SRI GIRI* [2]

[1]*PG Student, Dept. of ECE,* GODAVARI INSTITUTE OF ENGINEERING AND TECHNOLOGY, Rajahmundry, East Godavari, *India.*

[2]*Asst. PROFESSOR. Dept. of ECE,* GODAVARI INSTITUTE OF ENGINEERING AND TECHNOLOGY, Rajahmundry, East Godavari, *India.*

ABSTRACT:

Stochastic computing is a random probability distribution; it performs on low cost, low power and low error tolerance. Most of the digital systems operate on binary representation of data i.e., binary radix. In this binary radix representation higher order bits weighted more than lower order bits. In proposed approach we implement logical bit arithmetic operations using stochastic logic for better performance and the precision length of bit stream increases for better precision and increase in computation time. Compared to conventional binary radix representation stochastic logic is easy to implement and they occupies less area ,power reduction and hardware cost is low, Complex operations can be performed in simple stochastic logic.

INTRODUCTION:

Now a day, modern computing is constructing with the requirements like small size, low power consumption and high reliability. So in this paper we proposed a stochastic computing arithmetic operation. Stochastic computing operates on low area; low power and high latency. Stochastic computing paradigm is a low cost method alternative to conventional binary representation. The computation performed on data represents in probability distribution usually randomized bit streams. Stochastic computing can be applied in the fields like image processing-(edge detection, image thresholding, mean filtering…etc.,), neuron networks, error correction techniques(Ldpc

decoding), Nano technology and signal processing small fluctuations of errors tolerated but large errors are catastrophic. Soft errors caused by ionizing radiation when increasing a semiconductor is a main thing in circuits operate in harsh environments. Randomness is a main concern in stochastic computation. Randomness available in broad casting areas like communication and cryptography can perform with low complexity. Good pseudo randomness simulation obtains in random physics, quantum physics and biology.

We propose a new technique for computation is stochastic logic. Stochastic logic designs transform definite inputs and outputs Boolean integers and fractional integers are same. Synthesizing the circuits can be perform on logic to produce probability values input and ouput. The major approach is stochastic computing is applicable for randomized algorithms. The proposed approach we provide better error tolerance, low area and high power of design to implement8-bit arithmetic operations using stochastic logic computation. To implement this architecture introduce a new 8 bit accurate stochastic circuits addition, multiplication, subtraction and division arithmetic operations performed. Stochastic computing requires increase in the precision requires an exponentional increase in bit stream length and corresponding exponential increase in computation time to change the numerical precision of stochastic computation from 4 to 8 bits requires increasing bit-stream length from 2^4 =16 bits to 2^8 = 256 bits.

Related work:

Development of SC is started in 1956's by von Neumann; they defined fundamental concepts of probabilistic logic design. Modular redundancy technique is majorly used for fault tolerance. Memory based subsystems and communication purpose error correcting codes are used for both on chip and off chip. Probabilistic methods are appearing in system design and circuit level synthesizing. The main aim to applying probability is characterizing the unceratinity.For better performance of error tolerance through software mechanisms statistical timing analysis used. Error tolerance is mainly considered in application field. The proposed approach is mainly for less power consumption in hardware design. It is also applied to computing applications like Monte Carlo

simulations and they are tolerated errors. Presented a specific architectural design for data path computations, the main contribution is design a stochastic architecture is to detect the fault errors in previous approach. The new logic synthesis methodology the computation is performed in statistical probability distributions. Randomness process performed in serial or parallel bit streams in bit level. Accurate results depend on the statistics probabilities not their bit level. In the real valued based applications computation perform in analog character but they are implemented in digital components. This is for hardware based methods for error tolerance and memory based subsystems (error correcting codes).the proposed methodology is synthesizing the arbitrary polynomial functions implemented through Stochastic operation and also arbitrary continuous non polynomial expressions can be obtain through stochastic logic. Complex operations performed through probabilistic methods of stochastic operation.

Preliminaries:

Stochastic logic:

Stochastic operations performed at logic level in randomized bit streams in two ways either serial or parallel. Serial bit streams signals are performed in probabilistic in time and parallel bit streams they are performed in space. The bit streams carrying digital values zeros and ones and they are proceed to binary logic gates AND, OR. The signal is obtain through statistical distribution through logical values. Computation in Boolean domain represent probabilistic in real domain. Randomized bit streams the stream 'N' bits containing N1 ones and N-N1 zeros and stochastic number denoted as N1/N treated as probability. The main advantage of a stochastic computation is highly tolerated fault errors.

Existing method:

Conventional binary representation the logical multiplication and addition/subtraction performed in Boolean domain. In such cases the single bit flips the error result is high. For example x1=(0.110)2,x2=(0.100)2 are inputs and output isx3=(0.100)2 .the same process x2=(0.011)2 single bit flip the resulted output is x3=(0.010)2.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

Fig.1: conventional binary implementation

Fig.2: conventional binary implementation when one single bit flips error

Proposed method:

Stochastic logical computation process the logical synthesizing arithmetic operations are obtain in probabilistic domain. The arithmetic operations are simple logical multiplication, addition. stochastic computing there are two representations available, they are unipolar representation a real-valued number x $(0 \leq x \leq 1)$ is represented by stream in which each bit has probability x of being one and probability $1-x$ of being zero. In the "bipolar" representation, a real-valued number y $(-1 \leq y \leq 1)$ is represented by a stream in which each bit has probability $(y + 1)/2$.The main advantage of using Stochastic bit streams is the lack of a place value as compared to the conventional binary MSB and LSB.the environment gets corrupted by bit flips and suppose that the significant bit gets flipped with binary radix encoding. Here, the relative error will be 2^m-1/2^m =1/2. However, in stochastic encoding, the data is represented as the fractional weight on a bit stream of length 2^m. Thus the single bit flip changes the result only by ½^m.

Conventional binary radix is a positional encoding and maximally compressed. Stochastic is a uniform encoding and uncompressed.

Multiplication:

Stochastic multiplication can be implemented with less hardware compared to conventional representation. Multiplication process need only one AND gate. Assuming that x1, x2 two input independent bit streams, the output bit stream x3 is

X3=p (x3=1) = p (x1=1 and x2=1)

=P (x1=1). P (x2=1)

=x1.x2

Consider two uncorrelated, independent inputs x1=10110111, x2=00111001

Input probabilities of bit stream length m=8, the resolution is 1/8.

Probabilistic representation= number of one's / total stream bit stream length.

P(x1) =6/8, p(x2) =4/8 then

P(x3) =p(x1 and x2) =3/8=00110001

Stochastic multiplication obtains only in unipolar representation and inputs are uncorrelated or indepented bit streams. Resulted output x3=x1.x2. In case the inputs are correlated bit streams then the resulted output is same .stochastic multiplication of two correlated inputs x1=10110111, x2=10110111then x3=101101111 so they didn't applicable to multiplication process. Multiplication is complex in binary positional representation and easy in stochastic computation. Prior approach imposes a computational complex in arithmetic for each operation essence operands decoded and weighting higher order bits high and less in lower order bits.

The proposed approach is uniform and no re-encoding are required to operate the value

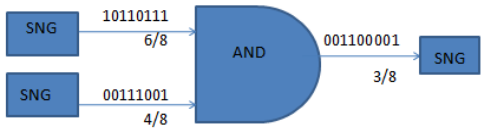

FIG.3:STOCAHSTIC MULTIPLICATION USING AND gate

Stochastic representation in single bit flip result the accurate value 3/8≈2/8 .stochastic representation bit flip result will be accurate error result. In case of conventional representation x1= (0.110)2, x2= (0.100)2 the resulted output x3=(0.100)2,when single bit flips x1=(0.110)2 and x2=(0.010)2 output is x3=(0.010)2 resulted output is not exact or accurate value. The output fault error is high.

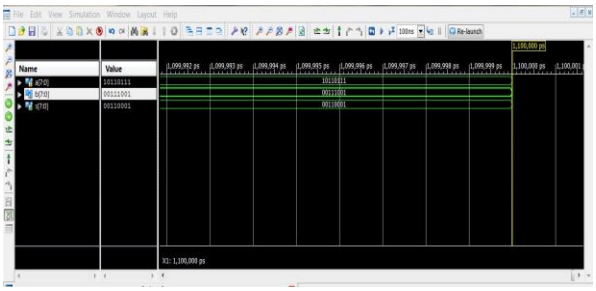

Fig.4: simulation results of stochastic multiplication

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

Scaled addition:

General addition is not feasible to add two probability values directly, the result value greater than one, cannot be represented probability value. So the proposed approaches perform scaled addition. Stochastic operation of scaled adder operating on real valued numbers. Consider two inputs multiplexer the output value based on third selecting input value.

Assuming two x1, x2 and S are independent stochastic bit streams the output bit stream x3 is

X3=p(x3=1)

=p(s=1 and x1=1) + p(s=0 and x2=1)

=p(s=1) p(x1=1) + p(s=0) p(x2=1)

=s.x1+ (1-s).x2

Stochastic addition performed independent inputs.x1=01000000, x2=10110110, s=00100001 output x3=10010110.

P(x3) =p(s=1) p(x1=1) +p(s=0) p(x2=1)

=p (2/8) p (1/8) +p (2/8) p (5/8)

P(x3) =p (4/8)

Stochastic representation of multiplexer scaled addition performs scaled factor s for x1 and (1-s) for x2.

The conventional representation of MUX implemented is x3=(x1^ s) w(x2 ^-s).

Consider two independent bit streams x1=(0.001)2 ,x2=(0.101)2, selecting input s=(0.010)2 output is x3=(0.100)2.

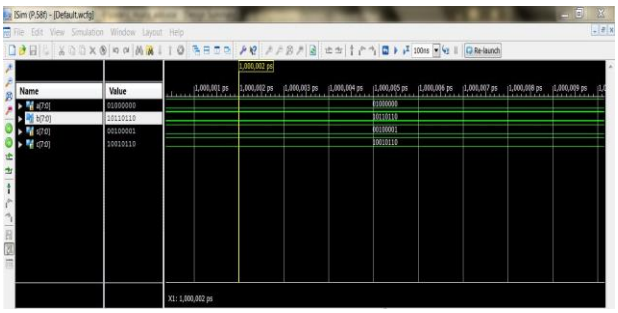

Fig.5: simulation results of stochastic addition .

Example p(x1) =01000000=1/8, p(x2) =10110111=6/8, p (s) =00100001=2/8, the resulted output p(x3) =10010110=4/8. Stochastic scaled addition using MUX ,when single bit flip error result the accurate or exact value When conventional representation single bit flips error is x1= (0.001)2, x2= (0.110)2, s=(0.010)2 then x3= (0.011)2 the bit flip error is high.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

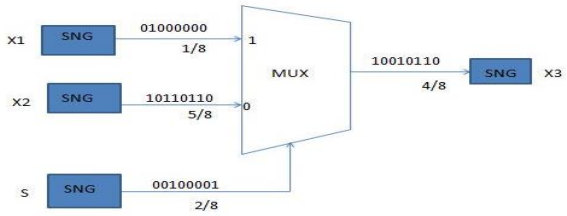

FIG.6: SCALED ADDITION USING MUX

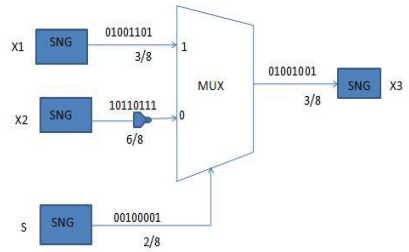

Fig.7:SCALED SUBTRACTION USING MUX

Scaled subtraction:

Scaled subtraction in stochastic logic implementation using MUX and not gate, subtraction can be obtained in bipolar format because negative value representation can be done in bipolar format. Unipolar format negative representation bit streams cannot be done. The scaled subtraction inputs are bipolar format and selective line input will be unipolar format. Scaled subtraction is similar to scaled addition only one input stream NOT gate will be connect to MUX.based on this logic subtraction performs as

P(x3) =p(x1 =1) p(s=1) —p(x2=0) p(s=0)

=x1.s-(1-s).x2

Assuming p(x1) =0100110=4/8, p(x2) =10110111=6/8, p(s) =00100001=2/8, the output p(x3) =010010001=3/8

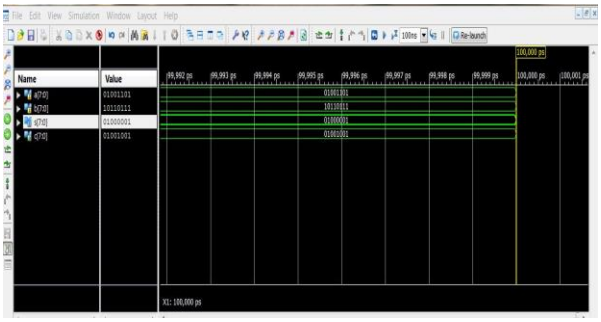

Fig.8: simulation results for stochastic subtraction

Consider p(x1) =01001101=4/8, p(x2) =10110111=6/8, p(s) =00100011=3/8, the single bit flips error p(x3) =01001001=3/8.single bit flips accurate result.

Stochastic division:

In this paper proposes a new division technique is CORDIV (correlated division) correlated bit streams are inputs. Area and

cost is low in CORDIV divider other than conventional dividers.CORDIV operate on fallowing two keys 1.The division probability px1/x2 defines the x1,x2 division probability px1,x2/px1

2. Correlated inputs x1, x2 efficiently transform this division operation px1/px2.

This process is achieving better accuracy.x1, x2 are maximally correlated bit streams if px1<px2 then px1x2=px1

It reduces toPx1/x2=px1/px2

Correlated division process only one random number generator is used so the area and cost is low.

If correlated division condition Z=x1/x2.the probability representation is Pz=px1/px2.the selective line l=1 MUX is controlled by x2, when x2=1.same way x2=0 the output result stored in D-flip-flop previous bit. This probability is PDFF=px1/px2.the output MUX is pz=px2.px1/px2+ (1-px2).pDFF=px1/px2. When l >1padding memory can be realized in several ways. It can simply be an l bit lfsr that stores l consecutive zeros in z.D-flip flop is feedback in MUX and x2 is 0 MUX select the line from D-flip flop.z padded memory is continuously stored consecutive 0s in shift register. To avoid such repetition, the divider must have large l bit padding memory when x2 have many consecutive zeros. Large Padding memory adds significantly l increase. In some cases for short stochastic numbers l is not usually preferable .l clock cycles needed to warm up to fill the effective bits in padding memory.

Assuming correlated inputs x1=1001 0100 0000 0000=3/16, x2=1001 1100 0001 0000=5/16, the output pz=px1/x2=3/5=10/16=1111 0111 1110 0000.

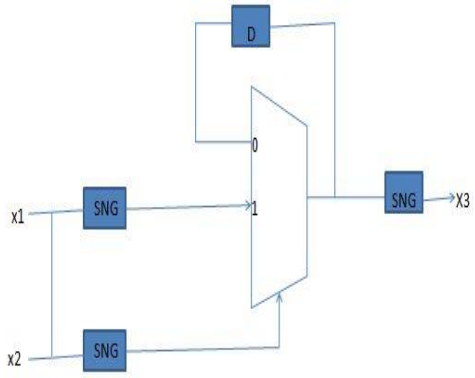

Fig.9: STOCHASTIC DIVIDER CIRCUIT

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 15
May 2018

The advantage of division circuit is conditional probability of x1, x2 and quotient x1x2 and the probability of x2.

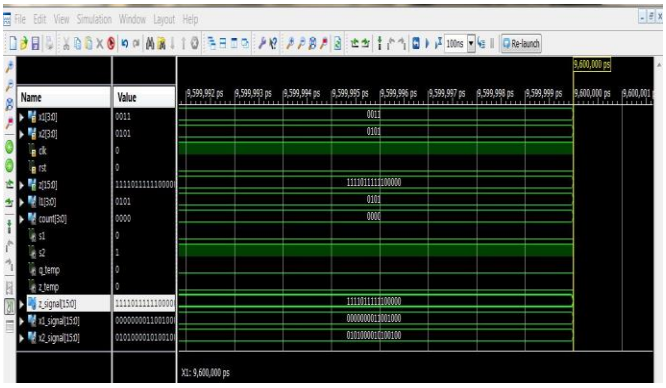

Fig.10: simulation results for division circuit

Synthesizing functions:

Consider the function y=x1x2S+x3(1-S) implementing these functions using stochastic arithmetic logic circuits.

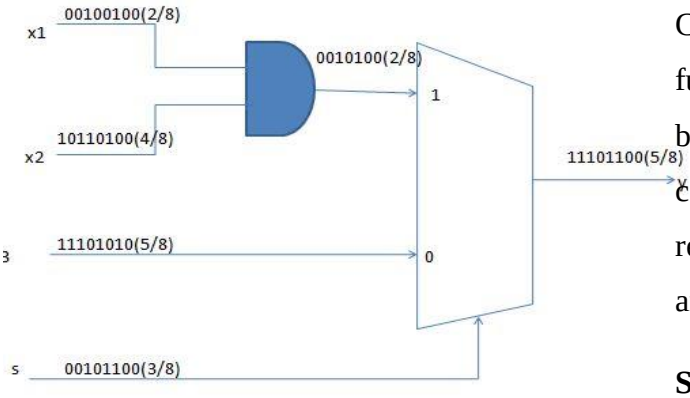

Fig.11: stochastic implementation function y=x1x2s + (1-s) x3.

Consider SNG inputs x1=2/8, x2=4/8, x3=5/8 and selective line s=3/8 and output y=5/8. Stochastic circuit function is implemented by using stochastic AND,MUX.

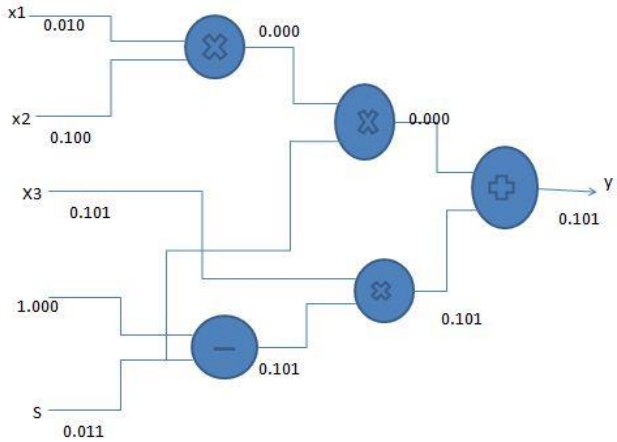

Fig.12: conventional binary representation

Conventional binary representation using function y=x1x2S + (1-s) x3 is implemented by addition, subtraction and multiplication circuits. Compare to conventional representation stochastic is to implement is area occupation is less.

Synthesizing polynomials:

Stochastic computation performs in synthesizing polynomials by using adder, mux operations. The proposed method is for implementing arbitrary polynomial functions. Synthesizing polynomial coefficients must be less than 0 or greater than 1.The polynomial function maps from the unit interval to values in the unit interval. Large and low coefficients are implemented by stochastic logic. The first step is implement synthesizing polynomial is transforming a power-form polynomial into Bernstein polynomial. Its degree n is of the form

$$B_n(t) = \sum_{i=0}^{n} b_{i,n} B_{i,n}(t)$$

Where b i,n is Bernstein polynomial c oefficient

Converting power form polynomial of degree n, $g(t) = \sum_{i=0}^{n} a_{i,n} t^i$, in to Bernstein polynomial of degree n $g(t) = \sum_{i=0}^{n} b_{i,n} B_{i,n}(t)$. the conversion from a i,n polynomial into b i,n polynomial the closed form is

$$b_{i,n} = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{n}{j}} a_{j,n}, \quad 0 \le i \le n.$$

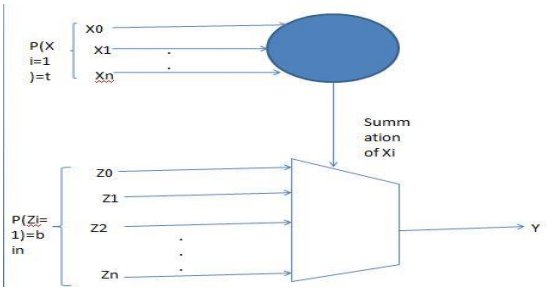

Fig.13: Stochastic Architecture for synthesizing polynomials

The stochastic consist of adder block and multiplexer. And inputs of the adder sets as t0,t1,.....tn ,the data inputs of MUX is z0,z1,....zn.The output of the adder is selective line input to the MUX. The input bit streams are independent and P (Xi=1) = xi ε (0,1)for $1 \le i \le n$ and P (Zi=1) =Zi € (0,1) for $0 \le i \le n$.

For example: g(t)=3/4-t+3/4t^2 maps the unit interval itself. It can be converted polynomial of degree 2

g(t)=3/4B0,2(t)+1/4B1,2(t)+1/2B2,2(t)

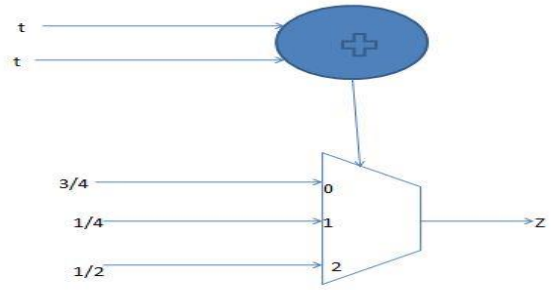

Fig.14: A generalized MUX implementing the polynomial function

RESULTS:

Stochastic implementation of area A(n) and D(n) delay product circuits compute the Bernstein polynomial degree n=2,3,4, and 5.then

A(n)D(n)2^M,

Where 2^M accounts for length of the bits tream.

Degree n of Bernstein polynomials	Area	Delay product
2	15	8
3	22	10
	40	17
4	49	20
5	58	20
6		

Table 1: area and delay product of circuit compute Bernstein polynomials of degree 2,3,4,5 and 6

n	M	AREA DELAY PRODUCT		STOCHASTIC
		CONVENTIONAL	STOCHASTIC	PRODUC/CONVENTIONAL PRODUCT
3	7	99207	28160	0.284
	8	152745	56320	0.369
	9	222615	112640	0.506
	10	310977	2225280	0.724
4	7	132276	87040	0.658
	8	203660	174080	0.855
	9	296820	348160	1.173
	10	414636	696320	1.1679
5	7	198414	125440	0.759
	8	305490	250880	0.986
	9	445230	501760	1.352
	10	621954	1003520	1.936

Table2: shows the area delay product for conventional and stochastic implementation for polynomials of degree n= 3, 4 and 5 resolutions 2^-M. M=7, 8, 9, 10,the last column shows the ratio of the two. We can see that for M ≤ 8.the

area and delay product of stochastic is always less than that of conventional implementation.

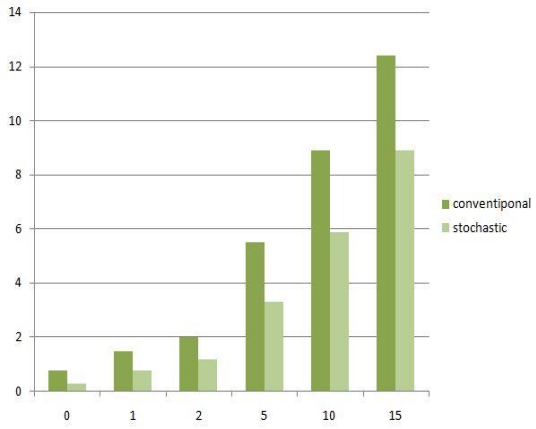

Fig: average output error rate of stocahstic and conventional implementations different error ratios.

Conclusion:

The computation that we are advocating in this paper has a pseudo analog character, reminiscent of computations performed by physical systems such as electronics on continuously variable signals such as voltage. In our case, the variable signal is the probability of obtaining a one in a stochastic yet digital bit stream. Indeed, our system is built from ordinary, cheap digital electronics such as CMOS. Digital constructs in CMOS operate on physical signals

such as voltage, of course. However, they are designed with the premise that these signals can always be unequivocally interpreted as zero or as one. This is certainly counterintuitive: why impose an analog view on digital values? As we have outlined in this paper, it might often be advantageous to do so, both from the standpoint of the hardware resources required as well as the error tolerance of the computation .Many of the functions that we seek to implement for computational systems such as signal processing are arithmetic functions, consisting of operations like addition and multiplication, subtraction and division. Complex functions, such as exponentials and trigonometric functions, are generally Computed through polynomial approximations, so through multiplications and additions. Proposed approach first to choose the problem of synthesizing arbitrary polynomial functions through logical computation on stochastic bit streams. The synthesis results for our stochastic implementations of a variety of functions are convincing. The area-delay

product is comparable to that of conventional implementations with adders and multipliers. Since stochastic bit streams are uniform, with no bit privileged above any other, the computation is highly error tolerant. As higher and higher rates of bit flips occur, the accuracy degrades gracefully. Indeed, computation on stochastic bit streams could offer tunable precision: as the length of the stochastic bit stream increases, the precision of the value represented by it also increases. Thus, without hardware redesign, we have the flexibility to tradeoff precision and computation time. In contrast, with a conventional binary radix implementation, when a higher precision is required, the underlying Hardware has to be redesigned. Note that we have been evaluating our stochastic implementations under the conservative assumption that the clock rate will be the same as that of a conventional implementation. However, with much simpler hardware— for instance a single AND gate performing a complex task like multiplication — we could potentially implement computation with much higher clock rates, particularly if it is pipelined

References:

1. Nepal, K., Bahar, R. I., Mundy, J., Patterson, W. R., and Zaslavsky, A. 2005. Designing logic circuits for

probabilistic computation in the presence of noise. In *Proceedings of the Design Automation Conference*.

485—490.

2. oppelbaum, W. J., Afuso, C., and Esch, J. W. 1967. Stochastic computing elements and systems. In *Proceedings*

of the AFIPS Fall Joint Computer Conference. 635—644.

3. M. Mitzenmacher and E. Upfal, Probability and Computing : Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, 2005

4. B. Gaines, “Stochastic computing systems,” in Advances in Information Systems Science. Plenum, 1969, vol. 2, ch. 2, pp. 37—172.

5. S. Narayanan, J. Sartori, R. Kumar, and D. Jones, “Scalable stochastic processors,”

in Design, Automation and Test in Europe,2010, pp. 335—338..

6. Naderi, A. et al., "Delayed stochastic decoding of LDPC codes," *IEEE Trans. Signal Proc.*, vol. 59, pp. 5617-5626, 2011

7. P. Li and D. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *Proc. Int. Conf. Comput. Design*,

2011, pp. 154—161.

8. A. Alaghi and J. Hayes, "Exploiting correlation in stochastic circuit design," in *Proc. IEEE 31st Int. Conf. Comput. Design*, 2013, pp. 39—46

9. K. Kim, J. Lee, and K. Choi, "An energy-efficient random number generator for stochastic circuits," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 256—261.

10. pages 261-267,1994. S.W. Golomb, *Shift Register Sequences,* Holden-Day, San Francisco, 1967.

11.J. Tomberg and K. **Kaski,** *"Feasibility of Synchronous Pulse-Density Modulation Arithmetic in Integrated8*May1992.