Leading one detectors and leading one position detectors – an evolutionary design methodology

Détecteurs de bit 1 principal et détecteurs de position de bit 1 principal – une méthodologie de conception évolutionnaire

K. Kunaraj and Dr. R. Seshasayanan *

Design of leading-one detector (LOD) and leading-one position detector (LOPD) are important as they are used for the normalization process in floating-point multiplication, floating-point addition/subtraction and in logarithmic converters. In this paper, the authors propose various gate-level architectures for LOD and LOPD. The LOD and LOPD circuits are evolved using the evolutionary algorithm (EA) and using the evolved lower-order gate structures, various higher-order circuits are constructed. To obtain better results, the EA is modified and a novel shuffling operation is performed to prevent the algorithm from settling in the local minima. Then the constructed LOD and LOPD circuit is synthesized using Cadence® RTLCompiler® using TSMC 180nm library. The LOD and LOPD circuits can be implemented in an Application Specific Integrated circuit (ASIC) or in a Field Programmable Gate Array (FPGA), and hence it is independent of the technology library. Perhaps the evolution can also be made as an intrinsic process during the application run time and the evolved best gate structure can be chosen. We restrict this paper to the extrinsic evolution of LOD and LOPD gate level architectures.

La conception des détecteurs de bit 1 principal (LOD) et des détecteurs de position de bit 1 principal (LOPD) est importante car ceux-ci sont utilisés dans le processus de normalisation lors de la multiplication et l'addition/soustraction en virgule flottante, ainsi que dans les convertisseurs logarithmiques. Dans cet article, nous présentons diverses architectures à portes logiques pour LOD et LOPD. Les circuits LOD et LOPD évoluent en utilisant l'algorithme évolutionnaire (AE) et les structures de portes logiques d'ordre inférieur évoluées. Divers circuits d'ordre supérieur sont construits. Pour obtenir de meilleurs résultats, l'AE est modifié et une nouvelle opération de brassage est réalisée pour éviter à l'algorithme de passer par un minimum local. Les circuits LOD et LOPD construits sont ensuite synthétisés en utilisant Cadence® RTLCompiler® qui utilise la librairie TSMC 180nm. Les circuits LOD et LOPD peuvent être implémentés en ASIC et en FPGA et sont ainsi indépendants de la librairie de la technologie utilisée. Peut-être que l'évolution peut aussi s'effectuer comme un processus intrinsèque pendant l'exécution de l'application et que la meilleure structure de portes évoluée peut être choisie. Dans cet article, nous nous limitons à l'évolution extrinsèque des architectures LOD et LOPD à portes logiques.

Keywords: leading one detector, leading one position detector, genetic algorithm, evolvable hardware.

I Introduction

Floating-point multiplication and addition are the most common floating-point operations used in various digital signal processing techniques. The speed of the Leading-Zero Anticipator (LZA) is important in designing high speed floating-point addition and its significance is highlighted in [1]. A clear analysis of various application data show that the signal processing algorithms require, on average, 40% multiplication and 60% addition operations [2]. As the floating-point multiplication and addition is the most complex part of various DSP algorithms, it needs to be optimized to overcome the critical bottlenecks viz., latency and area. LODs and LOPDs determine the location of the most significant one or a leading bit in a given binary. The position of leading one is used for the normalization process and shifting process in the floating-point multiplication, floating-point addition and also in binary logarithmic converters [3]. Over the years, the very large scale integration (VLSI) community has developed various architectures for LODs and LOPDs, aimed primarily at reducing the overall latency [4]. Research has been going on to evolve various combinatorial circuits in a constrained space with minimum effort [5]. A modular circuit for determining the leading one bit in a binary word is discussed and it is used for floating-point normalization [6]. Both leading zero and leading one anticipation circuits are discussed by Schmookler in [7] which uses AND gate in the final stage to equalize the total delay. Ji, Rong et al [8] provide useful analysis of error in the pre-encoding logic in the leading one predictor module proposed in [19]. A Hybrid LZA is proposed in [9] focusing on the improvement of delay and hardware area in floating-point addition. A concurrent position correction logic, operating in parallel with the LOP, to detect the presence of that error and produce the correct shift amount is discussed in [10]. By using binary trees and detecting larger groups of zeros, a better speed up can be obtained [11], [12]. Mahdiani et al. proposed evolutionary algorithm based bio-inspired computational blocks which are efficient in terms of area, speed and power consumption [13]. Various evolved gate level architectures of binary multiplier are proposed in [14] but the efficiency and tradeoffs with the existing method are not discussed. An evolvable hardware structure based on a Boolean functions network is implemented with the basic multiplexer circuit and configured by a hardware genetic algorithm in [15]. A complete survey of evolvable hardware with emphasis on some of the latest developments shows certain performance benefits exceeding the traditional methods [16]. Nancy Forbes discusses the evolution on a chip to optimize the circuit design based on firefly machine [17]. The on-chip adaptation and self configuration of reconfigurable hardware through evolutionary algorithms is termed as evolutionary hardware [18]. Adrian Stoica proposes an evolution-oriented field programmable transistor array (FPTA), reconfigurable at transistor level which allows evolutionary experiments with reconfiguration at various levels of granularity.

Manuscript received 20-May-2012; accepted 20-Jun-2013.

* K.Kunaraj is a Research Scholar and Senior Research Fellow, ECE, CEG with Anna University, Chennai – 600025, India. Dr.R.Seshasayanan is Associate Professor, ECE, CEG with Anna University, Chennai – 600025, India. Email: k.kunaraj@ieee.org, seshasayanan@annauniv.edu

Associate Editor managing this paper's review: Ali Miri

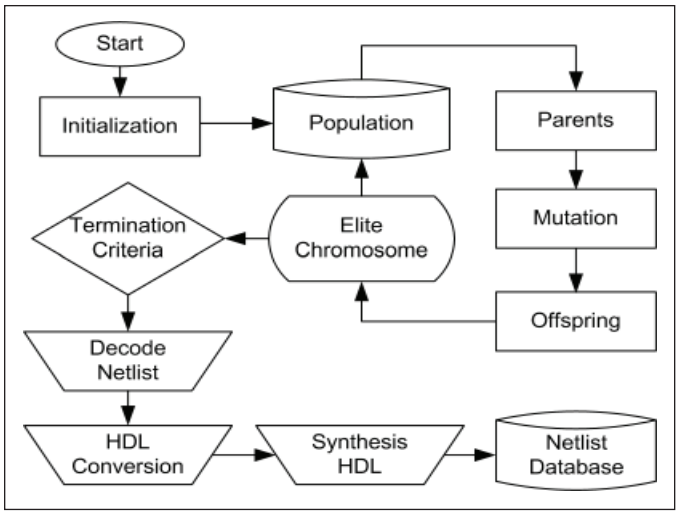


Figure 1: *GA for evolving LOD circuits*

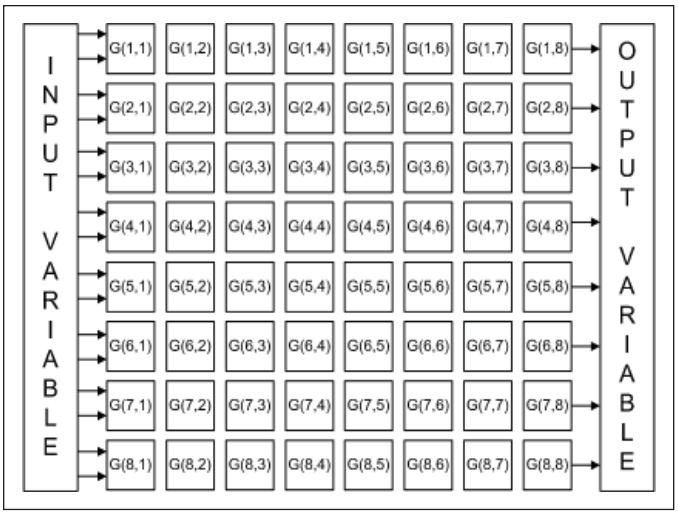


Figure 2: *Logic Element of dimensions 8X8 array*

The major contribution and objective of our work is to evolve and to construct various LOD architectures and to implement the evolved LOD circuit using Hardware Descriptive Language (HDL). Further a comprehensive analysis of power consumed, latency and gate count is done for various evolved LOD architectures. We propose a novel shuffling operation in the evolutionary algorithm to reduce the runtime of the algorithm and to prevent the algorithm from settling in the local minima. The strength and weakness of the evolutionary algorithm which is used for evolving the combinatorial circuits is discussed. The genetic algorithm for evolving LOD circuits is discussed in detail in section II. Section III analyzes the evolved gate structure of 4-bit LOD and the design of 8-bit, 16-bit, 32-bit and 64-bit LOD based on the evolved 4-bit LOD circuit. Section IV compares the synthesis results of varies LODs. In Section V we evolve 4-bit and 8-bit LOPD and construct higher order LOPDs and in section VI we compare the synthesis results of LOPDs. Section VII provides the conclusion.

II Genetic algorithm for evolving LOD and LOPD circuits

The GA is an adaptive search algorithm suitable for solving combinatorial optimization problems. Fig.1 shows the steps involved in evolving the LOD circuit. Based on the fitness function and the solution space, the GA finds the global maxima or the global minima. The convergence of the GA is guaranteed depending on the search space, the fitness function and the various genetic operations performed. An initial random population is generated and those populations which satisfy the fitness measure can be taken as the initial population. Thus using these initial populations, more offspring are generated and each offspring is checked using the fitness function.

The fittest offspring can be taken as the parent for the subsequent iterations and after performing genetic operation to the parent, the fittest individual is evolved. Each individual is termed as a chromosome which is a representation of the intended gate-level circuit as an array of bits or integers and the chromosome carries all the necessary information (logic gates and their interconnections) of a complete gate-level circuit. The genetic operation like cross-over and mutation is performed on the parent chromosome to obtain the child or offspring. To preserve elitism, only the best chromosome is selected and passed from one generation to the other. In crossover, the parts of chromosome are exchanged between the parents and in mutation the bits of the parent chromosome are inverted to obtain the child. The child chromosome resulting from the genetic operation can be taken as the parent chromosome for the next iteration. Multiple iterations are performed until we get the exact functionality of the circuit. The evolved gate structure which implements the complete functionality is saved and the best circuit is chosen taking into consideration the number of levels and gates.

II.A Gate array structure

As shown in Fig. 2 an 8X8 array of logic cell is formed and each logic cell can be any of the logic gates viz., AND, OR, NOT and WIRE. The first layer of 8 gates take the direct inputs (X0, X1, …, X8) and the subsequent layers can take input from the previous layers of gates and also can take the direct inputs (X0, X1, …, X8). Thus a total of 8 layers are formed and the binary representation of this array forms the chromosome. A higher flexibility is given to the gate structure to increase the possibility of evolving more gate level architectures for the given fitness function.

II.B Genetic operators (variation)

The mutation and crossover operators are used as the genetic operator. From the conducted experiments, it is clear that our problem converges better if we use mutation as the genetic operator rather than crossover. The mutation rate determines the time taken for the convergence of the algorithm and it is found that a 50% mutation rate is suitable for our problem. If the algorithm gets stuck in a local minima we shuffle the chromosome before passing the offspring to the next generation and this is called shuffling mechanism.

II.C Representation

This framework consists of a fixed number of 64 logic elements arranged in an array structure of 8X8 dimensions. A fixed length binary representation of the gate array structure forms the phenotype and the individual gate description forms the genotype. The number of logic elements in the gate array structure is fixed to 64 of dimension 8X8 array and each logic element can be any of the four gates viz., AND, OR, NOT and WIRE. The output of a particular gate can be connected to the output of the gate array structure or it can drive the input of the gates in the subsequent layer. Thus it improves the generation of more combinations of circuits for the same function. The logic elements are referenced by the position within the chromosome with circuit inputs encoded in the first section of chromosome. The encoding ensures that the number of inputs and outputs described by a chromosome remains consistent even after the genetic operation. Table 1 lists the parameter used in the genetic algorithm.

Table 1
Parameters used in GA

Sl. No.	Parameter	Value
1	Population Size	50
2	Mutation Rate	5% - 90%
3	Tournament Size	1-8
4	Elitism Size	1-2

Table 2
Fitness Table representation of fitness function of 4-bit LOD

Input				Output				Zero Flag
X_3	X_2	X_1	X_0	Y_3	Y_2	Y_1	Y_0	V
0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	1	X	0	0	1	0	0
0	1	X	X	0	1	0	0	0
1	X	X	X	1	0	0	0	0

Table 3
Fitness Table representation of fitness function of 4-bit LOPD

Input				Output		Zero Flag
X_3	X_2	X_1	X_0	Y_1	Y_0	V
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	X	0	1	0
0	1	X	X	1	0	0
1	X	X	X	1	1	0

II.D Fitness function

Each individual chromosome is taken and the set of all possible inputs are given and the corresponding outputs are evaluated. The degree of closeness of the obtained outputs with the actual outputs is calculated and it forms the fitness value of the individual chromosome. The fitness function is the required design to be obtained. For each chromosome the fitness function is calculated. The fitness measure is a measure between the fitness of each chromosome with the required fitness. The chromosome with the better fitness is chosen for the next generation and the process is repeated until the desired chromosome is obtained with the required fitness.

Fitness of the individual in a population is calculated using fitness function and we represent the fitness function as a fitness table. The fitness table of 4-bit LOD and 4-bit LOPD are shown in table 2 and table 3 respectively.

II.E Population

An initial 100 random-seeded chromosomes are selected and an initial evaluation process is done based on the fitness function. The selected best 50 chromosomes are taken as the initial population. The EA evolved the 4-bit LOD, 4-bit LOPD and 8-bit LOPD with only a random-seeded initial population. Hence it is clear that a 100% functional combinational circuit can be evolved with a random-seeded initial population.

III Evolved 4-bit LOD and higher order LODs

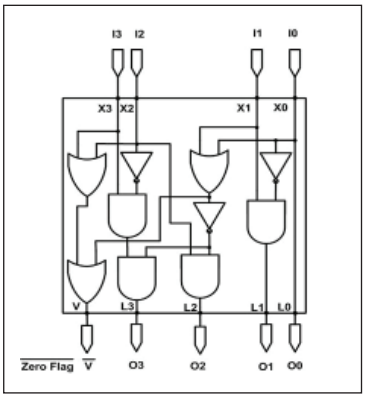


Figure 3: *Evolved four bit LOD*

In binary logarithmic circuits, the primary operation is to determine the integer and the fractional parts. The position of the leading one bit represents the integer part of the logarithm. The fractional part is obtained by shifting the input using the output from the LOD. So it is necessary to design a circuit which detects the leading one bit position with less hardware and consumes low power and operates with high speed. The LOD produces logic-1 in the output where the input has the leading 1 and all other outputs will be logic 0. The leading one detector discussed in paper [4] uses the multiplexers as the building block. Replacing the standard 2:1 multiplexer using the logic gates will be beneficial for reducing the delay.

In the proposed method, we evolve the architecture of the 4-bit LOD using the basic 2-input logic gates viz., AND, OR, NOT and WIRE and the evolved gate level circuit shows performance benefits. Using the evolved 4-bit LOD the higher order LODs of various sizes are to be constructed as discussed in [3]. In fig. 3, the best known evolved 4-bit LOD is shown.

III.A 8-bit LOD

For constructing an 8-bit LOD from the evolved 4-bit LODs, we require two 4-bit LODs and eight 2-input AND gates in the output stage. The zero flag of the most significant 4-bit LOD serves as an enable signal to the AND gates in the output stage. Thus a series of eight 2-input AND gates replaces the multiplexer from the output stage and reduces the overall latency of the 8-bit LOD circuit. Fig. 4 shows the architecture of the 8-bit LOD.

III.B 16-bit LOD

The architecture of a 16-bit LOD is similar to the technique discussed in [3]. The 16-bit LOD is constructed using five 4-bit LODs and sixteen 2-input AND gates. The first stage has four 4-bit LODs and the 16 bit input is partitioned into four groups of 4 bit each which are given to the first level of LODs. Hence the outputs of the first stage of LODs are the decoded binary value whose output is logic-1 in the corresponding position of the leading one of the individual 4-bit inputs given to each LOD. Fig. 5 shows the architecture of a 16-bit LOD circuit constructed using an evolved 4-bit LOD circuit. The next stage is a single 4-input LOD meant for selecting the leading one from the outputs of the previous stage. Hence among the four LODs from the first stage, the leading LOD which has the leading one is selected.

The final stage is the array of 2-input AND gate which is enabled by the output of 4-input LOD in the second stage. Hence the enabled group of four 2-input AND gates will carry the output of 4-input LOD in the first stage and the outputs of all other AND gates are zero.

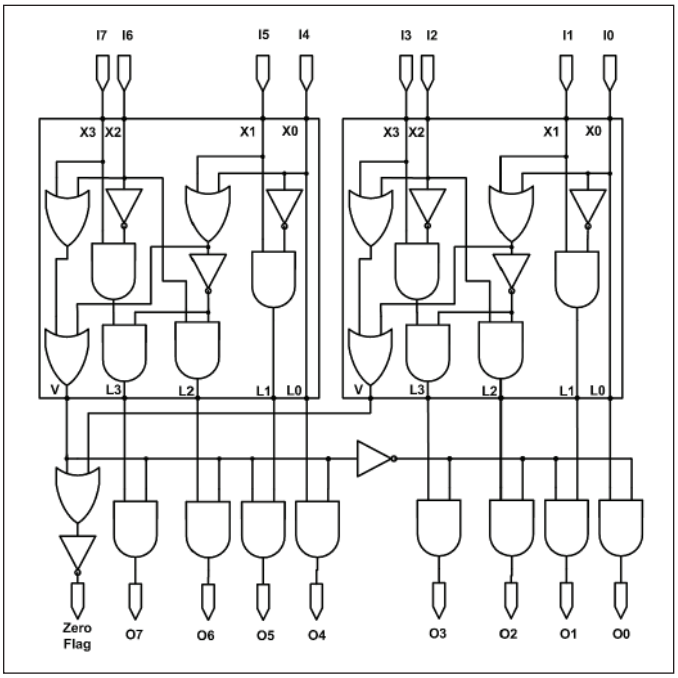


Figure 4: *8-bit LOD constructed using the evolved 4-bit LOD*

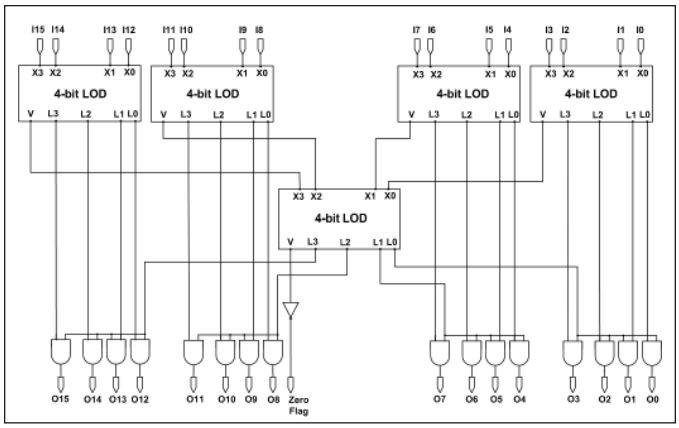


Figure 5: *Architecture of 16-bit LOD [3]*

Table 4
Results of various LOD circuits

LOD Circuit	Cell Area (μm^2)	Power (nW)	Delay (ps)
4-bit	019.051	0245.710	0296
8-bit	069.149	0745.000	0506
16-bit	153.821	1706.154	0831
32-bit	331.632	2795.282	0978
64-bit	673.142	5468.400	1303

III.C Architecture of 32-bit and 64-bit LOD

The 32-bit LOD is constructed similar to the 16-bit LOD and is shown in fig. 6. The first stage has eight 4-input LODs and the second stage has two 4-input LODs to determine the leading group of 4-bits from each group of 16 bits. The zero flag of the 4-bit LOD in the second stage can be used to find the leading one in the most significant 26-bit. The third stage has a series of 3-input AND gates similar to the last stage of 16-bit LOD discussed earlier. The zero flag from the 4-bit LOD in the second stage of the most significant 16-bit selects the corresponding output of the 4-bit LOD and all other outputs are made zero. A similar method followed for constructing a 32-bit LOD is used to construct a 64-bit LOD circuit from a 4-bit LOD circuit and is shown in fig. 7.

IV Comparison of synthesis results of LODs

Candence® RTLCompiler® with TSMC 180nm library is used to synthesize and to analyze the cell area and power consumed by the various LOD architectures. The results obtained for LODs of various sizes are shown in Table 4. The evolved gate level architecture shows promising enhancements in speed, power and area. Fig. 8 shows the plot of various synthesis results.

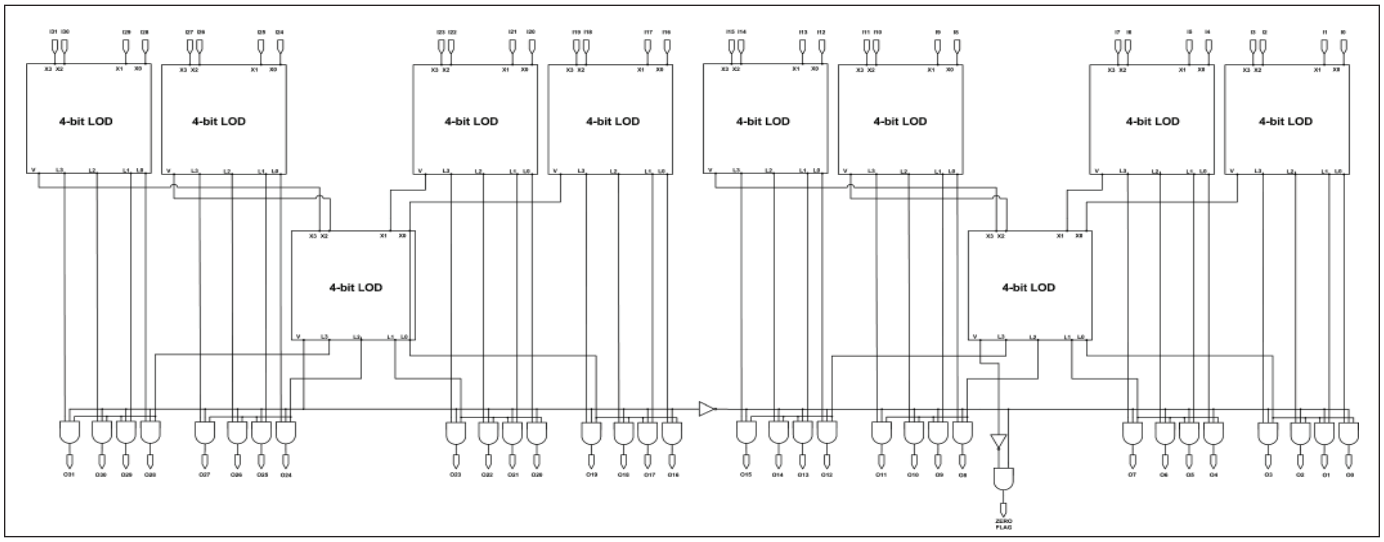


Figure 6: *32-bit LOD constructed from 4-bit LOD [3]*

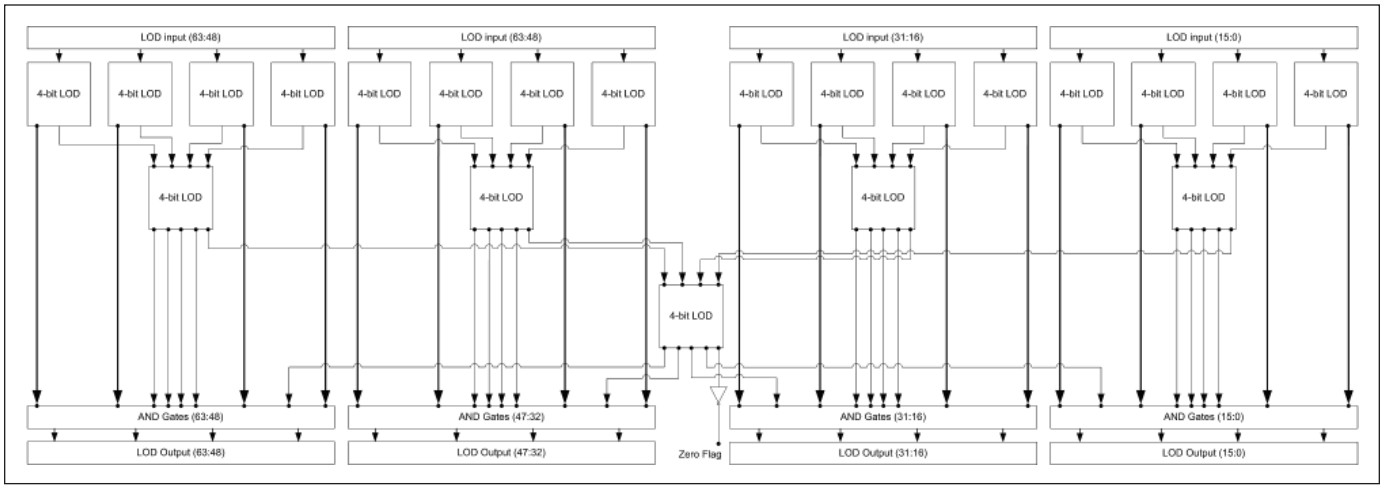
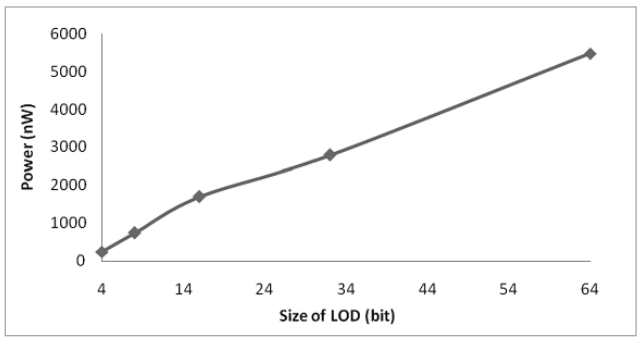
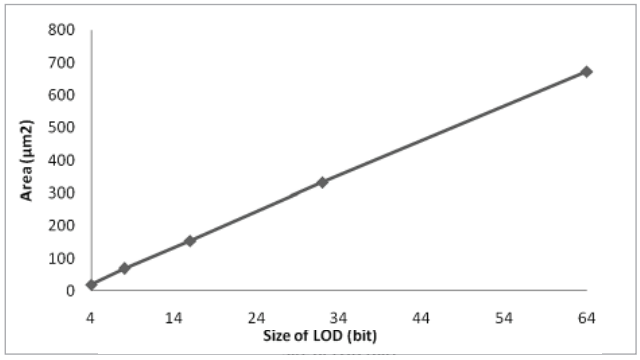


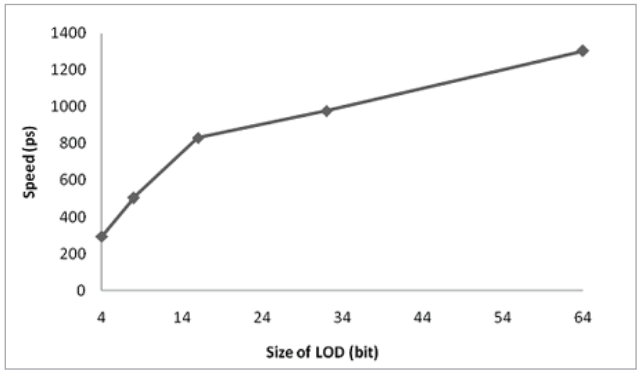
Figure 7:*64-bit LOD constructed from 4-bit LOD [3]*



(a)



(b)



(c)

Figure 8: *a) Plot of power for various sizes of LOD. b) Plot of cell area for various sizes of LOD. c) Plot of speed for various sizes of LOD*

V Evolved LOPDs and higher order LOPDs

We have evolved several circuits of 4-bit and 8-bit LOPDs and the higher order LOPD circuits are constructed from the best known configuration of the evolved lower-order LOPD circuits. Thus we have proposed various architectures for higher order LOPD circuits constructed using the evolved 4-bit LOPD and evolved 8-bit LOPD. The output value of LOPD is the position of the leading one in the input binary pattern. Additionally the flag bit is generated by doing OR operation of all of the input bits. If the flag bit is Logic-1, then the input values contain all zeros. We demonstrate a few evolved architectures of 4-bit LOPD and 8-bit LOPD using higher order LOPDs that can be constructed.

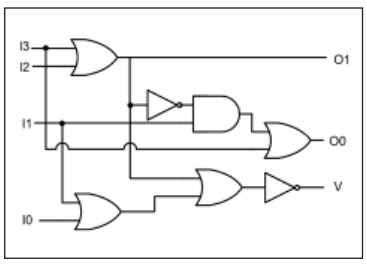


Figure 9: *Evolved four bit LOPD*

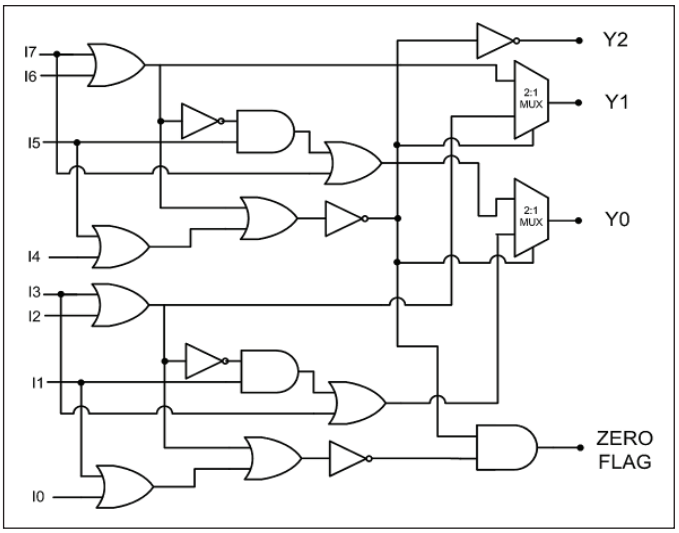


Figure 10: *8-bit LOPD using 4-bit LOPD*

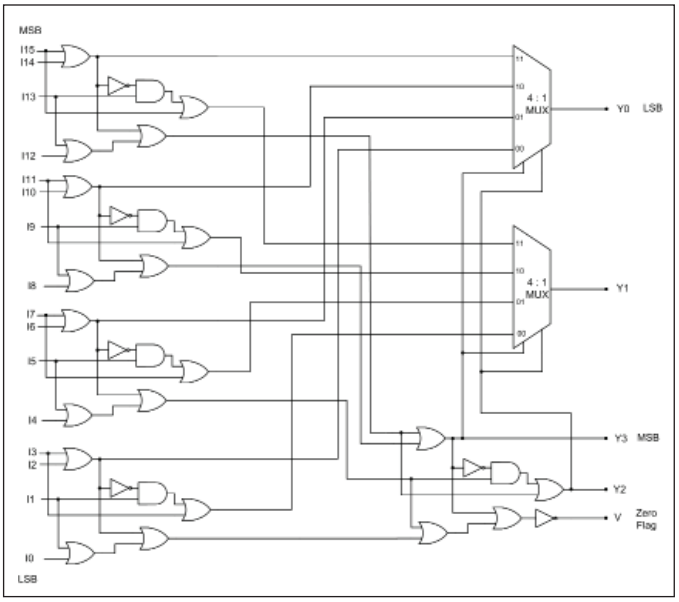


Figure 11:*16-bit LOPD using 4-bit LOPD*

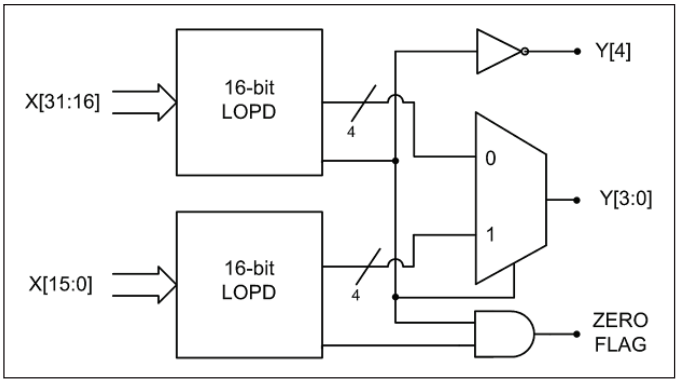


Figure 12: *32-bit LOPD using 4-bit LOPD*

V.A Evolved 4-bit LOPD circuit

Fig. 9 shows the logic diagram of an evolved 4-bit LOPD circuit. The basic 8x8 gate structure discussed earlier is used and with the help of the fitness table the 4-bit LOPD circuit is evolved.

Table 5
LOPD circuits constructed using 4-bit LOPD

LOPD Circuit	Cell Area (μm^2)	Power (nW)	Delay (ps)
4-bit	13.406	162.953	184
8-bit	45.158	643.699	540
16-bit	105.84	1267.23	585
32-bit	242.726	2834.23	1003
64-bit	519.322	6166.73	1224

Table 6
Comparison of various architectures of Evolved 8-bit LOPD circuit

LOPD Circuit	Cell Area (μm^2)	Power (nW)	Delay (ps)
LOPD8_1	539.881	35.280	423
LOPD8_2	**485.118**	**33.163**	**297**
LOPD8_3	539.881	35.280	423
LOPD8_4	539.881	35.280	423
LOPD8_5	534.624	35.986	448
LOPD8_6	**506.399**	**33.869**	**360**
LOPD8_7	539.881	35.280	423
LOPD8_8	539.881	35.280	423

V.B Construction of higher order LOPDs from the evolved 4-bit LOPD

The higher order LOPDs are constructed using the evolved 4-bit LOPD and multiplexer. The 8-bit LOPD circuit requires two 4-bit LOPDs and two 2:1 multiplexers as shown in fig. 10. Similarly a 16-bit LOPD circuit in fig. 11 needs five 4-bit LOPD circuits and two 4:1 multiplexers. Using these 8-bit and 16-bit LOPD circuits, the 32-bit and 64-bit LOPD circuits are constructed as shown in fig. 12.

V.C Synthesis results for various higher order LOPD circuits constructed using evolved 4-bit LOPD

Table 5 shows the cell area, power consumed and the speed of the various LOPD circuits.

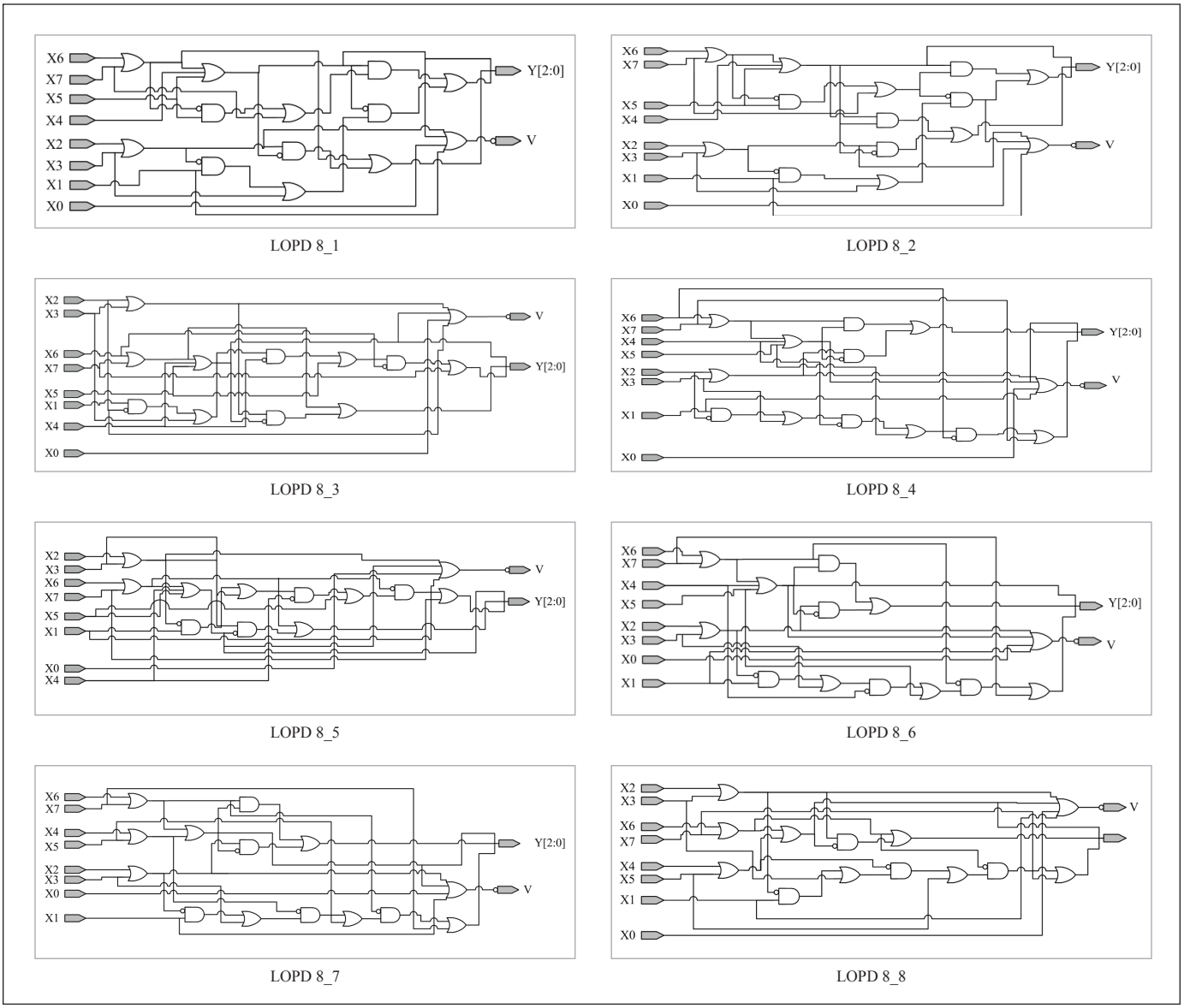


Figure 13: *Synthesized netlist of evolved 8-bit LOPD circuits*

Table 7
Comparison of synthesis reports of various LOPD circuits

LOPD Circuit	**Arch-1** (LOPDs using 4-bit LOPD)			**Arch-2** (LOPDs using 8-bit LOPD (LOPD8_2))			**Arch-3** (LOPDs using 8-bit LOPD (LOPD8_6))		
	Cell Area (μm^2)	Power (nW)	Delay (ps)	Cell Area (μm^2)	Power (nW)	Delay (ps)	Cell Area (μm^2)	Power (nW)	Delay (ps)
4-bit	13.406	162.953	184	Not Applicable			Not Applicable		
bit	45.158	643.699	540	33.163	393.999	297	33.869	418.342	360
16-bit	105.84	1267.23	585	91.022	994.689	846	92.434	1027.95	824
32-bit	242.726	2834.23	1003	213.091	2354.61	991	215.914	2400.16	969
64-bit	519.322	6166.73	1224	460.051	5142.7	1206	465.696	5226.66	1184

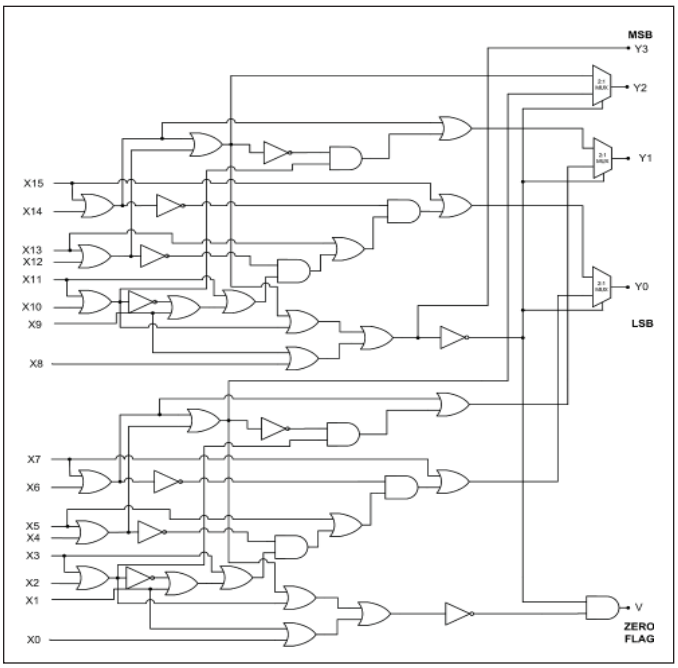


Figure 14: *16-bit LOPD using evolved 8-bit LOPD*

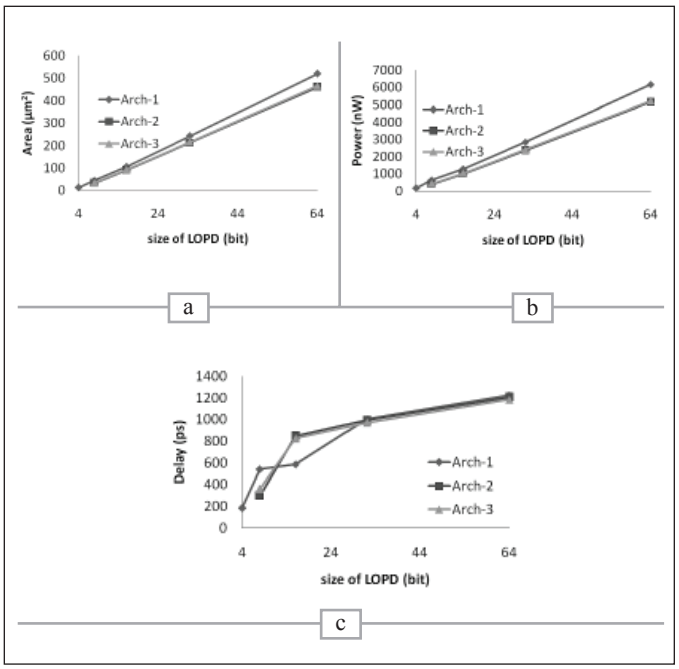


Figure 15: *a) Cell Area - comparative plot. b) Power-comparative plot. c) Delay - comparative plot.*

V.D Evolved 8-bit LOPD circuits

We have evolved various circuits for 8-bit LOPD and the best known eight circuit topologies are shown in fig. 13. Fig. 13 shows the synthesized netlist of the evolved 8-bit LOPD circuits.

All the evolved 8-bit LOPDs are synthesized using Cadence® RTLCompiler® and the results are listed in table 6. From the synthesized results it is clear that the circuits "LOPD_2" and "LOPD_6" show better performance in terms of cell area, power and speed. Hence we select those two LOPD circuits and we construct higher order LOPD circuits from those two elite LOPD circuits.

V.E Construction of higher order LOPDs from the evolved 8-bit LOPD

Higher order LOPDs can be constructed using 4-bit LOPDs or 8-bit LOPDs along with multiplexers. We propose two architectures for the construction of 16-bit LOPDs. Our first architecture utilizes five 4-bit LOPDs and two 4:1 multiplexers and the second architecture uses two 8-bit LOPDs and three 2:1 multiplexers with a 2-input AND gate for zero flag. Fig. 14 shows the architecture of a 16-bit LOPD constructed using the same methodology discussed earlier. Perhaps the higher LOPDs like 32-bit LOPDs and 64-bit LOPDs are constructed using the above discussed methodologies and the outputs are listed in table 7.

VI Comparison of various LOPD circuits constructed using 4-bit LOPD and 8-bit LOPDs (LOPD8_2 and LOPD8_6)

We compare the cell area, power and delay of various LOPD circuits constructed using 4-bit LOPD and 8-bit LOPDs (LOPD8_2 and LOPD8_6). From the comparison it is clear that "Arch-2" consumes less area and less power and is an area and power efficient architecture. For 8-bit LOPD circuit the "Arch-2" takes less delay and for 16-bit LOPD circuit "Arch-1" takes less delay.

The comparative plot of area, power and delay for various sizes of LOPDs are shown in Fig 15a, Fig. 15b and Fig. 15c respectively.

The constructed higher order LOPDs are synthesized and their synthesis reports are tabulated in Table 7.

VII Conclusions

In this paper, we have discussed an automatic and evolutionary approach to the design of various LOD and LOPD circuits. Though the higher order LOPD circuits cannot be completely evolved because of the limitation of the population size and search space, we have discussed few efficient methods to construct higher order LODs and LOPDs from the evolved lower order circuits. Thus the gate level evolution of LOD and LOPD circuits has proven performance benefits. All the circuits are synthesized using Cadence® RTLCompiler® with TSMC 180nm library. From the synthesis results, it is clear that the higher order LOPDs constructed using 8-bit LOPD (LOPD8_2) is more efficient in terms of cell area and power consumption and the higher order LOPDs constructed using 8-bit LOPD (LOPD8_6) is efficient in terms of delay. Perhaps, the 16-bit LOPD constructed using 4-bit LOPD has the least delay because of the optimized architecture. As the design complexity increases, convergence of evolutionary algorithm to suit the fitness function will take more time and it may settle to local maxima or minima and the shuffling mechanism introduced in the GA will answer this problem.

References

[1] H. Suzuki, H. Morinaka, H. Makino, Y. Nakase, K. Mashiko, and T. Sumi, "Leading-Zero Anticipatory Logic for High Speed Floating Point Addition," IEEE J. Solid-State Circuits, vol. 31, no. 8, pp. 1,157-1,164, 1996.

[2] F. Pappalardo, G. Visalli, and M. Scarana, "An application-oriented analysis of power/precision tradeoff in fixed and floating-point arithmetic units for VLSI processors," in Proc. IASTED Conf. Circuits, Signals, and Systems, pp. 416 – 421, 2004.

[3] Khalid H. Abed, Raymond E. Siferd, "VLSI Implementations of Low-Power Leading-One Detector Circuits" IEEE Southeast Conference, pp. 279 – 284, 2006.

[4] Oklobdzija, V.G., "An implementation algorithm and design of a novel leading zero detector circuit," Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on , vol., no., pp.391-395 vol.1, 26-28 Oct 1992.

[5] Vassilev, V.K.; Job, D.; Miller, J.F., "Towards the automatic design of more efficient digital circuits", 2000. Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware , vol., no., pp.151-160, 2000.

[6] P. Lee and A. Sartori, "Modular leading one detector for logarithmic encoder" Electron. Lett. 34, 727, 1998.

[7] Schmookler, M.S.; Nowka, K.J., "Leading zero anticipation and detection-a comparison of methods", 2001. Proceedings. 15th IEEE Symposium on Computer Arithmetic, vol., no., pp.7-12, 2001.

[8] Ji, Rong; Ling, Zhiqiang; Zeng, Xianjun; Sui, Bingcai; Chen, Liang; Zhang, Junfeng; Feng, Yingjie; Luo, Gang, "Comments on "Leading-One Prediction with Concurrent Position Correction", IEEE Transactions on Computers, vol.58, no.12, pp.1726-1727, Dec. 2009.

[9] Verma, A.; Verma, A.K.; Brisk, P.; Ienne, P., "Hybrid LZA: A near optimal implementation of the Leading Zero Anticipator," Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific , vol., no., pp.203-209, 19-22 Jan. 2009.

[10] Bruguera, J.D.; Lang, T., "Leading-one prediction with concurrent position correction", IEEE Transactions on Computers, vol.48, no.10, pp.1083-1097, Oct 1999.

[11] Oklobdzija, V.G., "Algorithmic design of a hierarchical and modular leading zero detector circuit," Electronics Letters , vol.29, no.3, pp.283-284, 4 Feb. 1993.

[12] Oklobdzija, V.G., "An algorithmic and novel design of a leading zero detector circuit: comparison with logic synthesis" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.2, no.1, pp.124-128, March 1994.

[13] Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C., "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.57, no.4, pp.850-862, April 2010.

[14] Benkhelifa, E.; Pipe, A.; Dragffy, G.; Nibouche, M. , "Towards evolving fault tolerant biologically inspired hardware using evolutionary algorithms," 2007. CEC 2007. IEEE Congress on Evolutionary Computation, vol., no., pp.1548-1554, 25-28 Sept. 2007.

[15] Mazare, A.; Ionescu, L.; Serban, G.; Barbu, V., "Evolvable hardware with Boolean functions network implementation", 2011 International Conference on Evolutionary Computation, vol., no., pp.1-6, 7-8 Sept. 2011.

[16] Lohn, J.D.; Hornby, G.S., "Evolvable hardware: using evolutionary computation to design and optimize hardware systems," Computational Intelligence Magazine, IEEE, vol.1, no.1, pp. 19- 27, Feb. 2006.

[17] Forbes, N., "Evolution on a chip: evolvable hardware aims to optimize circuit design," Computing in Science & Engineering , vol.3, no.3, pp.6-10, May-June 2001.

[18] Stoica, A.; Zebulum, R.; Keymeulen, D.; Tawel, R.; Daud, T.; Thakoor, A., "Reconfigurable VLSI architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution-oriented chips", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.9, no.1, pp.227-232, Feb. 2001.

[19] Javier D. Bruguer and Toma´s Lang , "Leading - One Prediction with Concurrent Position Correction," IEEE transaction on Computers , vol. 48, no. 10, pp. 1083-1097, Oct. 1999.

Kunaraj Kumarasamy received the B.Tech. Degree in Electronics and Instrumentation Engineering from Pondicherry University and the M.E. Degree in Applied Electronics from Anna University in 2007. He is currently pursuing the Ph.D. degree in bio-inspired algorithms for design and optimization of digital circuits and filters. His current research interests include evolutionary algorithms for digital circuit synthesis, filter design and FPGA based system design.

Dr. R. Seshasayanan was born in the year 1958 in India and received his B.E degree from College of Engineering, M.E. degree from Anna University in the year 1980 and 1983 respectively. He received his Ph.D from Anna University. He is presently working as Associate Professor in the Department of Electronics and Communication Engineering, Anna University and his area of interests are Low Power VLSI Design and Reconfigurable Architectures for Image processing. He is actively involved in various R&D projects funded by NIOT and CVRDE.