Leading One Detectors: Evolutionary Approach

Kunaraj Kumarasamy; and **Seshasayanan Ramachandran**

ECE, CEG, Anna University, Chennai, 600025, India.
k.kunaraj@gmail.com

ID# (2677)
Rec.: 20/07/2012
In-revised: 02/08/2013
Corresponding Author;
Kunaraj Kumarasamy
E: mail: k.kunaraj@gmail.com

KEYWORDS

Leading one detector, Genetic, Algorithm, Evolvable Hardware, Floating point multiplier.

ABSTRACT

Leading One Detector (LOD) is an important and preliminary stage 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 the LOD. The LOD circuit is evolved using the evolutionary algorithm (EA) and using the evolved lower order LOD gate structure, various higher order LOD circuits are constructed using a hierarchical methodology. To obtain better results, the evolutionary algorithm is modified and a novel shuffling operation is performed to prevent the algorithm from settling in the local minima. The convergence and the robustness of the evolutionary algorithm is verified using standard test functions. The constructed LOD circuits are synthesized using Cadence® RTL Compiler® using TSMC 180nm library. The results obtained in terms of cell area and power of the elite LOD circuits show that the proposed evolved-architecture outperforms the existing circuits. The proposed architectures show a maximum of 31.18% improvements in Cell area and 31.27% in power for the 64-bitLOD circuit with an increase of 3.9% in the propagation delay.

منهجية تطوير الكاشفة وحيدة الاتجاه (LOD)

كوناراج كوماراسامي، سيشايانان راماشاندران
جامعة أننا، شيناي، الهند
بريد الكتروني:k.kunaraj@gmail.com

رقم المسودة: # (2677)
إستلام المسودة: 2012/07/20
إستلام المُعَدَلة: 2013/08/02
الباحث المُرَاسل: كوناراج كوماراسامي
بريد إلكتروني:
k.kunaraj@gmail.com

المُستلخص

تعتبر عملية تحويل العدد العشري إلى عدد بسيط هي المرحلة الأولي والأهم في منهجية تطوير الكاشفة وحيدة الاتجاه (LOD) والتي تشير إلى مُضاعفات الرقم العشري سواء كان بالزيادة أو النقصان (موجب أو سالب). أشار الباحث في هذه الدراسة إلى أنظمة ومنهجية متعددة تعتمد على ما يُعرف ببوابة (OR-IND)، والتي منها منهج دوائر التحويل اللوغراثيمي للرقم الصحيح إلى الرقم العشري (Logarithmic Converters) ومنهج الدوائر المُستعملة للطريقة الخوارزمية المُشابهة لنظرية تطور الجينات البشرية. هذا وقد تم في هذه الدراسة تصميم نظاماً مُصغراً وخاصاً من هذه الدوائر يمكن استخدامه كنظام فرعي يؤدي استخدامه للوصول إلى نتائج أفضل، ويتم ذلك بتعديل الطريقة الخوارزمية المتطورة والدمج بينهما (اللوغراثيمي والخوارزمية المتطورة) للوصول لمستوى نقطة الحد الأدنى. هذا وتستخدم هكذا المنهجية المُدمَجة على المستوى المحلي. عند استخدام منهجية الدمج يستوجب التحقق من مدى التقارب أو التوافق مع منهج استخدام الطريقة الخوارزمية المتطورة ويكون التحقق بإجراء الاختبارات المعتمدة. تم إنشاء هذه الدوائر وتجميعها باستخدام نظام (IRT,) 1800مم والذي هو عبارة عن مجموعة من المناهج والأنظمة الرقمية يمكن من خلالها اختيار أو تحديد النظام الانسب في شكل شرائح رقيقة مصنوعة من مادة السليكون. يستهدف هذا النظام أو المنهج الحصول على نسبة (31.18 %) تحسين وعلى نسبة (31.27 %) من الطاقة في الدوائر التي تعتمد بعملها على (64-bt) مع زيادة قدرتها للحصول على نسبة (3.9 %) في حالة النقصان.

الكلمات الدالة

كاشفة وحيدة الاتجاه، اللد، جيني، حلول حسابية، جهاز، نقطة الحياد

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 (Suzuki; *et,al,* 1969). A clear analysis of various application data shows that the signal processing algorithms require an average of 40% multiplication and 60% addition operations (Pappalardo; *et,al,*, 2004). 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. Leading One Detectors (LODs) and Leading One Position Detectors (LOPDs) determines the location of the most significant one or a leading bit in a given binary. The position of leading one is used for normalization process and shifting process in the floating point multiplication, floating point addition and also in binary logarithmic converters (Oklobdzija, 1993). 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 (Khalid; *et,al,*, 2006). Research has been going on to evolve various combinatorial circuits in a constrained space with minimum effort (Vesselin; *et,al,*, 2000). 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 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 2. Section 3 analyses the evolved gate structure of 4-bitLOD and the design of 8-bit, 16-bit, 32-bitand 64-bitLOD based on the evolved 4-bitLOD circuit. Section 4 compares the synthesize results and section 5 provides conclusion.

Material and Methods

(1) Genetic Algorithm (GA) for Evolving Leading One Detectors (LOD) circuits

The Genetic Algorithm (GA) is an adaptive search algorithm suitable for solving combinatorial optimization problems. We propose an evolutionary methodology involving Cartesian genetic programming (CGP) for evolving the LOD circuits as shown in figure 1. Perhaps the figure 1 shows the steps involved in evolving the LOD net list, synthesizing and creating the netlist database. 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.

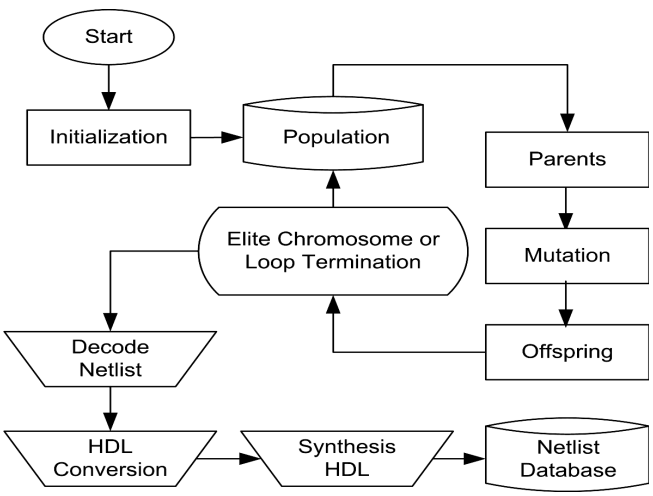


Figure 1: Genetic Algorithm (GA) for Evolving Leading One Detectors (LOD) Circuits

The fittest offspring can be taken as the parent for the subsequent iterations and after performing genetic operation to the parent, it yields fittest individual. Each individual is termed as 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 (Mahdiani; *et,al,* 2010) & (Benkhelifa; *et,al,* 2007) . In crossover, the parts of chromosome are exchange between parents and in mutation the bits of the parent chromosome are inverted to obtain the child. The resulting child chromosome resulting from the genetic operation can be taken as the parent chromosome for the next iteration. Multiple iterations are performed till we get the exact functionality of the circuit. The circuit evolution process is carried out till the algorithm yields an elite chromosome which matches the fitness function or the evolutionary process can be stopped after a predefined number of iterations (Lohn; and Hornby, 2006**);** (Benkhelifa; *et,al,*2007); (Mahdiani; *et,al,*2010) & (Mazare; *et,al,*2011). The evolved gate structure which implements the complete functionality is synthesized and stored in the net list database and the best circuit is chosen taking into consideration the number of logic levels and gates.

(1.1) Gate Array Structure

We propose to use a GA framework as shown in the Figure 2 built from an 8X8 array of logic cell 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 these array forms the chromosome. Thus a higher flexibility is given to the gate structure to increase the possibility of more gate structures for the given function.

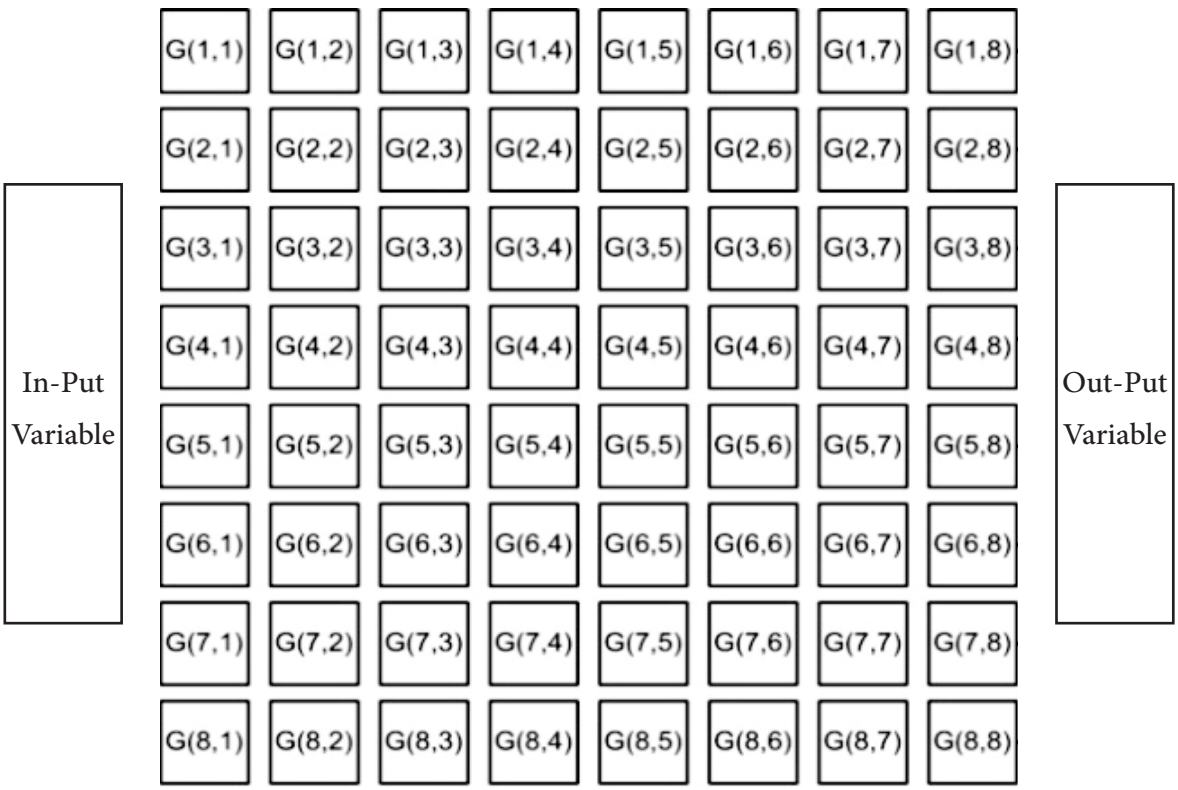


Figure 2: Logic Element of Dimensions 8X8 Array

(1.2) 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 struck in a local minima we shuffle the chromosome before passing the offspring to the next generation and this is called shuffling mechanism.

(1.3) Representation

This framework consists of fixed number of 64 logic elements arranged in an array structure of 8 X 8 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 8 X 8 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 combination of circuits for the same function. The Logic elements are referenced by position within the chromosome with circuit inputs are encoded in the first section of chromosome. The encoding ensures that the number of inputs and outputs described by a chromosome remains consistent after the genetic operation. Table 1 show the parameters used in the genetic algorithm.

Table 1: Parameter 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

(1.4) 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 of the each chromosome with the required fitness is measured. 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, (Mahdiani; *et,al,*2010), and for evolving LOD we represent the required fitness function as a fitness table shown in table 2.

Table 2: Representation of Fitness Function of 4-bitLOD

Input				Output				Zero Flag
X3	X2	X1	X0	Y3	Y2	Y1	Y0	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

(1.5) 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 Evolutionary algorithm evolved the 4-bitLOD with only a random-seeded initial population. Hence it is clear that a 100% functional combinational circuit can be evolved with random-seeded initial population.

(2) Evolved 4-bitand Higher Order LODs

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 used the multiplexers as the building block (Vesselin; *et,al,*, 2000). Replacing the standard 2:1 multiplexer using the basic logic gates will be beneficial if we can reduce the latency. In the proposed method, we evolve the architecture of the 4-bitLOD 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-bitLOD the higher order LODs of various sizes are be constructed as discussed in (Vesselin; *et,al,*, 2000). In Figure 3, the best evolved 4-bitLOD is shown.

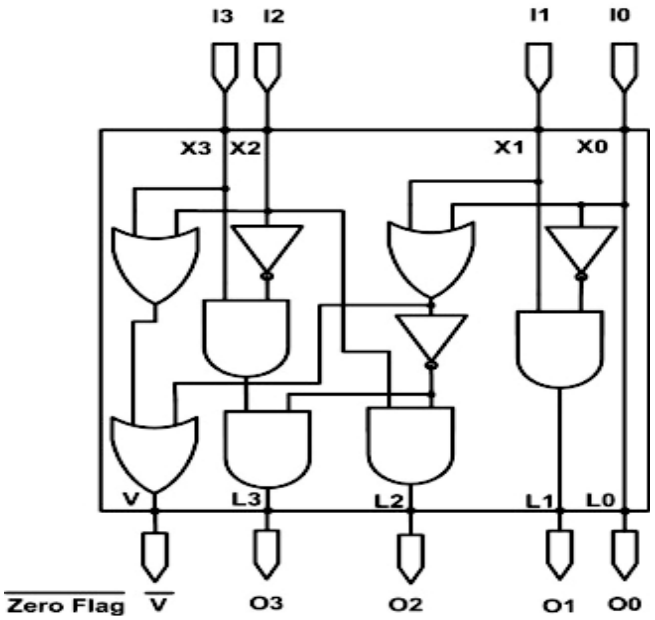


Figure 3: Evolved Four Bit LOD

(2.1) 8- Bit LOD

For constructing an 8-bitLOD from the evolved 4-bitLODs, we require two 4-bitLODs and eight 2-input AND gates in the output stage. The zero flag of the most significant 4-bitLOD 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-bitLOD circuit. Figure 4 shows the architecture of 8-bitLOD.

(2.2) 16-bitLOD

The architecture of 16-bitLOD is similar to the technique discussed in (Vesselin; *et,al,*, 2000). The 16-bitLOD is constructed using five 4-bitLODs and sixteen 2-input AND gates. The first stage has four 4-bitLODs and the 16 bit input is partitioned into four groups of 4 bit each which is 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-bitinputs given to each LOD. Figure 5 shows the architecture of 16-bitLOD circuit constructed using evolved 4-bitLOD 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 gate will carry the output of 4-input LOD in the first stage and the outputs of all other AND gates are zero. The replacement of multiplexers in the output stage with the AND gates reduces the overall propagation delay of the circuit.

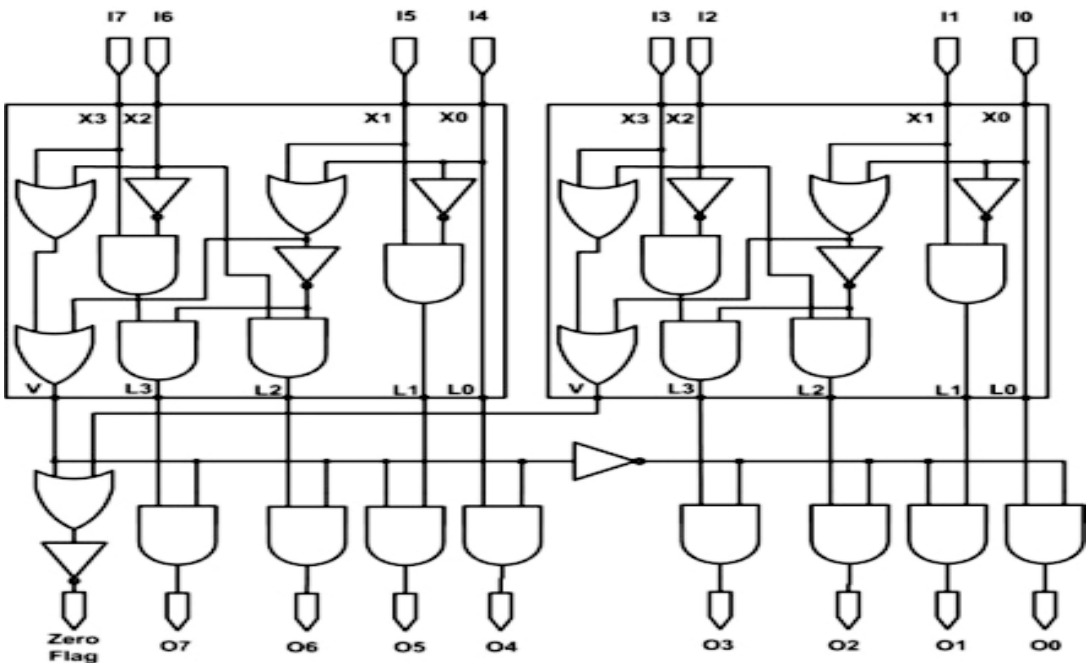


Figure 4: 8-bitLOD Constructed using the Evolved 4- Bit LOD

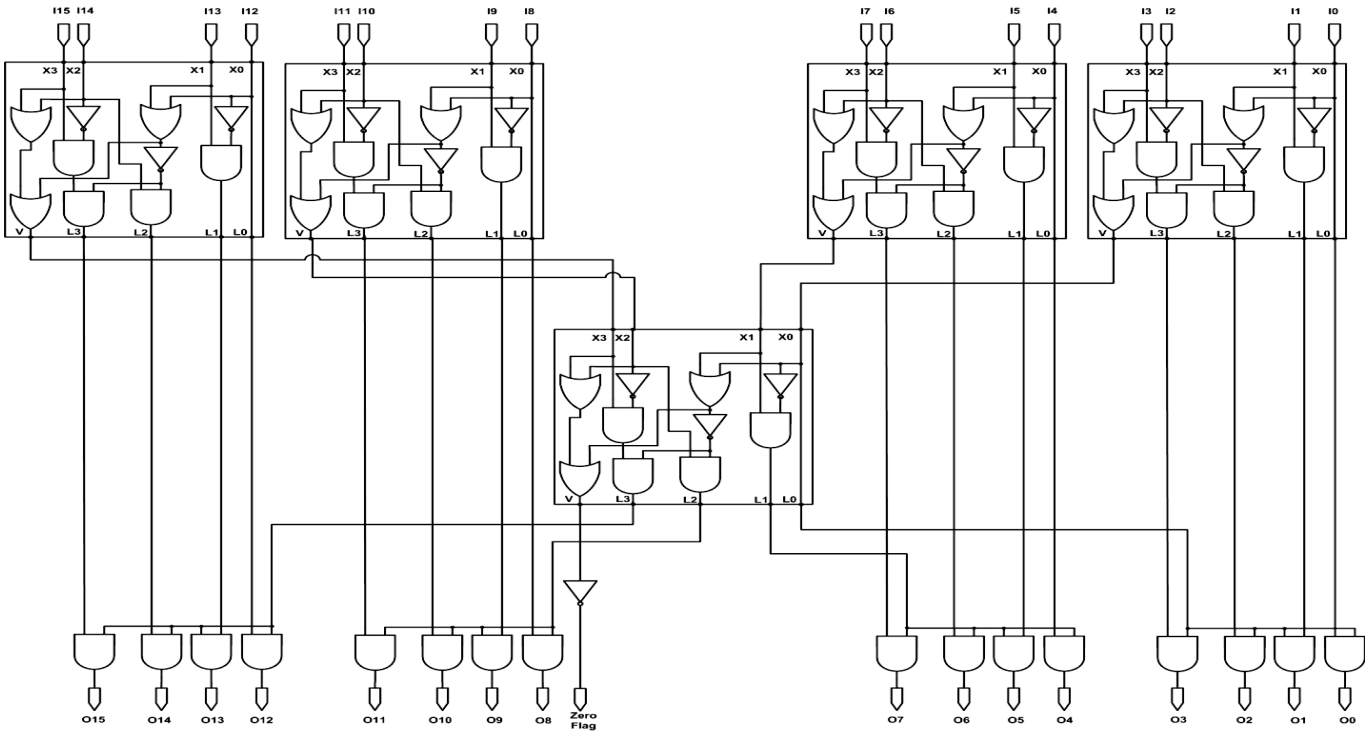


Figure 5: Architecture of 16-bitLOD

(2.3)Architecture of 32-bit and 64-bit LOD

The 32-bitLOD is constructed similar to 16-bitLOD and is shown in Figure 6.

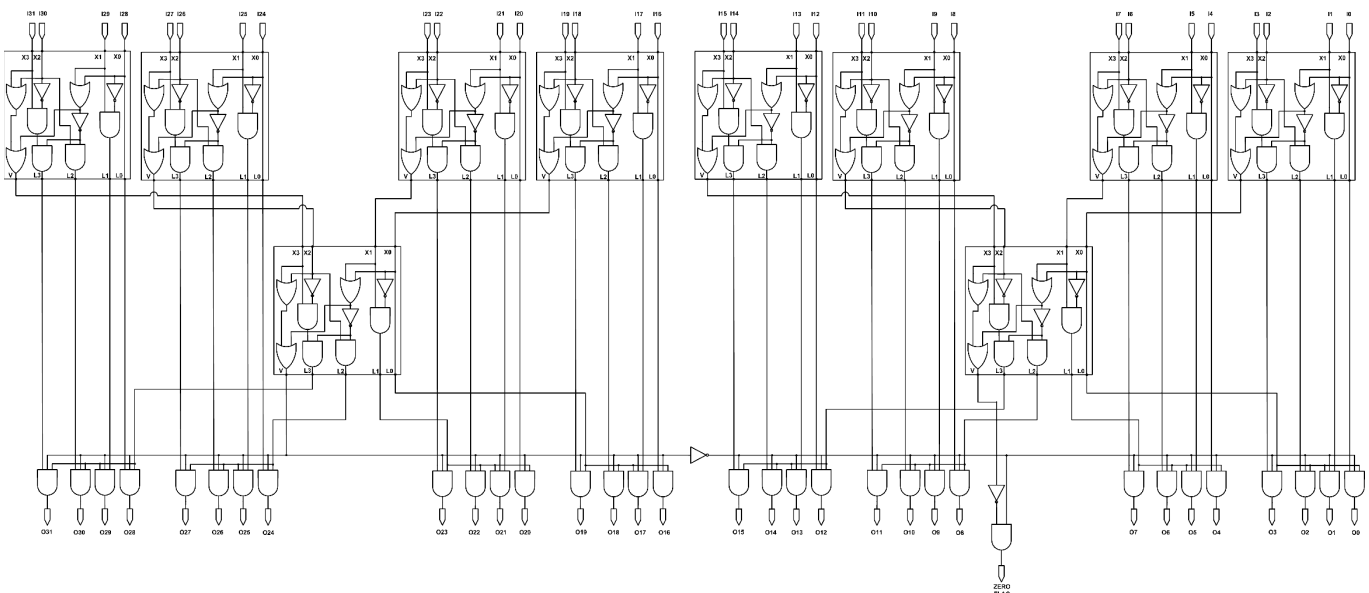


Figure 6: 32-bitLOD Constructed from 4-bitLOD

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-bitLOD in the second stage can be used to find the leading one in the most significant 16-bit. The third stage has series of 3-input AND gates similar to the last stage of 16-bitLOD discussed earlier. The zero flag from the 4-bitLOD in the second stage of the most significant 16-bitselects the corresponding output of the 4-bitLODs and all other outputs are made zero. A similar method followed for constructing 32-bitLOD is used to construct a 64-bitLOD circuit from 4-bitLOD circuit and is shown in Figure7.

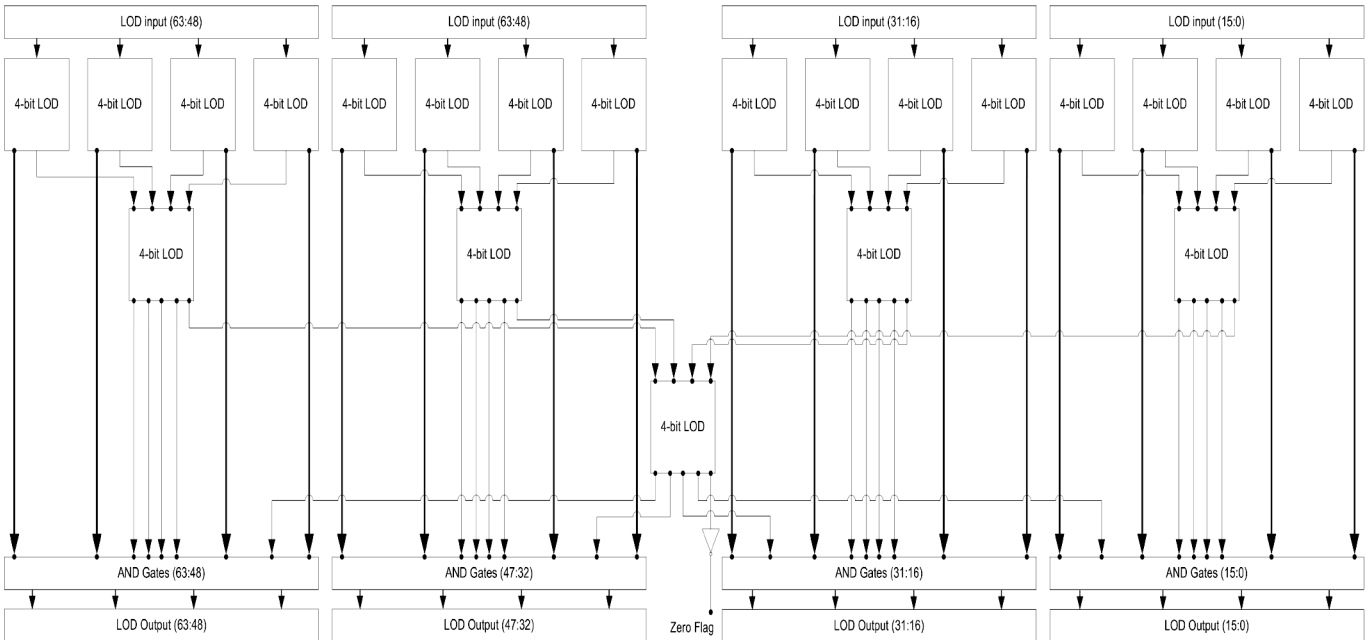


Figure 7: 64-bit LOD Constructed from 4-bit LOD

(3) Comparison of Synthesize Results

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 tabulated in Table 3.

The evolved gate level architecture shows promising enhancements in power area as depicted in table 4.

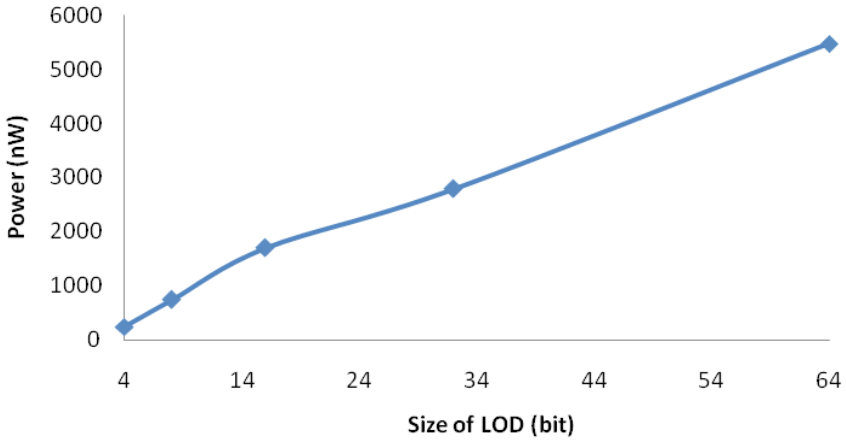
Table 3: Results of Various LOD Circuits

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

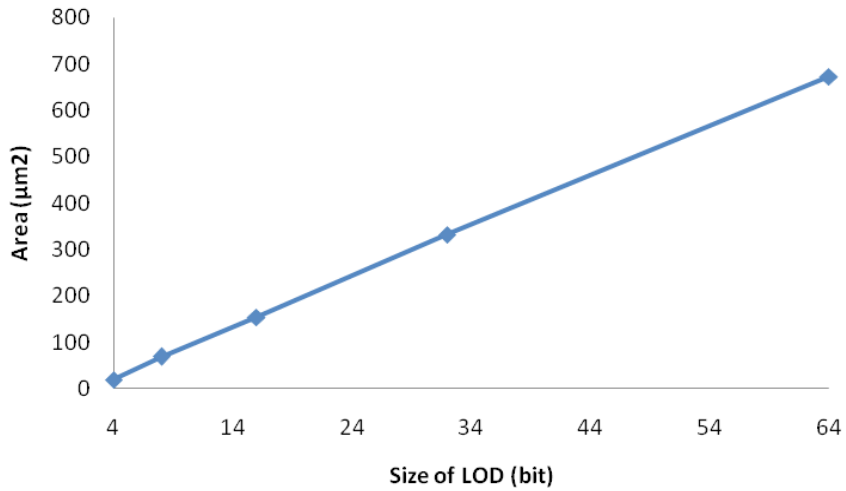
Table 4: The Synthesis Results Obtained for Various LOD Architectures of Various Sizes

Comparison of	**Size of the Circuit**	**(Oklobdzija, 1993).**	**(Dimitrakopoulos; *et,al,* 2008)**	**Proposed method(LOD)**
(a) Area for Various LOD Circuits (μm^2)	16-bit	213.087	178.642	153.821
	32-bit	465.996	389.971	331.632
	64-bit	978.133	791.021	673.142
(b) Power for Various LOD Circuits (nW)	16-bit	2363.02	1980.845	1706.154
	32-bit	4255.32	3391.66	2795.282
	64-bit	7956.52	6415.529	5468.402
(c) Comparison of Delay for Various LOD Circuits(ps)	16-bit	792	792	831
	32-bit	928	928	978
	64-bit	1183	1183	1303

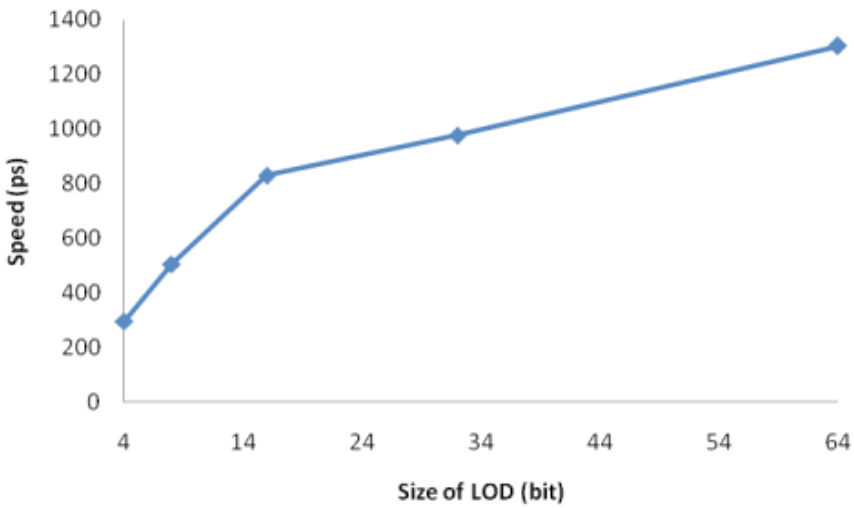
Figure 8 shows the plot of various synthesis results. From the Figure 8 (c) it is clear that the pace of increase in delay has been reduced because of the architectural changes and it is less pronounced as the size of the LOD circuit increases. Perhaps when compared to other existing architectures, the proposed LOD circuits have a slight increase in the propagation delay and a huge reduction in cell area and hence the power consumed as compared in table 4 (see, table 4).



(**Figure 8.a**) Plot of power for various sizes of LOD



(**Figure 8.b**) Plot of cell area for various sizes of LOD



(**Figure 8.c**) Plot of speed for various sizes of LOD

Figure 8: Plot of Various Synthesis Results

((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)*

We compare the cell area, power and the delay of various higher order LOD circuits constructed using lower order LOD. 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 synthesis results obtained for various LOD architectures of various sizes are shown in table 4. The evolved gate level architecture shows promising enhancements in power and area.

Conclusion

In this paper, we have discussed an evolutionary approach to the design of various LOD circuits. Though the higher order circuits cannot be completely evolved because of the limitation of the population size and search space, we have discussed few methods to construct higher order LODs from the evolved lower order circuits. Thus the gate level evolution of LOD circuits has proven performance benefits. As the design complexity increases, convergence of evolutionary algorithm to suit the fitness function will take more time and in some cases algorithm will settle to a local maxima and the shuffling mechanism introduced in the GA will answer this problem.

References

Benkhelifa E; Pipe A; Dragffy G; and **Nibouche M** (2007) Towards Evolving Fault Tolerant Biologically Inspired Hardware using Evolutionary Algorithms. *In: IEEE Congress on Evolutionary Computation, 25-28 Sept. 2007*. University of the West of England, Bristol, UK. pp1548-1554.
Available at: http:www.ieeexplore.org/xpl/

Dimitrakopoulos G; Galanopoulos K; Mavrokefalidis; Christos; and **Nikolos D** (2008) Low-Power Leading-Zero Counting and Anticipation Logic for High-Speed Floating Point Units. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **16** (7): 837-850.

Khalid H; Abed; and **Raymond Siferd E** (2006) VLSI Implementations of Low-Power Leading-One Detector Circuits *In: IEEE Southeast Conference, 31/ March- 2/April/ 2005*, Memphis, TN, pp 279 - 284.

Lohn JD; Hornby GS (2006) Evolvable Hardware: using Evolutionary Computation to Design and Optimize Hardware Systems. *Computational Intelligence Magazine, IEEE*, **1** (1): 19- 27.
Available at: http:www.ieeexplore.org/xpl/articleDetails.jsp? arnumber=1597058

Mahdiani HR; Ahmadi A; Fakhraie SM; and **Lucas C** (2010) Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. *IEEE Transactions on Circuits and Systems: Regular Paper*, **57** (4): 850-862.

Mazare A; Ionescu L; Serban G.; and **Barbu V** (2011) Evolvable Hardware with Boolean Functions Network Implementation". *In: International Conference on Applied Electronics, Published, 7-8 September 2011*. University of West Bohemia, Pilsen, Czech Republic. pp.1-6.

Oklobdzija (1993) An Algorithmic and Novel Design of a Leading Zero Detector Circuit: Comparison and Logic Synthesis". *IEEE Transactions on VLSI Systems*, **2** (1): 124-128.

Pappalardo F; Visalli G; and Scarana M; (2004) An application Oriented Analysis of Power/Precision Tradeoff in Fixed and Floating Point Arithmetic Units for VLSI Processors. *In: Proceedings of the Second IASTED International Conference on Circuits, Signals, and Systems, 28/Nov. - 1/Dec., 2004, Clearwater Beach,* FL, IASTED/ACTA Press, USA. pp416-421.

Suzuki H; Morinaka H; Makino H; Nakase Y; Mashiko K; and **Sumi T** (1969) Leading-Zero Anticipatory Logic for High Speed Floating Point Addition, *IEEE Journal on Solid-State Circuits,* **31** (8) 1157-1164.
Available at: http:www.ieeexplore.org/xpl/login.jsp?tp=&arnumber=508263&url=

Vesselin K; Vassilev; Julian F; Miller; Terence Fogarty C (2000) Towards the Automatic Design of More Efficient Digital Circuits. *In: Proceedings of the Second NASA/DOD Workshop on Evolvable Hardware (EH '00) 13 -15/ July/ 2000*, Palo Alto, CA. pp151 – 160.