Middle-East Journal of Scientific Research 23 (6): 1181-1186, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.06.22137

Synthesis of Adder Circuit Using Cartesian Genetic Programming

S. Asha and R. Rani Hemamalini

Department of Electronics and Communication Engineering,
St. Peter's University, Avadi, Chennai, India

Abstract: Digital adders form a significant part of the arithmetic unit in the processors. Many Digital Signal Processing (DSP) algorithms equally uses adder and multiplier element as its component to achieve the required arithmetic operation. Hence it is important to optimize the adder circuit in the gate-level itself to design it for the required standards. Recently there are various bio-inspired optimization algorithms which efficiently synthesize digital circuits like adders and multipliers. Optimization algorithms like genetic Algorithm (GA), Particle swarm optimization (PSO) and Harmony Search (HS) has proved its efficiency in various optimization problems. We utilize the conventional Cartesian genetic programming (CGP) along with the shuffling mechanism to evolve the 4X4 adder circuit using only two input NAND gate library. The evolved adder circuit is compared with the existing adder circuits to prove its performance benefits. This evolved 4-bit adder is used further to synthesis higher order adders for its real time performance benefits.

Key words: Evolved adder · Cartesian Genetic Programming · Partitioned Multiplier · Bio-Inspired Computation · Genetic Algorithm · Optimization of Digital circuits

INTRODUCTION

There are many Boolean simplification methods like Karnaugh Maps and Quine Mc-Cluskey's tabulation method which are suitable for manual simplification and may not be efficient to implement in a computer as a computer-aided design (CAD) tool. Also commercially available CAD tools are efficient in optimizing the digital circuits when compared to conventional techniques. Recently bio-inspired algorithms used for optimizing digital combinational circuits show good results in terms of efficient optimization of the digital circuits based on user demand. Evolving a huge combinational circuit with more number of logic gates is a complex task to achieve for any optimization algorithms and still scalability is an unresolved issue [1].

A 2-bit multiplier and 4-bit odd parity generator circuits have been evolved using a new technique for the synthesis of combinational circuits by using CGP and uniform NAND gate based templates. Many researchers concentrated on evolving the circuit layout after confirming the evolution of functionality of the multiplier circuit. The dynamic power consumed by the adder and multiplier can be reduced by minimizing the switching activity of the partial products in 2's complement multiplier [3-5]. The operand decomposition algorithm proposed in [6] reduces the logic transition of array multiplier and tree multiplier and hence the dynamic power can be reduced at the expense of little more gates.

Cartesian Genetic Programming (CGP)

Conventional CGP: CGP was developed by Miller *et al.* and it is widely used for evolving digital circuits [9-11].

CGP allows the re-use of nodes and any node in the directed graph can be reused and the output of any node serves as the input of the node in the subsequent stage. The length of the genotype in CGP is fixed, but it does not mean that all the nodes in the graph should be used. The phenotype will be of variable length which means the circuit can change in size and levels and is limited by the maximum number of nodes in CGP. There may be many unused nodes or genes which will not have any impact on the circuit operation or the fitness function (neutrality) and those nodes can be avoided when mapping the genotype to phenotype [2-11].

Corresponding Atuhor: S. Asha, Department of Electronics and Communication Engineering, St. Peter's University, Avadi, Chennai, India.

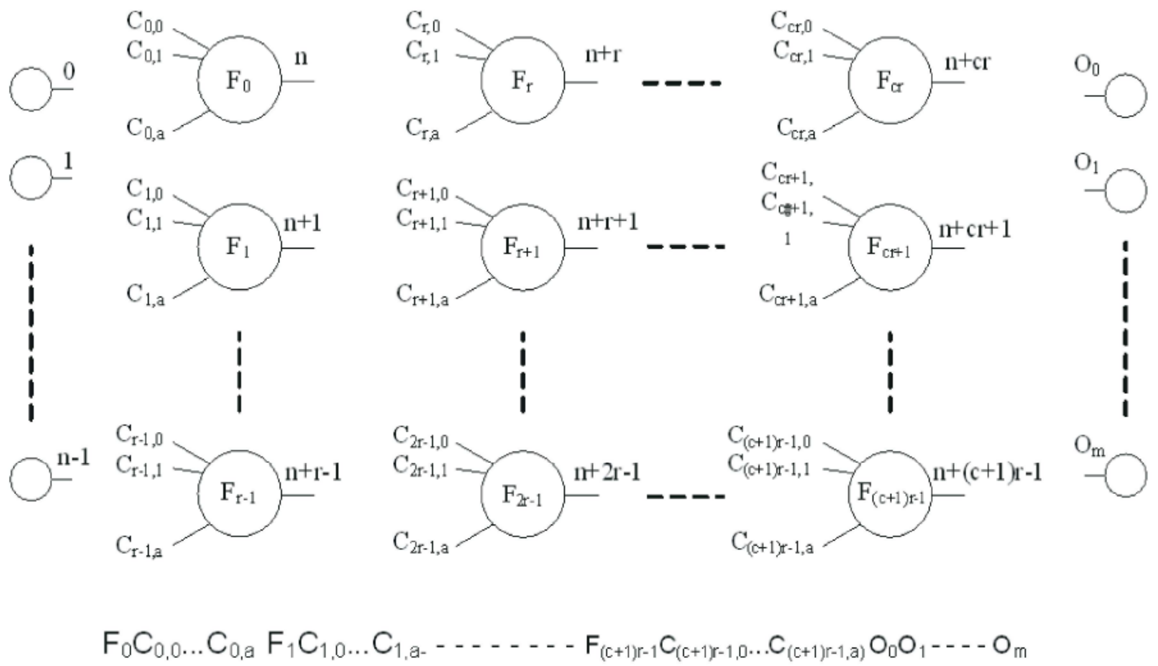

F0C0,0...C0,a F1C1,0...C1,a- - - - - - - - - F(c+1)r-1C(c+1)r-1,0...C(c+1)r-1,a) O0O1 - - - - Om

Fig. 1: Node Structure of CGP

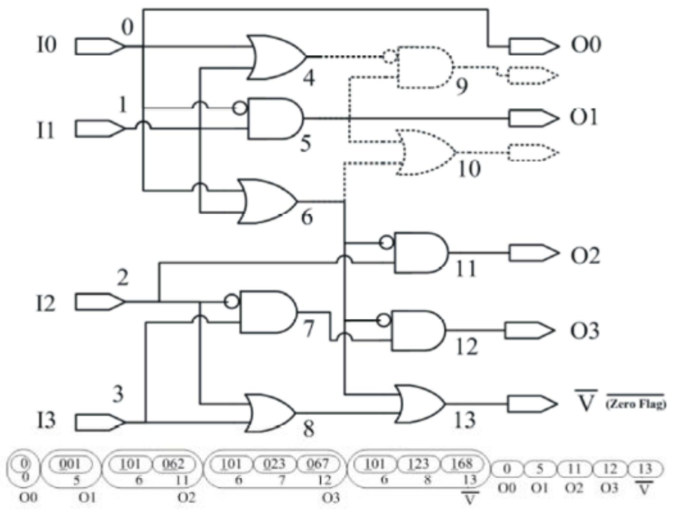
CGP and its Variants: CGP fails to solve the problems with multiple outputs and the computational effort drastically raises when the problem size and the number of input and output increases. The multi-chromosome CGP (MC-CGP) proposed in [12] has a considerable improvement in the performance and hence can arrive at a solution even for problem with multiple inputs and multiple outputs. But the solutions arrived by MC-CGP are much larger and possesses huge number of nodes which may not be suitable for evolving area efficient digital circuits such as multipliers [12]. Hence we modified the conventional CGP to optimize both the run time (i.e.) computational effort and also to find the optimal solution by changing the fitness into hierarchical fitness function.

There are several variations to the existing conventional CGP. Iterative self-modifications are done to allow the phenotype changes and they are termed as self-modifying CGP [13, 14]. For modular problems the re-use of subroutines in CPG has shown significant improvement which is called embedded CGP or modular CGP. To reduce the complexity of a bigger or multiple output problems, it is decomposed into smaller problems and is made easier to find a solution. This technique is called multi-chromosome CGP. In Real-valued CGP a cross over operator is introduced to improve the evolution process and extra level of neutrality are added and real-valued genotype is introduced [15]. The recombination is improved using Implicit-context CGP which introduces implicit context representation to CGP and hence enables positional independence [13]. The shuffling mechanism introduced in the conventional CGP avoids the algorithm from settling in the local maximum and tends to find an optimal solution rather than sub-optimal solutions [2].

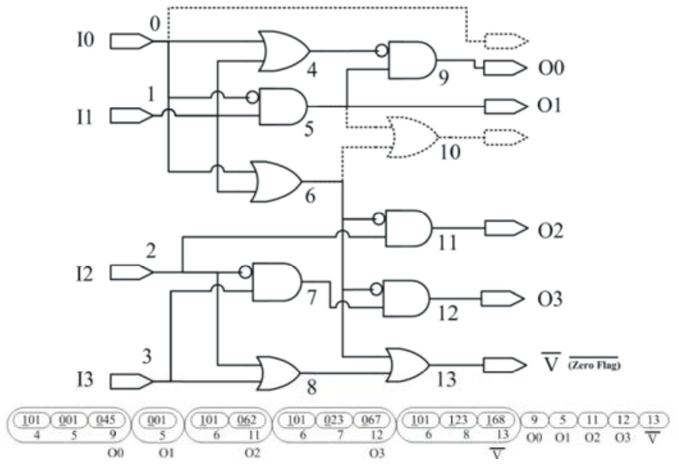
Selection and Fitness in CGP: In the CGP, the parent is the best in the total population in terms of functionality and the gate count. We slightly modify the selection mechanism of the standard CGP to cater the needs of evolving adders based on NAND gate template alone. If the chromosome has satisfied the functionality and if the gate count of the chromosome is lesser than its predecessor, then it is selected as the parent for the next run.

Mutation in CGP: The mutation operator used in CGP is typically a point-mutation operator similar to CGP, in which a number of randomly chosen genes in the genotype are changed to other valid randomly chosen values.

Fig. 2a represents an evolved 4-bit digital circuit which has 4 inputs and 5 outputs and the multi-chromosome representation of the evolved circuit are shown in the same figure. There are totally five chromosome nodes O0 t0 O3 and V'. The value of chromosome node O1 is 001, where 0 denotes the "one



a) Evolved 4-bit circuit



b) Point mutated output node (O0) shifts from 0 to 9

Fig. 2: An example of point-mutation operator

input bubbled-AND" gate and 1 represents 2-input OR gate. Here 0 and 1 represents the input nodes I0 and I1. Similarly all other chromosomes can be decoded. In Fig. 2b the output O0 is mutated from the value 0 to 9 and hence the node 9 becomes active which includes an additional gate.

Fitness in CCGP: Fitness function of the adder is similar to the truth table of the 4-bit adder itself. As we are trying to evolve the precise 4-bit adder, the circuit should satisfy all the input possibilities (24) otherwise the evolved circuit cannot be considered.

Synthesis and Optimization of Adder Circuit: The combinational circuit can be optimized by the minimization of the netlist with respect to logic cell area, power dissipation and the propagation delay for the corresponding technology library used. The optimization algorithm in the EDA tools in general follows the following steps to optimize any given logic.

- Flattening
- Logic minimization
- Timing-driven factorization
- Technology mapping

Also the degree of optimization depends on the logic depth of the circuit which is taken into consideration and particularly the arithmetic circuits are challenging to perform the optimization process. Perhaps the efficient synthesis and realization of these arithmetic circuits like adders and multipliers relies on datapath synthesis and also efficient logic optimization at the very fundamental level. Thus the synthesis and optimization of 4-bit adder is very important as it contributes to the datapath of the arithmetic circuit.

RESULTS

The 4-bit adder circuit is evolved using the CGP along with the shuffling mechanism which tends to avoid local maxima. The evolved circuit gene is then decoded to obtain the gate equivalent phenotype. Also the phenotype is decoded using Verilog Hardware Descriptive Language (HDL) to make it suitable for synthesis and implementation in any commercially available Electronic Design Automation (EDA) tool.

Synthesized Netlist of the Evolved 4-Bit Adder: The decoded netlist is given as input to the standard synthesis tool to check its functionality and performance. We use Altera© Quartus II© to synthesis the gate-level decoded architecture and from the synthesized circuit, the logic area and the time delay consumed by the circuit can be estimated. Fig. 3 shows the synthesized RTL structure of the evolved 4-bit adder [10].

Construction of Higher Order Adders from the Evolved 4-Bit Adder: The higher order adders are constructed from the evolved lower order 4-bit adder. A hierarchical tree approach is considered when higher level adder circuits are constructed. As evolving 16-bit and higher order adders are practically limited because of the scalability of the CGP, it is worthwhile to evolve lower order adders from which higher order adders can be constructed. Construction of higher order adders from the evolved 4-bit adder increases the overall propagation delay if it constructed as a ripple adder. Hence we use separate circuit to compute the carry so that the excessive increase in the delay can be reduced at the expense of extra gates and a small increase in the logic cell area. Table 1 shows the comparison of gate count the additional hardware resources such as multiplexers used for various adders. Table 2 compares the cell area, delay and the power consumed for the listed adders [11].

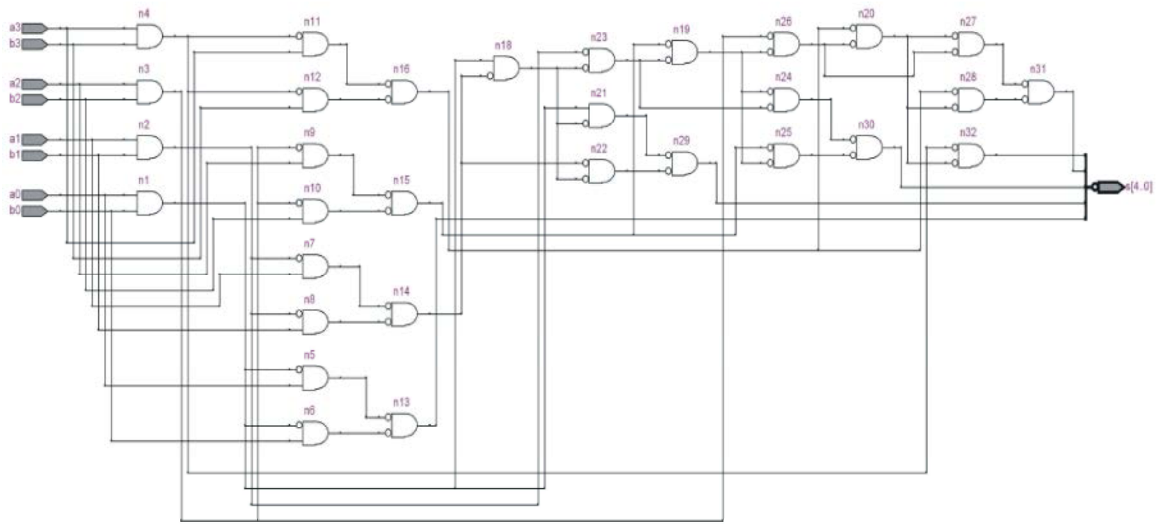
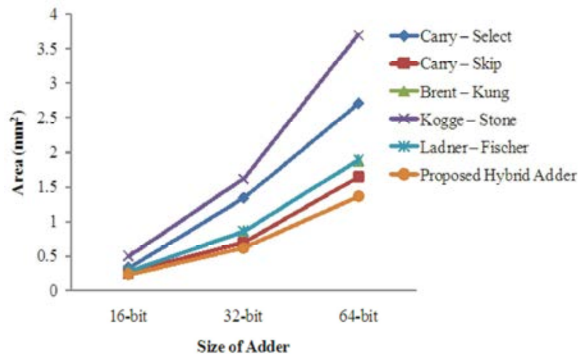


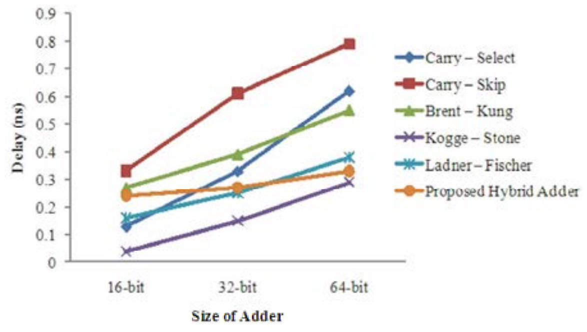
Fig. 3: Synthesised netlist showing the evolved 4-bit adder

Table 1: Comparison of hardware resources of various adders

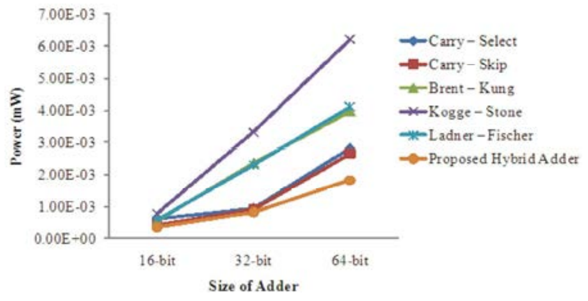
	16 bits		32 bits		64 bits	
Type of Adder	No. of Gate	No. of Mux	No. of Gate	No. of Mux	No. of Gate	No. of Mux
Carry – Select	160	3	320	7	640	15
Carry – Skip	86	3	170	5	329	10
Brent – Kung	123	0	254	0	510	0
Kogge – Stone	179	0	449	0	1089	0
Ladner – Fischer	129	0	305	0	705	0
Proposed Hybrid Adder	124	0	253	0	511	0



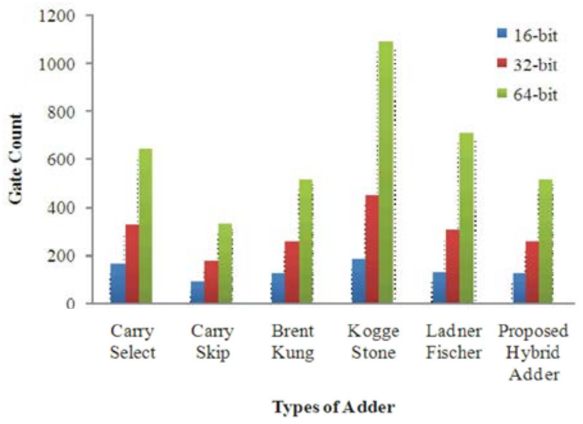
(a) Cell Area



(b) Delay



(c) Power



(d) Gate Count

Fig. 4: Comparison of (a) Cell Area, (b) Delay, (c) Power, (d) Gate Count

Table 2: Comparison of synthesis results of various adders

(a) 16-bit

	16 bits		
Type of Adder	Area (mm^2)	Delay (ns)	Power (mW)
Carry – Select	0.327	0.13	6.04E-04
Carry – Skip	0.259	0.33	4.27E-04
Brent – Kung	0.276	0.27	5.63E-04
Kogge – Stone	0.496	0.04	7.78E-04
Ladner – Fischer	0.276	0.16	5.81E-04
Proposed Hybrid Adder	0.232	0.24	3.62E-04

(b) 32-bit

	32 bits		
Type of Adder	Area (mm^2)	Area (mm^2)	Area (mm^2)
Carry – Select	1.3367	0.33	9.47E-04
Carry – Skip	0.6914	0.61	9.05E-04
Brent – Kung	0.8551	0.39	2.35E-03
Kogge – Stone	1.6146	0.15	3.31E-03
Ladner – Fischer	0.8583	0.25	2.30E-03
Proposed Hybrid Adder	0.6126	0.27	8.21E-04

(c) 64-bit

	64 bits		
Type of Adder	Area (mm^2)	Area (mm^2)	Area (mm^2)
Carry – Select	2.707	0.62	2.81E-03
Carry – Skip	1.6432	0.79	2.64E-03
Brent – Kung	1.8843	0.55	3.97E-03
Kogge – Stone	3.701	0.29	6.21E-03
Ladner – Fischer	1.8871	0.38	4.11E-03
Proposed Hybrid Adder	1.3621	0.33	1.82E-03

The graphical representation of the comparison of logic cell area, delay, power and gate count is depicted in Fig. 4.

CONCLUSION

Thus we have evolved a NAND gate based 4-bit adder by employing the CGP along with the shuffling operator to obtain global optimal solution. The higher order adders like 16-bit, 32-bit and 64-bit adders are constructed using the lower order evolved 4-bit adder using a hierarchical approach. As the optimization algorithm cannot evolve huge multipliers, this hybrid approach is suitable as it exploits the benefits of both the conventional design method and the robust optimization algorithm. The synthesized adder proves performance benefits in terms on logic cell area and hence the power consumed. This hybrid approach can be extended to other combinational circuits to synthesize a real time circuit. This adder can be used in an application to prove its suitability in real time hardware implementation.

REFERENCES

1. Vassilev, V.K. and J.E. Miller, 2000. Scalability problems of digital circuit evolution evolvability and efficient designs, Proceedings. The Second NASA/ DoD Workshop on Evolvable Hardware, pp: 55-64.
2. Kumarasamy Kunaraj and Seshasayanan Ramachandran, 2013. Leading One Detectors: Evolutionary Approach, Arabian Gulf Journal of Scientific Research, 32(2/3): 145-153.
3. Vassilev, V.K., D. Job and J.F. Miller, 2000. Towards the automatic design of more efficient digital circuits, Proceedings, The Second NASA/DoD Workshop on Evolvable Hardware, pp: 151-160.
4. Nan-Ying Shen and O.T.C. Chen, 2002. Low-power multipliers by minimizing switching activities of partial products, ISCAS 2002. IEEE International Symposium on Circuits and Systems, 4: 93-96.
5. Chen, O.T.C., Sandy Wang and Yi-Wen Wu, 2003. Minimization of switching activities of partial products for designing low-power multipliers, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 11(3): 418-433.
6. Ito, M., D. Chinnery and K. Keutzer, 2003. Low power multiplication algorithm for switching activity reduction through operand decomposition, Proceedings. 21st International Conference on Computer Design, 21- 26: 13-15.
7. Irfan, M., Q. Habib, G.M. Hassan, K.M. Yahya and S. Hayat, 2010. Combinational digital circuit synthesis using Cartesian Genetic Programming from a NAND gate template, 6th International Conference on Emerging Technologies (ICET), 343-347: 18-19.
8. Kalganova, T. And J. Miller, 1999. Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness,Proceedings of the First NASA/DoD Workshop on Evolvable Hardware, pp: 54-63.
9. Miller, J.F., D. Job and V.K. Vassilev, Principles in the Evolutionary Design of Digital Circuits - Part l," Genetic Programming and Evolvable Machines, 1(1): 8-35.
10. Vassilev, V.K., D.Job and J.F. Miller, 2000. Towards the automatic design of more efficient digital circuits, Evolvable Hardware, 2000. Proceedings, The Second NASA/DoD Workshop on Evolvable Hardware, pp: 151-160.
11. James Alfred Walker and Julian Francis Miller, 2005. Improving the evolvability of digital multipliers using embedded cartesian genetic programming and product reduction, In Proceedings of the 6th international conference on Evolvable Systems: from Biology to Hardware (ICES'05).
12. Walker, Miller, Parallel evolution using multi-chromosome Cartesian genetic programming, Genet Program Evolvable Mach.
13. Lones, M. and S.L. Smith, 2010. Objective Assessment of Visuo-spatial Ability using Implicit Context Representation Cartesian Genetic Programming, Genetic and Evolutionary Computation: Medical Applications, Wiley.
14. Harding, S., J.F. Miller and W. Banzhaf, 2009. Self modifying Cartesian Genetic Programming: Parity, Evolutionary Computation, 2009. CEC '09. IEEE Congress on, pp: 285-292.
15. Walker James Alfred and Julian Francis Miller. 2007. Solving real-valued optimisation problems using cartesian genetic programming, In Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07). ACM, New York, NY, USA, pp: 1724-1730.