Cost Effective Implementation of Fixed Point Adders for LUT based FPGAs using Technology Dependent Optimizations

Burhan Khurshid and Roohie Naaz

***Abstract*—Modern day field programmable gate arrays (FPGAs) have very huge and versatile logic resources resulting in the migration of their application domain from prototype designing to low and medium volume production designing. Unfortunately most of the work pertaining to FPGA implementations does not focus on the technology dependent optimizations that can implement a desired functionality with reduced cost. In this paper we consider the mapping of simple ripple carry fixed-point adders (RCA) on look-up table (LUT) based FPGAs. The objective is to transform the given RCA Boolean network into an optimized circuit netlist that can implement the desired functionality with minimum cost. We particularly focus on 6-input LUTs that are inherent in all the modern day FPGAs. Technology dependent optimizations are carried out to utilize this FPGA primitive efficiently and the result is compared against various adder designs. The implementation targets the XC5VLX30-3FF324 device from Xilinx Virtex-5 FPGA family. The cost of the circuit is expressed in terms of the resources utilized, critical path delay and the amount of on-chip power dissipated. Our implementation results show a reduction in resources usage by at least 50%; increase in speed by at least 10% and reduction in dynamic power dissipation by at least 30%. All this is achieved without any technology independent (architectural) modification.**

***Index Terms*— FPGA, LUT, FPGA primitives, Technology mapping, Boolean Network.**

Original Research Paper
DOI: 10.7251/ELS1519014K

I. INTRODUCTION

FIELD programmable gate arrays provide an alternative approach to application specific integrated circuits (ASIC) implementation [1] with features like large-scale integration, design verification post production, lower non-recurring engineering (NRE) costs, reconfigurable design approach etc. [2, 3]. FPGAs also offer an attractive platform for development of novel systems for rapid system prototyping and low to medium volume productions [1, 4, 5, 6]. Most of the modern day FPGA devices contain programmable logic blocks that have look-up table (LUT) as the basic programmable logic element [7, 8]. A k-input LUT is a digital memory that can implement any Boolean function of k variables. The k inputs in an LUT address 2k storage elements that store the truth table of the Boolean function. LUT based FPGAs account for a significant part of the commercial FPGA market [9, 10].

Since their genesis in 1985 [10], FPGAs have evolved enormously with state-of-art devices having in-built full custom processing elements like multipliers, DSP blocks, fast carry chains, high speed clocking, I/O resources etc. [11, 12, 13]. These blocks are highly optimized in terms of speed or area thereby facilitating efficient realization of complex functions [14, 15]. One of the major changes in the FPGA architecture has been the introduction of 6-input LUT as a logic element [11, 16]. With this FPGA primitive, the logic implementation would lead to higher logic densities resulting in a minimal-depth circuit and hence higher speed - a trend towards which the current FPGAs are oriented [17, 18, 19].

Perhaps the biggest issue with 6-input LUTs is their under-utilization while implementing a particular logic function, since many logic functions do not require six inputs [10]. This leads to low logic density and thus slower structures. Although many FPGA vendors have designed these elements with dual output capabilities [17, 18], their usage in implementing a Boolean function still remains far from the optimum.

Another issue is regarding the technology mapping of Boolean networks representing a combinational function. Logic synthesis in FPGAs has a well-defined flow that starts with design entry and proceeds through phases like synthesis, translation, mapping and place and route (PAR). Technology mapping is one of the most important phases in the FPGA computer aided design (CAD) flow that is directly concerned with selecting the circuit elements that will implement a given Boolean network [1, 7]. For LUT based FPGAs the target circuit element is the *k*-input LUT. Technology mapping is always cost driven. The goal is to produce a minimum-cost circuit that implements a desired Boolean function [1, 7, 8]. The cost of the circuit is typically a measure of its area, speed, power or a combination of these and accordingly there are algorithms that drive the technology mapping process towards area optimization [19, 20, 21, 22, 23, 24], delay optimization

Manuscript received 30 October 2014. Received in revised form 27 March 2015. Accepted for publication 10 May 2015.

Burhan Khurshid is with the Department of Computer Science and Engineering, NIT, Srinagar, India (phone: +91-9797875163; e-mail: burhan_07phd12@nitsri.net).

Roohie Naaz is with Faculty of Computer Science and Engineering, NIT, Srinagar, India (e-mail: naaz310@nitsri.net).

[25, 26, 27, 28, 29, 30], power optimization [31, 32, 33, 34, 35, 36], or area and delay optimization [37, 38, 39, 40, 41].

Modern day commercially available tools from leading vendors like Xilinx and Altera have fully automated technology mapping. The technology mappers in these tools take the native generic circuit (NGC) file from synthesis and create a native circuit description (NCD) file as per the desired cost function. The NCD file contains the physical description of design in terms of the components in the target device. Thus the mapping process is fully automated and the designer has no control over the selection of circuit elements that will implement the given function.

In this paper, we aim at tackling these two issues. The contributions of this paper are:

i) We re-design the initial Boolean network for RCA based adders for area, speed, and power optimality. Achieving area and power optimality for Boolean networks with inputs greater than four is NP-hard [31, 42, 43], however, since the basic cell in RCA based fixed point adders is very simple our approach simultaneously addresses the area, speed and power issues and a complete utilization of 6-input LUTs is assured. We do not particularly use any of the algorithms listed previously but our approach uses a combination of techniques like node decomposition, exploiting re-convergent nodes, logic replication etc.

ii) Since design entry is the only manual phase in the FPGA design flow, we try to control the mapping of the Boolean networks at the design entry step only. This involves modifying the coding style and writing VHDL codes for optimized Boolean networks based on direct instantiations of the targeted circuit elements. This is in contrast to the conventional coding styles that are typically behavioral and rely completely on the synthesizer to map the Boolean network by inferring the logic.

We have compared our implementation results against various adder designs listed in [44]. Our implementation results show a reduction in resources usage by at least 50%; increase in speed by at least 10% and reduction in dynamic power dissipation by at least 30%. We have also compared our implementation against the Xilinx IP adder v 11.0 and a subsequent improvement in performance is observed.

The rest of the paper is organized as follows. Section II discusses some basic terminology used in this paper. Section III discusses the basic technology mapping of the Boolean network corresponding to the RCA cell on LUTs. In section IV we redesign the initial Boolean network to ensure proper utilization of the 6-input LUT. Synthesis and implementation is carried out in section V. Conclusions are drawn in section VI. References are listed at the end.

II. Definitions and Terminology

A *Boolean network* is a directed acyclic graph (DAG) with *nodes* corresponding to logic gates, primary inputs and primary outputs and directed edges corresponding to wires connecting the gates. Since the Boolean networks considered in this paper are simple full-adder circuits, we will use actual gates for nodes. Further the term *network* will be used to refer to a Boolean network representing a combinational function and the term *circuit* will be used to refer to a Boolean network representing a circuit net-list i.e. a network of connected LUTs.

A node in a network may be driven by zero or more predecessor nodes known as *fan-in* nodes and may drive zero or more successor nodes known as *fan-out* nodes. The primary inputs (PIs) of the network are nodes without any fan-in. Similarly primary outputs (POs) are nodes without any fan-out. A network is said to be *k-bounded* if the number of fan-ins of each node does not exceed *k*.

The *level* of a node is the length of the longest path from any PI to the node. The node itself is counted in the path length. In this paper we have considered buffered inputs and outputs so that PIs and POs also contribute to the network *depth* which is defined as the largest level of a node in the network. The delay and area of a mapped circuit is measured by the depth and number of LUTs respectively.

A *cone* of a node v, C_v, is a sub-graph consisting of the node v and some of its non-PI predecessors, such that any node in this cone has a path to the node v that lies entirely in C_v. Node v is referred to as the *root* of the cone.

III. Mapping the Basic RCA Cell

Addition is one of the basic operations in digital signal processing (DSP) systems. It is used as a primitive operation in various arithmetic circuits like multipliers, multiply-adders etc. In order to maximize the performance of the adder circuit various technology independent (architectural) approaches have been used. However, this work focuses on carrying out the technology dependent optimization of the conventional ripple carry adder on LUT based FPGAs.

Technology mapping using LUTs is a two step process. In the first step, the entire network is partitioned into suitable sub-networks. The individual nodes within each sub-network are then covered with suitable cones. The logic implemented by each cone is then mapped onto a separate LUT and an optimal LUT net-list is obtained. In the second step, the net-list for the entire network is constructed by assembling the individual net-lists. The overall goal is to have a circuit implementation that uses minimum possible LUTs and has minimum possible depth.

The basic cell in an RCA network is a full adder. Fig. 1 shows the Boolean network for a full adder circuit. The network is partitioned into two sub-networks corresponding to sum (S) and carry (C), by dividing it at fan-out nodes. Each sub-network is separately mapped into a circuit of LUTs by covering the individual nodes with suitable cones. A straight forward approach would be to cover each node within a sub-network with a separate cone. The sub-network is then traversed in post-order depth-first fashion and each cone is assigned to a separate LUT as shown in Fig. 2(a). The number at the lower-right corner of each LUT indicates the level of the LUT assuming each LUT has a delay of one unit. The overall depth of the circuit at PO nodes S and C is four (including the buffers at PIs and POs). The total number of LUTs needed is six. The number of LUTs may be reduced by decomposing the 3-input OR gate in the carry sub-network. The decomposed node is included in two separate cones and the sub-network is

again traversed in post-order depth-first fashion to have a circuit implementation of Fig. 2(b). The number of LUTs is now reduced to three. However, an optimal implementation may be obtained by exploiting the reconvergent PI nodes in the carry circuit. Reconvergent nodes share the same inputs and can be exploited to reduce the number of PIs to a sub-network. This is shown in Fig. 2(c) where the reconvergent paths are included within the LUTs and the circuit is implemented with a single LUT for each sum and carry sub-network. The number of LUTs utilized is two and the overall depth including the buffers at PIs and POs is three. An n-bit adder implemented using the optimized circuit of Fig. 2(c) will have an overall depth of n+2 and will require 2n LUTs.

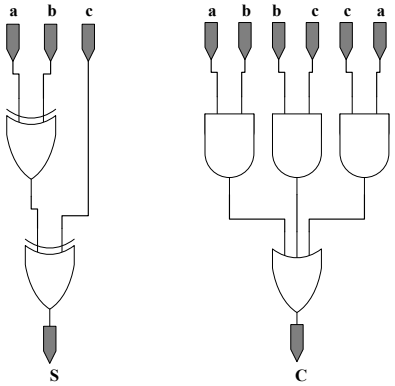

Fig. 1. Basic RCA cell.

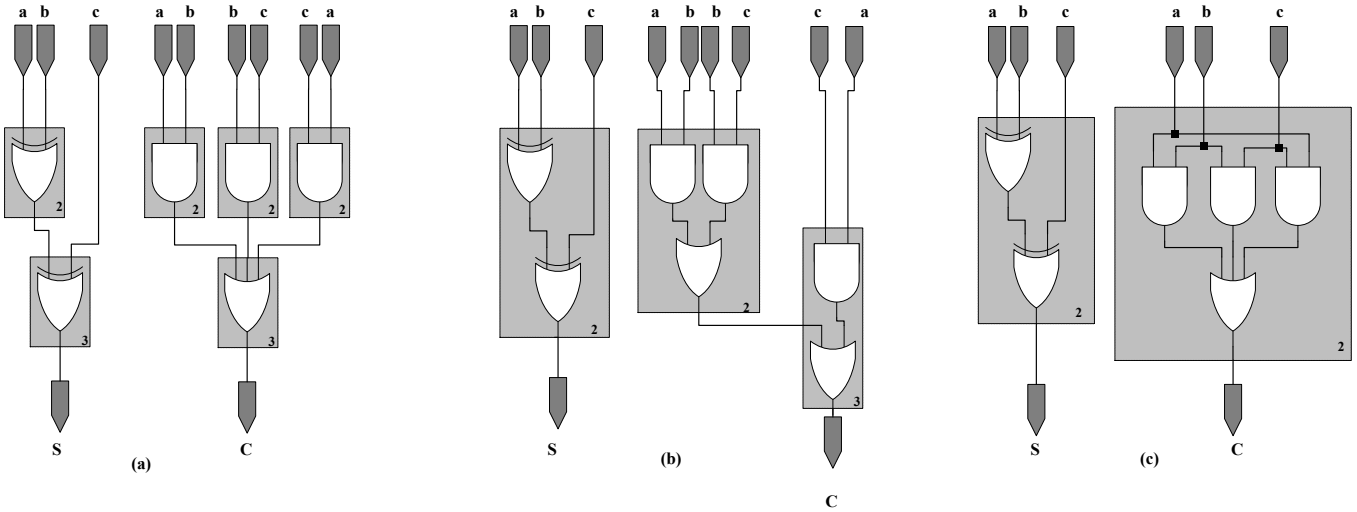

Fig. 2. Mapping of the RCA cell. a) Direct Mapping. b) Mapping using node decomposition c) Mapping exploiting reconvergent nodes.

The following instantiations were used in the design entry phase to map the circuit in Fig. 2(c).

```
    begin
    -- Optimal mapping for sum output
       L_1 : LUT3_L generic map (INIT => X"96")
          port map (S, c, b, a);
    -- Optimal mapping for carry output
       L_2 : LUT3_L generic map (INIT => X"E8")
          port map (C, c, b, a);
end Behavioral;
```

IV. Optimal Mapping for 6-Input LUT

The circuit in Fig. 2(c) may be an optimal circuit for a 3-input LUT but when the target element is a 6-input LUT, it leads to severe under-utilization of the resources resulting in a low-density circuit. Since most of the modern day FPGAs have 6-input LUTs as their basic logic element, it is compelling to devise a method that utilizes this circuit element efficiently. We counter this issue by considering two RCA cells simultaneously and restructuring the initial Boolean network so that the circuit obtained after transformation utilizes the targeted 6-LUT efficiently. Fig. 3(a) shows the Boolean network that corresponds to two full adder cells. The network may be partitioned into three separate sub-networks corresponding to two sum bits S(0) and S(1) and a carry bit C(1). The reconvergent nodes in the carry sub-network are exploited to reduce the number of inputs and a circuit implementation similar to Fig. 2(c) is obtained. This is shown in Fig. 3(b). However, an optimal 6-input LUT implementation may be achieved by replicating the logic at fan-out node Z as shown in Fig. 4(a). The replicated nodes are shown by shaded portions. Node replication ensures that the sub-networks S(1) and C(1) have the same inputs. The covering process covers the individual networks with suitable cones and each cone is mapped onto a separate LUT. Sub-networks S(1) and C(1) share the same inputs and are implemented using a single 6-LUT with dual outputs. The overall circuit is shown in Fig. 4(b). The depth of the circuit is three and the number of LUTs utilized is two. An n-bit adder implemented using this circuit will have an overall depth of (n/2)+2 and will utilize only n LUTs. Thus the implementation based on Fig. 4(b) is theoretically 50% more efficient than the one based on Fig. 2(c). Fig. 5 shows an 8-bit adder unit constructed using the optimized circuit of Fig. 4(b).

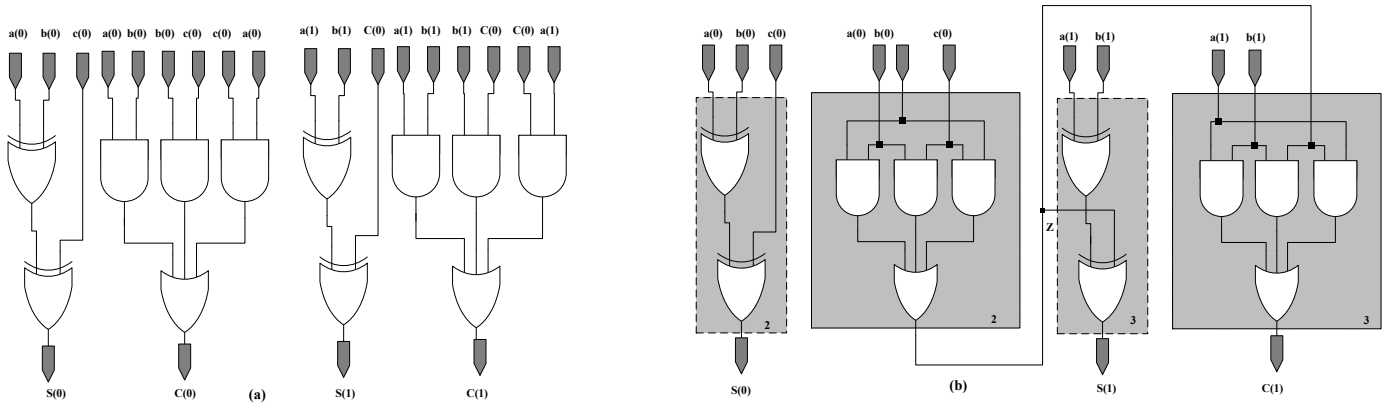

Fig. 3. a) Logic diagram for a 2-bit RCA adder. b) Mapping using 3-input LUTs.

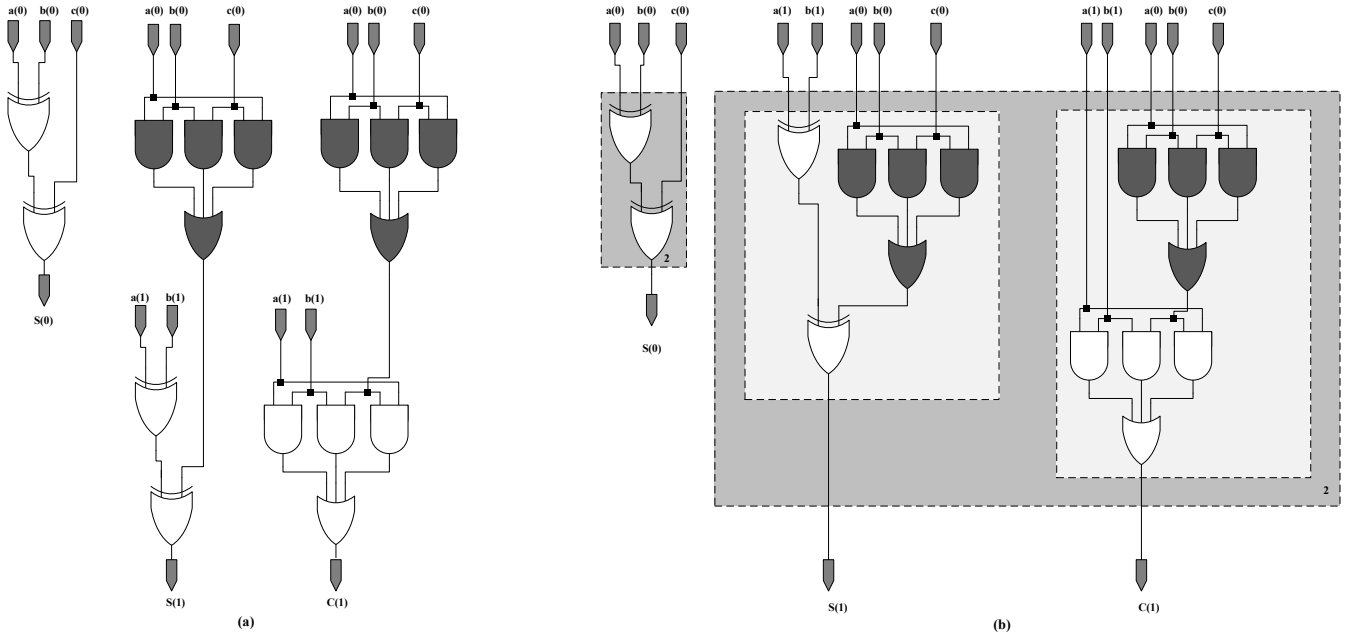

Fig. 4. a) Logic replication at node Z. b) Optimal 6-input LUT implementation.

The following instantiations were used in the design entry phase to map the circuit in Fig. 4(b).

```
begin
-- Optimal mapping for S(0) output
   L_1: LUT3_L generic map (INIT => X"96")
      port map (S(0), c(0), b(0), a(0));
-- Optimal mapping for S(1) and C(1) output
   L_2: LUT6_2 generic map ( INIT => X"E81717E8FFE8E800")
      port map (C(1), S(1), c(0), b(0), a(0), b(1), a(1), '1' );
end Behavioral;
```

V. Synthesis and Implementation

A. Methodology

The implementation in this work targets the XC5VLX30-3FF324 device from Xilinx Virtex-5 FPGA family. The implementation is carried out for an input word-length varying from 8 to 64 bits. The parameters considered are area, timing and dynamic power dissipation. Area is considered in terms of the number of occupied slices. Timing analysis may be static or dynamic. Static timing analysis gives information about the delay associated with the critical path and the maximum frequency at which the design may be operated. Dynamic timing analysis verifies the functionality of the design by applying test vectors and checking for correct output vectors. Dynamic timing analysis is done post implementation and PAR. The quality of dynamic timing analysis depends on the number of test vectors used. An important result from dynamic timing analysis is the switching activity information (toggle rates, signal rates etc.). This information is captured in the value charge dump (VCD) file and helps in determining accurate power measurements. Dynamic power dissipation is related to the charging and discharging of various node capacitances along different switching elements. To ensure a fair comparison, similar test benches have been used for all the implemented designs i.e. the input statistics remain same in each case. The initial design entry is done using VHDL through direct instantiation of the primitives rather than writing inferential codes and letting the synthesizer decide how to infer the logic. This ensures a fairly controlled mapping. The constraints relating to synthesis and implementation are duly provided and a complete timing closure is ensured in each case. The design synthesis and implementation is carried out in Xilinx ISE 12.1 [45] and the simulator database is then analyzed for speed and area metrics. Power metrics are obtained from Xpower analyzer.

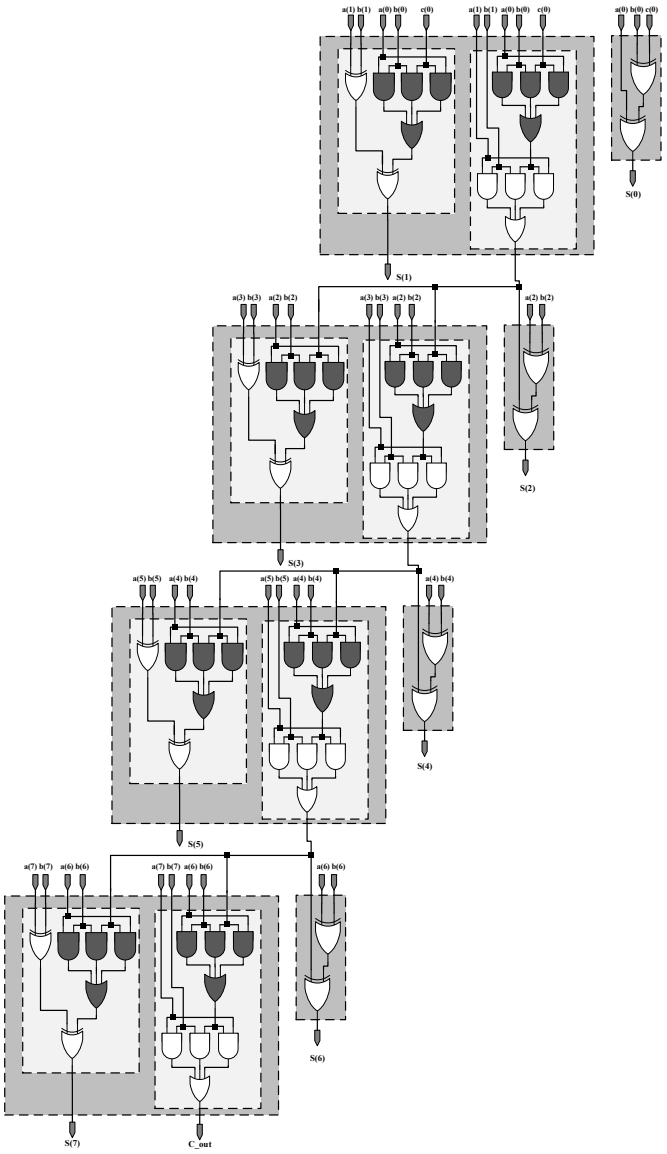

Fig. 5. 8-bit adder structure based on technology optimized binary adder cell.

B. Experimental results

We have compared our implementation results against the various fixed point adder designs in [44] and the Xilinx IP adder v 11.0.

TABLE I
RESOURCE UTILIZATION FOR DIFFERENT ADDERS ON XC5VLX30 FOR 16 BIT INPUT WORD-LENGTH

Adder Design	No. of occupied slices
Carry chain adder (CCA) [44]	9
Carry select adder (CSA) [44]	7
Carry skip adder (CKA) [44]	7
Carry look ahead adder (CLA) [44]	16
Sign magnitude adder (SMA) [44]	15
Xilinx IP adder v.11.0	4
3-input LUT based adder (LUT_3)	7
6-input LUT based adder (LUT_6)	**3**

Table I gives the comparison of resource utilization for various adder designs. The comparison is carried out for an input word-length of 16 bits. It is observed that technology mapping using LUTs results in a subsequent reduction of the on-chip resources being utilized. The most area efficient structure is obtained using 6-input LUT because of its ability to implement sum and carry sub-networks in a single LUT. Further analysis is carried out for different adders for varying word-lengths. The results are plotted as a function of word-length and appear in Fig. 6.

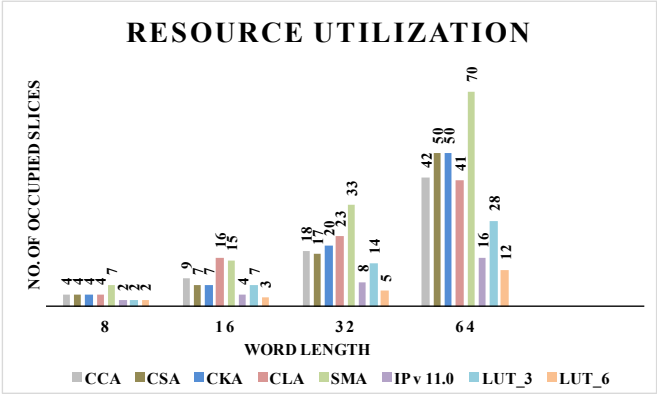

Fig. 6. Resource utilization for different adder structures.

Technology mapping using LUTs also reduces the depth of the implemented circuit resulting in shorter critical paths. Table II provides a comparison of the critical path delay for various adders for an input word-length of 16 bits.

TABLE II
CRITICAL PATH DELAY FOR DIFFERENT ADDERS ON XC5VLX30 FOR 16 BIT INPUT WORD-LENGTH

Adder Design	Critical path delay (ns)
Carry chain adder (CCA) [44]	7.872
Carry select adder (CSA) [44]	7.64
Carry skip adder (CKA) [44]	7.872
Carry look ahead adder (CLA) [44]	9.165
Sign magnitude adder (SMA) [44]	12.16
3-input LUT based adder (LUT_3)	7.148
6-input LUT based adder (LUT_6)	**6.969**

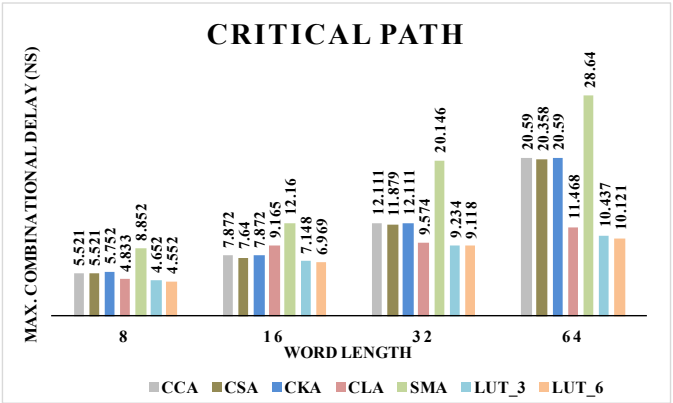

Fig. 7. Critical path delay variation for different adders.

Fig. 7 gives the variation in critical path delay for different adders as word-length is varied from 8 to 64 bits. We have also compared the maximum clock frequency for the 6-LUT

based adder and the Xilinx IP adder v 11.0. The results are shown for different word lengths in Table III.

TABLE III
MAX CLOCK FREQUENCY FOR IP BASED AND TECH. MAPPED DESIGNS

Word length	Max. Clock frequency (MHz)	
	Xilinx IP v 11.0	LUT_6
8	355.493	498.008
16	266.028	378.5011
32	211.372	287.7934
64	145.65	210.32

Finally dynamic power dissipation for different structures is considered. The dynamic power dissipation is a function of the input voltage (V^2), the clock frequency (f_{clk}), the switching activity (α), the total capacitance seen by a particular node (C_L) and the number of elements used (σ). The capacitance C_L, which needs to be driven at each toggling node, varies with the type, fan-out, and capacitance of the logic and routing resources used in the design. The use of LUTs ensures that the high activity switching nodes remain hidden. This reduces the charging and discharging of the capacitances associated with these nodes, resulting in reduced dynamic power dissipation. In addition, there is also a reduction in the number of elements (σ) being utilized which reduces the power dissipated in the logic. The analysis is done for a constant supply voltage and at maximum operating frequency for each structure. To ensure a reasonable comparison the test vectors provided during post route simulations are selected to represent the worst case scenario for data coming into the adders. Same test bench is used for all the synthesized structures. The design node activity captured in the VCD file along with the power constraint file (PCF) is used for power analysis in the Xpower analyzer tool. Table IV shows the comparison of dynamic power dissipation for various adders for an operand length of 16 bits.

Further analysis is carried out by plotting the total dynamic power dissipation as a function of input word-length for different adders. The result is shown in Fig. 8.

TABLE IV
DYNAMIC POWER DISSIPATION FOR DIFFERENT ADDERS ON XC5VLX30 FOR 16 BIT INPUT WORD-LENGTH

Adder Design	Dynamic power dissipation (Watt)
Carry chain adder (CCA) [44]	0.03608
Carry select adder (CSA) [44]	0.03604
Carry skip adder (CKA) [44]	0.03604
Carry look ahead adder (CLA) [44]	0.03625
Sign magnitude adder (SMA) [44]	0.03631
Xilinx IP adder v.11.0	0.026
3-input LUT based adder (LUT_3)	0.0193
6-input LUT based adder (LUT_6)	**0.01136**

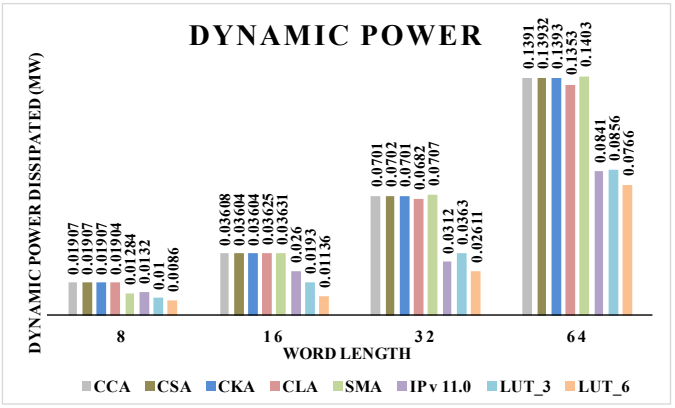

Fig. 8. Variation in Dynamic power dissipation with word-length.

VI. CONCLUSION

This paper implemented RCA based fixed-point adders by considering their mapping on LUT based FPGAs. The paper in particular targeted the 6-input LUT that is an inherent basic logic element in most of the modern day FPGAs. The optimization techniques used in this paper are purely technology dependent. Further hardware implementations presented in this paper were based on the primitive instantiations rather than the conventional inferential approaches. This ensured a controlled mapping of the optimized Boolean networks. The analysis and the experimental results presented in this paper clearly indicate that a considerable improvement in performance is achievable by technology dependent optimizations. No such analysis has been reported so far. By using a coding strategy based on instantiations the on-chip FPGA components can be used in a manner that fully utilizes their potential. This paper deliberately ruled out any technology independent (architectural) modification that may be carried out at the top level of the design. The idea was to present a clear cut analysis that will provide an insight about the performance speed-up that may be achieved by utilizing the huge primitive support provided by FPGA families through technology dependent optimizations. The future discourse will focus on achieving performance speed up in larger circuits like multipliers, multiply-accumulators etc. Also a combination of technology independent and technology dependent optimizations can lead to enormous improvement in performance and will encourage hardware intensive processing using FPGAs as a platform.

REFERENCES

[1] R. Naseer, M. Balakrishnan, and A. Kumar, "Direct Mapping of RTL Structures onto LUT-Based FPGAs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 17, No. 7, July 1998.

[2] R. Tessier and W. Burleson, "Reconfigurable Computing for DSP: A Survey," Journal of VLSI Signal Processing, Vol. 28, pp. 7-27, 2001, Kluwer Academic Publisher.

[3] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung, "Reconfigurable Computing: Architecture and Design Methods," IEEE Proceedings. Computer Digital Technology, Vol. 152, No. 2, March 2005.

[4] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software," ACM Computing Surveys, Vol. 34, No. 2, pp. 171-210, June 2002.

[5] Scott Hauck and Andre Dehon, "Reconfigurable Computing: The Theory and Practice of FPGA based Computation," Morgan Kaufmann Publisher, November 2007.
[6] Stephen D. Brown, Jonathan Rose, Robert J. Francis and Zvonko G. Vranesic, "Field Programmable Gate Arrays," Kluwer Academic Publisher, 1992.
[7] A. Ling, D. P. Singh and S. D. Brown, "FPGA Technology Mapping: A Study of Optimality," IEEE Proceedings Design Automation Conference, pp. 427-432, June 2005.
[8] D. Chen and J. Cong, "DAO map: A Depth-optimal Area Optimization Mapping Algorithm for FPGA Designs," IEEE/ACM International Conference on Computer Aided Design, November 2004.
[9] R. Rohleder, "Marker Overview: User_Programmable Logic", In-Stat Services Research Report, March 1991.
[10] J. H. Anderson, Q. Wang, "Area-Efficient FPGA Logic Elements: Architecture and Synthesis," 16th Asia and South Pacific Design Automation Conference (ASP-DAC), January 2011.
[11] G. C. Cardarilli, S. Pontarelli, M. Re and A. Salsano, "On the use of Signed Digit Arithmetic for the new 6-Inputs LUT based FPGAs," 15th IEEE International Conference on Electronics, Circuits and Systems, ICECS, September 2008.
[12] G. Zhou, L. Li and H. Michalik, "Area Optimization of Bit Parallel Finite Field Multipliers with Fast Carry Logic on FPGAs," International Conference on Field Programmable Logic and Applications, FPL, September 2008.
[13] S. Gao, D. A. Khalili and N. Chabbini, "Optimized Realization of Large-Size Two's Complement Multipliers on FPGAs," IEEE Northeast Workshop on Circuits and Systems, NEWCAS, August 2007.
[14] Altera Corporation, "Stratix III Device Handbook," V.1, November 2006.
[15] Xlinx DSP Design Considerations, Xtreme DSP for Virtex-4 FPGAs, UG073, (v 2.2), July 2006.
[16] Virtex-5 Family Overview LX, LXT, and SXT Platforms, Xilinx, Inc., San Jose, CA, 2010.
[17] Virtex-6 FPGA Data Sheet, Xilinx, Inc., San Jose, CA, 2010.
[18] Stratix-IV FPGA Family Data Sheet, Altera, Corp., San Jose, CA, 2010.
[19] R. Francis, J. Rose and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs," Proceedings of the 28th ACM/IEEE Design Automation Conference, 1991.
[20] R. Murgai, N. Shenoy, R. K. Brayton and A. S. Vinccentelli, "Improved Logic Synthesis Algorithms for Table Look Up Architectures," IEEE International Conference on Computer-Aided Design, ICCAD-91. Nov., 1991.
[21] K. Karplus, "Xmap: A Technology Mapper for Table-lookup Field-Programmable Gate Arrays," DAC, 1991.
[22] N. S. Woo, "A Heuristic Method for FPGA Technology Mapping Based on the Edge Visibility," DAC, 1991.
[23] P. Sawkar and D. Thomas, "Technology Mapping for Table-Look-Up Based Field Programmable Gate Arrays," ACM/SIGDA Workshop on Field Programmable Gate Arrays, Feb. 1992.
[24] J. Cong, C. Wu, and E. Ding, "Cut Ranking and Pruning: Enabling A General and Efficient FPGA Mapping Solution," FPGA, Feb. 1999.
[25] R. Francis, J. Rose, and Z. Vranesic, "Technology mapping for lookup table-based FPGA's for performance," IEEE International Conference on Computer-Aided Design, ICCAD-91. Nov., 1991.
[26] R. Murgai, N. Shenoy, R. K. Brayton and A. S. Vinccentelli, "Performance Directed Synthesis for Table Look Up Programmable Gate Arrays," IEEE International Conference on Computer-Aided Design, ICCAD-91. 11-14Nov., 1991.
[27] K. C. Chen, et al., "DAG-Map: Graph-based FPGA Technology Mapping for Delay Optimization," IEEE Design and Test of Computers, vol. 9, no. 3, pp. 7-20, Sep., 1992.
[28] J. Cong and Y. Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," ICCAD, Nov. 1992.
[29] H. Yang and D. F. Wong, "Edge-map: Optimal Performance Driven Technology Mapping for Iterative LUT based FPGA Designs," ICCAD, Nov. 1994.
[30] P. Pan and C.L. Liu, "Optimal Clock Period FPGA Technology Mapping for Sequential Circuits," DAC, June 1996.
[31] A.H. Farrahi and M. Sarrafzadeh, "FPGA Technology Mapping for Power Minimization," Proc. of Intl. Workshop in Field Programmable Logic and Applications, 1994.
[32] J. H. Anderson and F. N. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs," Proceedings of IEEE International Conference on Field-Programmable Technology (FPT), December 2002.
[33] Z. H. Wang et al., "Power Minimization in LUT-Based FPGA Technology Mapping," ASPDAC, 2001.
[34] H. Li, W. Mak, and S. Katkoori, "Efficient LUT-Based FPGA Technology Mapping for Power Minimization," ASPDAC, 2003.
[35] J. Lamoureux and S.J.E. Wilton, "On the Interaction between Power-Aware CAD Algorithms for FPGAs," IEEE/ACM International Conference on Computer Aided Design, 2003.
[36] D. Chen, et al., "Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages," FPGA, Feb. 2004.
[37] J. Cong and Y. Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 2, Issue. 2, pp. 137-148, June 1994.
[38] J. Cong and Y. Hwang, "Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping," Proceedings of the Third International ACM Symposium on Field-Programmable Gate Arrays, FPGA '95, Feb. 1995.
[39] Legl, B. Wurth, and K. Eckl, "A Boolean Approach to Performance-Directed Technology Mapping for LUT-Based FPGA Designs," DAC, June 1996.
[40] J. Cong and Y. Ding, "Beyond the Combinatorial Limit in Depth Minimization for LUT-Based FPGA Designs," ICCAD, Nov. 1993.
[41] S. C Chang, M. Marek-Sodowska, and T. Hwang, "Technology Mapping for TLU FPGA Based on Decomposition of Binary Decision Diagrams," IEEE Transactions on CAD, Vol. 15, No. 10, pp. 1226-1236, Oct. 1996.
[42] A. Farrahi and M. Sarrafzadeh, "Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping," *IEEE TCAD,* Vol. 13, No. 11, pp. 1319-1332, Nov. 1994.
[43] I. Levin, R.Y. Pinter, Realizing expression graphs using table-lookup FPGAs, in: Proceedings of the European Design Automation Conference, Hamburg, Germany, 1993, pp. 306–311.
[44] S. Bhattacharjee, S. Sil, B. Basak and A. Chakarbarti, "Evaluation of Power Efficient Adder and Multiplier Circuits for FPGA Based DSP Applications," International Conference on Communication and Industrial Application (ICCIA), December 2011.
[45] http://www.xilinx.com