Hindawi
International Journal of Reconfigurable Computing
Volume 2017, Article ID 7021056, 9 pages
https://doi.org/10.1155/2017/7021056

Research Article

Exploring Shared SRAM Tables in FPGAs for Larger LUTs and Higher Degree of Sharing

Ali Asghar,[1] Muhammad Mazher Iqbal,[1] Waqar Ahmed,[1] Mujahid Ali,[1] Husain Parvez,[1] and Muhammad Rashid[2]

[1]*Karachi Institute of Economics and Technology, Karachi, Pakistan*
[2]*Umm Al-Qura University, Makkah, Saudi Arabia*

Correspondence should be addressed to Ali Asghar; aliasghar89@gmail.com

Received 8 January 2017; Accepted 23 April 2017; Published 13 June 2017

Academic Editor: Seda Ogrenci-Memik

In modern SRAM based Field Programmable Gate Arrays, a Look-Up Table (LUT) is the principal constituent logic element which can realize every possible Boolean function. However, this flexibility of LUTs comes with a heavy area penalty. A part of this area overhead comes from the increased amount of configuration memory which rises exponentially as the LUT size increases. In this paper, we first present a detailed analysis of a previously proposed FPGA architecture which allows sharing of LUTs memory (SRAM) tables among NPN-equivalent functions, to reduce the area as well as the number of configuration bits. We then propose several methods to improve the existing architecture. A new clustering technique has been proposed which packs NPN-equivalent functions together inside a Configurable Logic Block (CLB). We also make use of a recently proposed high performance Boolean matching algorithm to perform NPN classification. To enhance area savings further, we evaluate the feasibility of more than two LUTs sharing the same SRAM table. Consequently, this work explores the SRAM table sharing approach for a range of LUT sizes (4–7), while varying the cluster sizes (4–16). Experimental results on MCNC benchmark circuits set show an overall area reduction of ~7% while maintaining the same critical path delay.

1. Introduction

Look-Up Tables (LUTs) in an FPGA offer generous flexibility in implementing logic functions. LUT is an $N:1$ multiplexer (MUX) with an N-bit memory [1]. Since MUX is a universal logic block; a k-input LUT can implement any k-variable Boolean function. Several LUTs are grouped together to form larger aggregates called Configurable Logic Blocks (CLBs) or simply clusters. LUTs inside a CLB are connected via intra-clustering routing network, while CLBs are connected with each other through a configurable routing network. However, this flexibility in an FPGA comes at the expense of area and performance overheads [2] when compared with their Application Specific Integrated Circuits (ASICs) counterparts, which are highly optimized for a certain class of applications. Hence, the very feature of FPGAs that makes them special is also responsible for their inferior performance to ASICs.

To bridge this gap between FPGAs and ASICs, FPGA architectures have been under continuous overhaul, ever since their inception. Previously published articles such as [3–6] attempt to explore the optimum values for coarser architecture level details such as cluster size (N), the number of inputs to a cluster (I), and the cross-bar topologies [7, 8]. FPGA's reconfigurable routing network, its switch box, and connection boxes have also been explored in detail.

In the past few years, some research has been focused towards exploring innovative logic blocks for FPGA, such as [9–11], which can compromise flexibility in favor of improving area and performance. The logic block architectures proposed in these works [9–12] replace legacy LUTs with innovative high coverage logic elements derived from frequently appearing logic functions. The idea is based on the fact that not all logic functions appear with the same frequency in digital circuits [9, 12].

All of the architectures discussed above utilize the concept of NPN-class equivalence [13] to characterize the frequency with which logic functions occur in a circuit.

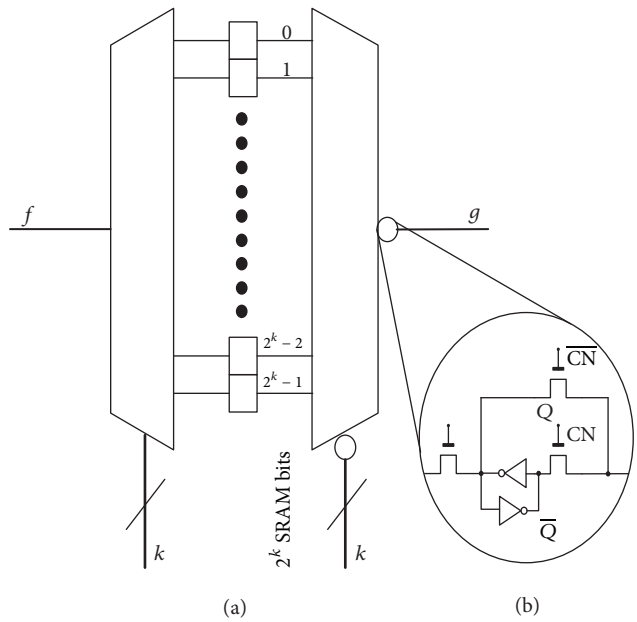

Figure 1: (a) LUTs with shared SRAM vectors and (b) CN logic.

The use of NPN-equivalent classes removes the redundancy (because of LUTs) inherent to FPGAs with some compact high coverage logic blocks. Other researchers have attempted to optimize FPGA logic blocks on a coarser architecture level which include [14, 15]. An SRAM table sharing based CLB [15] shares a single SRAM table between two or more LUTs, where all the LUTs sharing a single SRAM table map NPN-equivalent functions. The novel SRAM table sharing based CLB proposed in [15] has been improved and further explored in this research work.

The main drawback of the logic blocks proposed in [9–11] is that they are derived on the basis of NPN-equivalent classes for a particular benchmark suite; hence, they offer high-end efficiency only for the circuits from which their NPN classes were derived. For example, the logic blocks of [9] perform remarkably for the MCNC benchmarks, while for the VPR benchmark suite they fail to provide coverage for most of the frequently appearing logic functions. However, the SRAM table sharing based CLB is generic enough to provide area benefits for any set of benchmark circuits.

Meanwhile, a lot of research has also been directed towards architectures with reduced number of configuration memory cells. Architectures such as [14–17] fall in this category. The work in [17] utilizes the concept of Shannon Decomposition to trim down a larger k input function into two smaller ones, where one of the partial functions has less than $k - 1$ variables. The logic blocks (termed as Extended-LUT) used to map these functions require a smaller number of configuration memory bits than the conventional LUTs. The authors of [17] estimate improvements in area-depth product, without performing the place and route experiments. Hence, the efficiency of their proposed Extended-LUT [17] remains vague. Also the proposed logic cells are not fully permutable, which may result in routing overheads. Another study [16] introduces the Scalable Logic Module (SLM) architecture which like [17] makes use of the Shannon Decomposition to find NPN-equivalent interconvertible partial functions, which can allow the sharing of their memory tables. The results show that a high percentage of functions with input size of 5–7 can be decomposed into interconvertible partial functions.

Kimura et al. [14] proposed function folding to reduce the number of configuration memory bits. The truth table of a function is divided into 2 parts; the whole truth table is then reconstructed using only a single part, while the other half is extracted using NOT, bit-wise OR, or any other suitable operation. However, [14] does not include delay results.

This work employs a novel CLB [15] to reduce the number of configuration memory bits. The reduction in configuration memory bits will reduce not only the area of the FPGA architecture, but also the configuration time and the size of the external memory used to store the bitstream. The CLB proposed in [15] allows sharing of memory vectors between 2 LUTs (as shown in Figure 1) on which NPN-equivalent functions are mapped. To realize NPN equivalence on hardware level, the inputs and output of one of the two shared LUTs are negated with the help of conditional negation (CN) block, as shown in Figure 1(b). To allow sharing of SRAM tables between two NPN-equivalent functions, an additional circuitry, conditional negation (CN) logic is added to the I/Os of one of the two shared LUTs which share their SRAM vectors.

The experiments in [15] were performed for LUT sized 4 ($k = 4$), while varying the cluster size (4–16) and the number of shared LUT pairs (2–4). The results show an estimated ~2% reduction in total FPGA (logic + routing) area. In this article, we investigate the feasibility of architecture presented in [15] for a range of input ($k = 4$–7) and cluster ($N = 4$, 10, 16) sizes. We have also performed experimentations with

a much higher number of shared LUT pairs (i.e., 2–5 and 2–8, for $N = 10$ and 16, resp.). We have performed all the experimentation on the newer VTR exploration environment [18] with single driver unidirectional [19] routing network. On the contrary, the work done by [15] employed the older VPR with bidirectional routing network. We also propose a new clustering technique which attempts to pack the NPN-equivalent LUTs together inside the same CLB and present a comparison of our proposed technique with the one implemented in [15]. To find the NPN-equivalence among logic functions, we make use of an extremely efficient, high performance Boolean matching algorithm [20] instead of the brute-force approach employed in [15].

In an earlier work [21], we showed the feasibility of this architecture by obtaining an area gain of ~3.7%, for degree of sharing 2, that is, 2 LUTs sharing the same SRAM table. In this paper we attempt to enhance the area savings further, by exploring the possibility of CLBs with a higher degree of sharing (i.e., 3 or more LUTs sharing the same SRAM table).

The remainder of this paper is organized as follows. Section 2 covers the implementation and CAD flow details. The results are discussed in Section 3, while Section 4 presents the conclusion.

2. Implementation Details

This section describes the steps involved in mapping NPN-equivalent functions to LUTs with shared SRAM tables.

2.1. Finding NPN-Equivalent Classes. Two functions, say f and g, are NPN equivalent, if one can be derived from the other by negating (N) and/or permuting (P) some/all of the inputs and/or by negating (N) the outputs. If f and g are NPN equivalent they belong to the same NPN class.

For n-inputs, there are $2\exp(2^n)$ distinct possible functions. Hence, the Boolean space grows very rapidly as the number of inputs increase. However, this space can be compressed to a much smaller number of unique functions using the concept of NPN-equivalent classes. For example, for $n = 4$, there are $2\exp(2^n) = 65536$ possible functions while there are only 222 distinct NPN-equivalent classes [13]. The task of Boolean matching (which involves pruning the logic space to find equivalent functions) becomes much simpler for the NPN-equivalence space compared to the humongous $2\exp(2^n)$ possible logic functions.

In [15], a brute-force method has been used to find NPN equivalence among logic functions. However, this approach is not scalable as the number of inputs increases. Hence, in this paper we make use of a recently proposed state-of-the-art Boolean matching algorithm [20], which has been integrated into the ABC framework [22] for finding NPN-equivalent classes.

2.2. Clustering NPN-Equivalent Classes. After finding the equivalent classes, LUTs are clustered using T-VPACk algorithm [23] with an emphasis on packing NPN-equivalent functions in such a way that LUTs with shared SRAM tables employ functions which belong to the same NPN class. For example, consider a CLB (shown in Figure 2) with 10 LUTs ($N = 10$), having 3 shared LUT pairs, that is, 3 LUT pairs sharing their SRAM tables ($p = 3$); hence, there will be a total of 7 SRAM vectors for 10 LUTs.

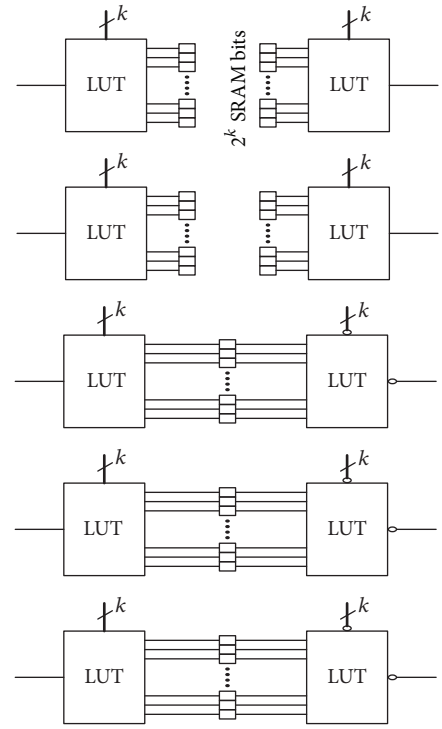

Figure 2: A CLB with 10 LUTs ($N = 10$) and 3 shared pairs ($P = 3$).

We employ two clustering approaches which attempt to map NPN-equivalent functions on these shared pairs. The one used in [15] (Algorithm 1) and its modified version (Algorithm 2) are proposed in this work. The first step of the two clustering methods is the same as both algorithms start by filling one of the two LUTs sharing their SRAM tables, with the LUTs present in the priority queue generated by the T-VPACK [23] algorithm. However, it is the second part in which the two clustering methods change course. To fill the second of the two LUTs in each pair, the method presented in [15] performs NPN-equivalence checks with only a single pair out of the possible p shared pairs. On the other hand, our clustering approach is far more flexible as it performs equivalence checks for all the possible p shared pairs. Hence, our proposed method offers flexibility at the expense of $p - 1$ more equivalence checks as compared to [15].

A comparative analysis of the two approaches has also been performed which will be discussed in Section 3.2. We have not made any changes to the priority queue generated by the T-VPACK [23] to ensure the optimal packing results. Finally, placement and routing results were obtained using VTR [18] with default settings. This modified CAD flow is shown in Figure 3.

2.3. Architecture Details. All the architecture files we have used for experimentation extract their parameters from the iFAR *(intelligent FPGA architecture repository)*. The iFAR

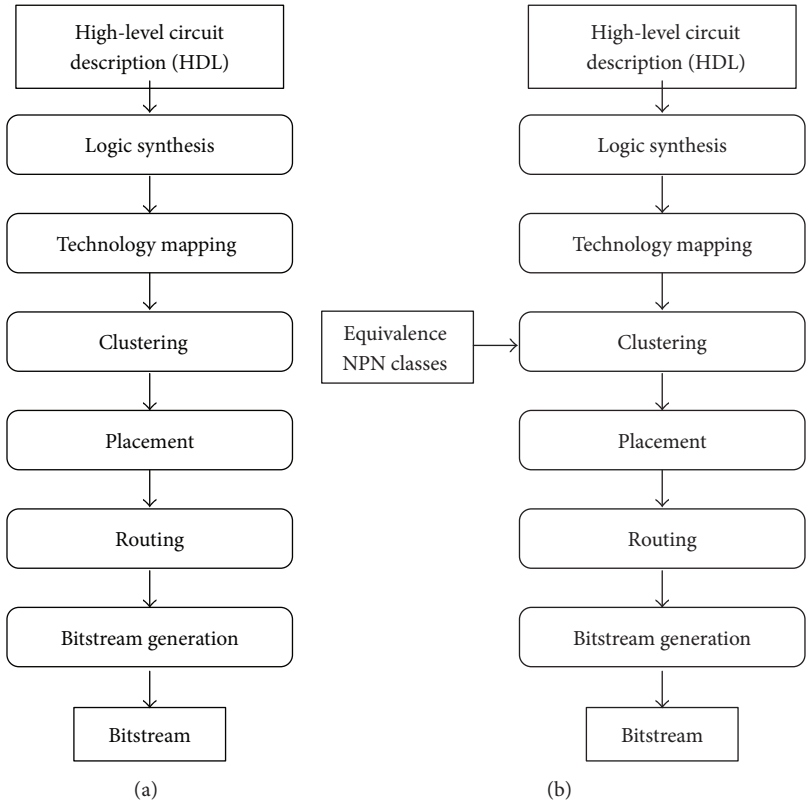

Figure 3: Modified CAD flow with equivalence analysis.

repository [24] has a wide variety of architecture files for different input and cluster sizes. It also contains architecture files for four different technology nodes (45 nm, 65 nm, 90 nm, and 130 nm) which extract parameters from *Predictive Technology Model* (PTM) [25]. The PDK available to us to estimate the delay of CN logic (discussed in Section 1) is on 150 nm process. Hence, the delay estimated using the PDK available to us, that is, 150 nm, would best correlate with the delay results for architecture files based on 130 nm technology node. We do not expect the delay values to vary much in moving from 150 nm to 130 nm process. Therefore, all the parameters for the architecture files used in our simulations have been obtained from the iFAR repository architecture files for 130 nm.

The number of inputs to the cluster (I) has been calculated using this well-established relation [26]:

$$I = \frac{K}{2} * (N + 1). \quad (1)$$

Table 1 lists some of the most important architecture file parameters which affect the final results.

2.4. Configuration Bits Analysis. Since k input LUT requires 2^k configuration memory bits, hence, for every shared SRAM pair 2^k memory bits are saved. Xilinx [27] uses 5-transistor-based 5T SRAM cell as a configuration memory bit for its FPGA families. VPR [18] employs a 6T SRAM cell for LUT configuration and for the overall architecture. The paper [15] uses an 8T SRAM cell, while the [16] architecture utilizes latch as configuration memory cells. Naturally, larger memory cells (in terms of transistor count) result in greater area savings for an FPGA. Since we have obtained the entire place and route results using VPR, we use a 6T SRAM cell to get a true picture of the overall area savings for an FPGA.

Table 1: Architecture file parameters.

Parameter	Value
RMinWidth NMOS	2800
Switch block type	Wilton
Switch type	MUX
Segment length	4
RMinWidthPMOS	7077
Fs	3
Switch delay	103 ps
Segment type	Unidir.

2.5. Area Savings. The model employed in VPR to perform area estimation is known as Minimum Width Transistor Area Model (MWTM) [1]. MWTA is simply the layout area occupied by the smallest possible transistor that can be contacted in a process, plus the spacing to other transistors above and to its right. Using MWTA, the calculated area for a 6T SRAM cell is 6 MWTAs (6 minimum width transistor areas). The CN logic block presented in [15] consists of a 5T

SRAM cell and two pass-transistors which overall requires 7 MWTAs. For the remainder of this paper, we will use these values to perform area calculations.

For a k input LUT, the number of shared pairs equals p. The number of transistors removed from a CLB can be given as

$$\text{No. of Transistors Removed} = p * \left(6 * 2^k - 7 * m\right), \quad (2)$$

where $m = k + 1$ represents the number of CN logic cells required to realize a k input LUT pair with shared SRAM vectors.

Since, for a k input function, an SRAM vector has 2^k, 6T SRAM cells, therefore, for p shared pairs, $p * (6 * 2^k)$ transistors are removed from each CLB, while the term $7 * (m * p)$ in (2) accounts for the transistor overhead required to implement the CN logic appearing at $k + 1$ I/Os. Consequently, the area saved (in terms of transistors count) resulting from the reduction of SRAM configuration cells over the entire FPGA can be calculated by simply multiplying (2) by the FPGA grid size:

$$\text{Area Savings} = \left(6 * 2^k * p * \text{Grid Size}\right) - \left(7 * m * p * \text{Grid Size}\right). \quad (3)$$

2.6. Delay Analysis with CN Logic Cells. The LUTs whose I/Os are appended with CN logic cells will incur an additional delay in their look-up times. To estimate this delay, we simulated the CN logic cell in Cadence Virtuoso 6.1.5 using 150 nm process. The propagation delay is the average value for rise and fall transitions. The propagation delay of the CN logic (ε) was found to be ~15 psec, which is a small fraction of the overall time (τ) required to look up a value from the SRAM table. The value of (τ) for LUT-4 (as reported in the iFAR [24] repository files) is ~294 psec. It increases as the number of inputs of an LUT increases. Hence, the total delay through an LUT with CN logic cells at both input and output should be $\tau + 2\varepsilon$.

3. Results

To evaluate the performance of the CLBs with shared SRAM tables, we have performed rigorous testing on a variety of architectures by varying k, N, and p.

In this section, we present our observations for the following set of results:

(1) Number of NPN-equivalence classes as a function of input size

(2) Comparison of the two clustering methods (discussed in Section 2.2)

(3) Effects of varying the number of shared pairs (p)

(4) Effects of varying the degree of sharing (d); number of LUTs sharing the same SRAM vector

(5) Impact of modified clustering on routability

(6) Channel width, critical path, logic area, and total FPGA area for cluster sized $N = 10$ and 16

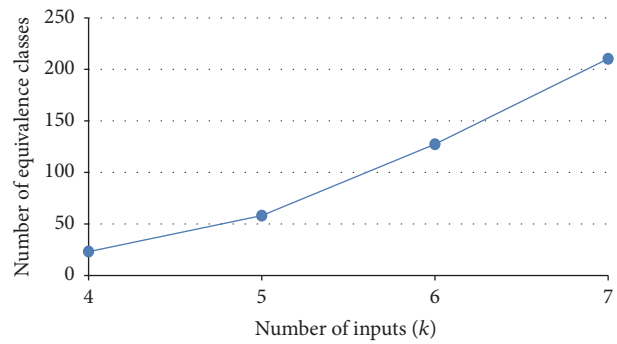

Figure 4: Number of used NPN-equivalence classes for input size k = 4–7.

3.1. Number of NPN-Equivalence Classes as a Function of Input Size. NPN-Equivalence analysis is performed for an input circuit after it is synthesized as shown in Figure 3. As discussed earlier, for n inputs there are $2\exp(2^n)$ possible Boolean functions. These Boolean functions can be mapped upon a limited set of NPN-equivalent classes. For example, for $K = 4$, there can be a total of 222 NPN-equivalent classes. However, a given input circuit does not utilize all of the NPN-equivalent classes. As the number of inputs increases the Boolean space grows exponentially. A direct consequence of this behavior is the quick rise in the number of NPN-equivalent classes used. Figure 4 which shows the average number of NPN-equivalence classes, for input sizes $k = 4$ to 7, over the 20 MCNC benchmark circuits, depicts this behavior very clearly. The NPN-class size grows from ~23 for $k = 4$ to ~210 for $k = 7$. Therefore, as the number of inputs increases the probability of mapping two NPN-equivalent classes on a shared LUT pair decreases and vice versa.

3.2. Comparison of the Two Clustering Methods. In Section 2.2, we discussed two clustering methods (Algorithm 1 and 2). The CLB, shown in Figure 2, contains a mix of LUTs, those that could share their SRAM tables and others which do not share their tables. Hence, both the clustering approaches (discussed in Algorithms 1 and 2) pay attention to this detail while packing CLBs. If a packing algorithm fails to map NPN-equivalent functions on a LUT pair with shared SRAM vectors, then the LUT with CN logic at its I/Os is left vacant. As a result, the CLB does not get fully packed, if this happens for a large number of shared LUT pairs, then the number of utilized CLBs in the FPGA architecture would rise which will affect the final place and route results and may eventually increase the FPGA grid size.

Hence, to evaluate performance of the two methods, we observed the number of CLBs required to implement the whole circuit. The number of required CLBs serves as a measure of relative efficiency; the clustering approach which requires more CLBs is not well-suited to map NPN-equivalent functions on LUTs which allow sharing their SRAM tables. Also an increase in the number of CLBs affects the final place and route results like critical path delay and routing channel width.

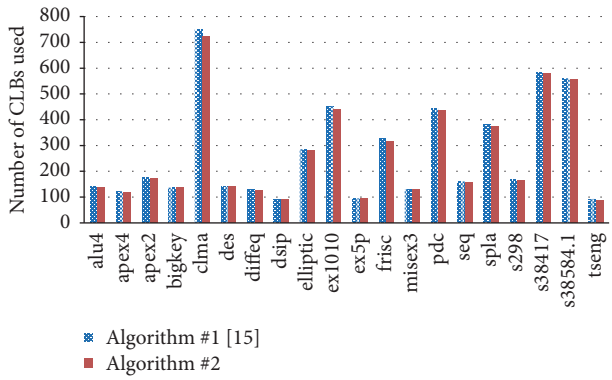

Figure 5: Comparison of the clustering methods for $N = 10$.

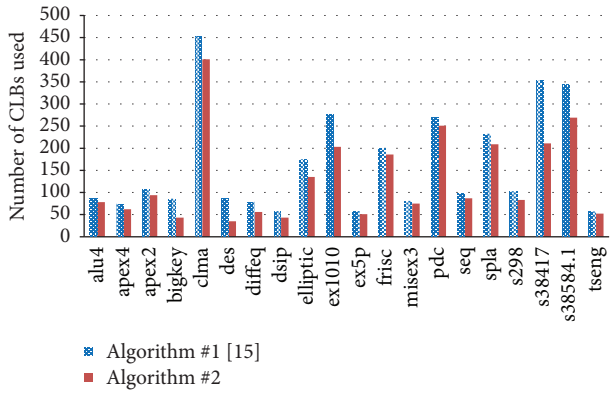

Figure 6: Comparison of the clustering methods for $N = 16$.

In Figures 5 and 6, we present the performance of the two algorithms for $k = 4$ and 5, with a cluster size of $N = 10$ and 16, respectively, and number of shared pairs $p = 3$. As seen from the results in Figures 5 and 6, our proposed method outperforms the other approach for both the architectures $k4N10$ and $k5N10$, over the entire MCNC benchmark suite. The reason for this improved performance is the flexibility (at the expense of more equivalence checks) of our clustering approach (Algorithm 2) compared to the one presented in [15]. Since Algorithm 2 is better suited to map NPN-equivalent functions on a shared LUT pair, we will use the results obtained from it for the remainder of this section.

3.3. Effects of Varying the Number of Shared Pairs. The area and configuration memory savings are directly related to the number of shared SRAM tables in a CLB. For example, in a CLB with cluster size $N = 10$ and $p = 5$, there are 5 SRAM tables which are shared between 10 LUTs resulting in exactly 50% reduction in configuration memory. For $p = 4$ and 3, the memory savings reduce by 40% and 30%, respectively. Hence, greater are the numbers of shared pairs greater are the configuration memory savings, which result in more area savings.

Although a high value of p seems ideal for the CLBs under exploration, there is an issue which restricts setting arbitrarily high values for p. Since a high value of p means that a large number of LUTs with shared memory vectors should

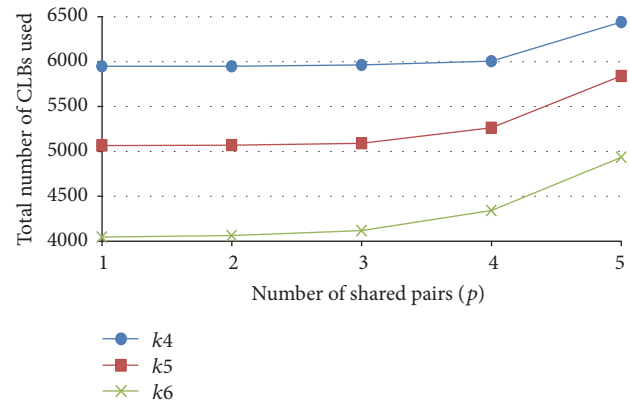

Figure 7: Number of CLBs utilized for cluster size of $N = 10$ and $p = 1$–5.

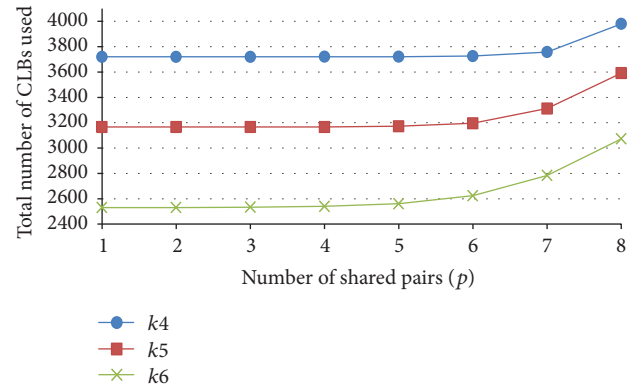

Figure 8: Number of CLBs utilized for cluster size of $N = 16$ and $p = 1$–8.

be mapped with NPN-equivalent functions, as the input size increases the number of NPN-equivalent classes grows rapidly (discussed in Section 3.1). As a result, it becomes increasingly difficult to map all the LUTs (with shared SRAM vectors) with NPN-equivalent functions.

Figures 7 and 8 depict this behavior with a high degree of clarity. The figures show the sum of the number of CLBs utilized to implement the entire MCNC benchmark suite, by varying the number of shared LUT pairs (p) from 1 to 5, for $N = 10$ and $p = 1$–8, for $N = 16$. The results obtained are concordant with our earlier observations that a high value of k or p makes it difficult to map all the shared LUT pairs with NPN-equivalent functions. The results in Figures 7 and 8 show a trend similar to a switching behavior; that is, for a particular value of k, the number of shared LUT pairs (p) can be increased up to a certain limit, after which the number of CLBs required to implement a circuit grows rapidly. For $k = 4$, this switching threshold is at $p = 4$ and $p = 6$, for cluster sizes $N = 10$ and 16, respectively. However, for $k = 6$, these values drop down to $p = 2$ and 4, for $N = 10$ and 16, respectively.

3.4. Effect of Varying the Degree of Sharing. The idea of LUT pair sharing a memory vector can be extended to 3 or more

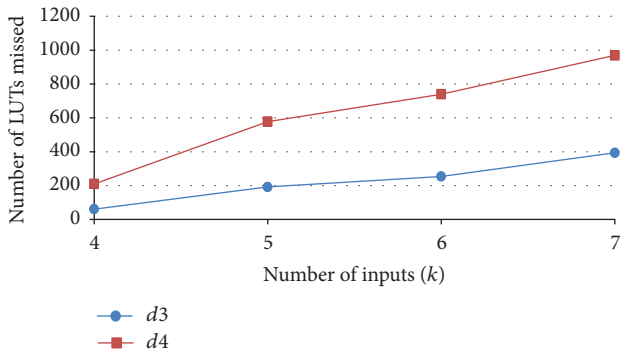

FIGURE 9: Number of LUTs missed for cluster size $N = 16$ and $p = 3$ for $d = 3$ and 4.

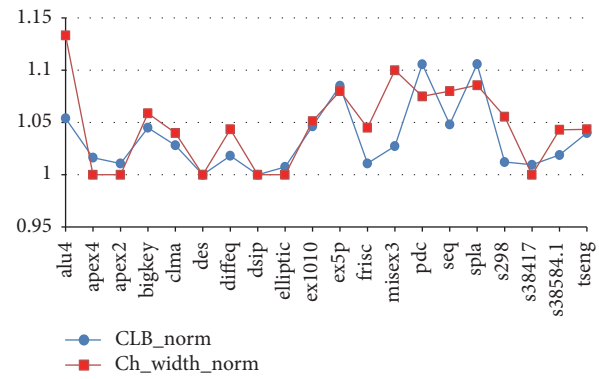

FIGURE 10: Impact of high CLB usage on channel width.

LUTs, if all the LUTs have been mapped with the same NPN class. In this article, we extend the degree of sharing (d) of SRAM tables from 2 to 3 and 4 LUTs sharing the same SRAM vector. Naturally, a high value of d means greater area savings because now a larger number of LUTs are sharing the same configuration bits. To account for the increased value of d, we have augmented the relationship for area savings presented in (3) with a new term $(d - 1)$

$$\text{Area Savings} = \left(6 * 2^k * p * (d-1) * \text{Grid Size}\right) - \left(7 * m * p * \text{Grid Size}\right). \quad (4)$$

However, similar to the number of shared pairs (p), the value of d cannot be made arbitrarily high. A high value of d means a large number of LUTs should be mapped with the same NPN class. This will happen only when there is high percentage of functions in the priority queue belonging to the same NPN class. For small input sizes ($k = 4$ or 5) the chances of finding such functions are pretty high, as the number of equivalent classes for functions with smaller inputs is usually low.

To depict this behavior we plot the number of LUTs left empty during clustering due to the lack of NPN-equivalent classes in the priority queue against the varying input size (k). Figure 9 shows the number of unmapped LUTs for $d = 3$ and 4 for the architecture having $p = 3$ and $N = 16$.

3.5. Impact of Modified Clustering on Routability. In Sections 3.3 and 3.4, we showed that high values of p and d make it very difficult to map all the shared LUTs with equivalent classes. As a result, the number of CLBs required to implement the whole circuit goes up, which may affect the final place and route results. Figure 8 shows the number of CLBs used to implement the entire MCNC benchmark suite for cluster size $N = 16$ and by varying the number of shared pairs (p) from 1 to 8. The results show a slight increase in the number of required CLBs for $k = 6$ and $p = 6$.

To analyze the impact of greater CLB usage on the circuits routability, we normalize the number of CLBs and channel widths required to implement the entire MCNC benchmark circuits using the modified architecture $k6N16$ (with $p = 6$ and $d = 2$) with the number of CLBs and channel widths required using the default $k6N16$ architecture. The results in Figure 10 show a clear trend; as the number of utilized CLBs goes above the default value, the channel width required to route the circuit increases accordingly.

3.6. Cluster Size $N = 10$. We first consider the CLBs with cluster size $N = 10$. The results obtained for the 20 MCNC benchmark circuits have been presented in Figures 11 and 12. All the results have been obtained by running VPR in the default mode of operation. In the default mode, VPR attempts to place the circuit on the minimum possible number of logical resources and for routing employs a binary-search algorithm which repetitively searches for the minimum value of channel width for which the circuit becomes routable. We compare our results with the default cluster size $N = 10$ architecture, without any shared LUT pairs.

For the experimentation with cluster size $N = 10$, we set the number of shared pairs equal to 3 ($p = 3$), while for the degree of sharing ($d = 3$ and 4) p was set to a value of 2. As can be seen from Figure 7, for input sizes $k = 4$, 5, and 6; the number of utilized CLBs stays relatively constant up to $p = 3$, after which they start growing rapidly. As mentioned in Section 3.3, the higher the value of p or d, the more difficult it becomes for the clustering algorithm to map NPN-equivalent functions to a large number of LUTs with shared SRAM vectors. Moreover, large values of k further exacerbate this problem, as the size of NPN-equivalence class grows rapidly with the increasing input size (as shown in Figure 4), which results in a wide variety of NPN-equivalence classes in the priority queue, making it difficult for the clustering algorithm to find NPN-equivalent functions for a large number of shared LUT pairs, resulting in higher CLB utilization (as discussed in Section 3.2).

The results in Figure 11 show the average (over the 20 MCNC benchmark circuits) savings in the total FPGA area, as the number of inputs and degree of sharing increase. The results in Figure 11 show a progressive increase in the total area savings with the increasing input size and degree of sharing from ~0.8% for $k = 4$ and $d = 2$ to ~3.9% for $k = 7$ and $d = 4$. The behavior depicted by the results in Figure 11 is in agreement with (4), according to which the progressive area gains can be attributed to the rapidly growing 2^k term.

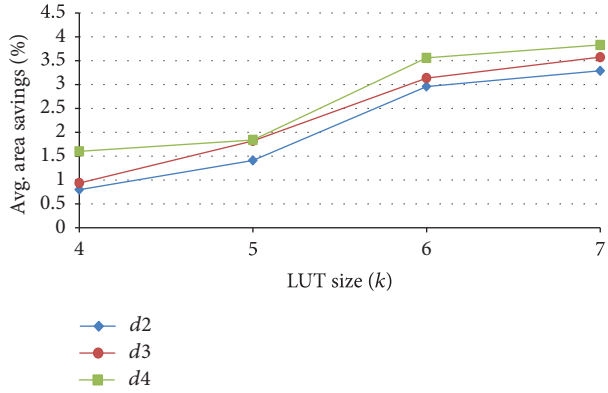

Figure 11: Average area savings for $N = 10$ and $k = 4$–7 over the 20 MCNC benchmark circuits.

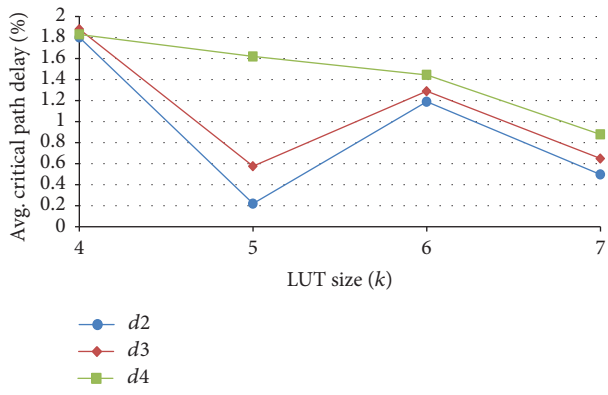

Figure 12: Average critical path for $N = 10$, $k = 4$–7, and $d = 2$–4 over the 20 MCNC benchmark circuits.

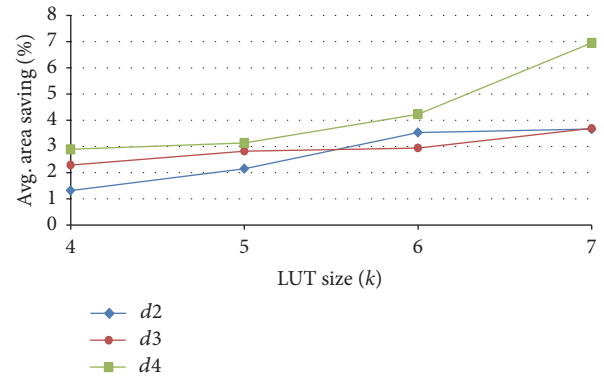

Figure 13: Average area savings for $N = 16$, $k = 4$–7, and $d = 2$–4 over the 20 MCNC benchmark circuits.

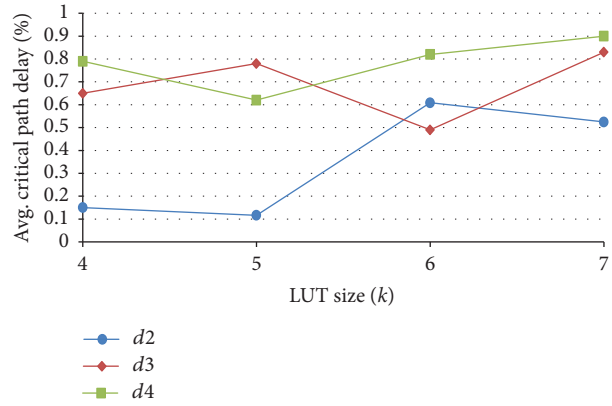

Figure 14: Average critical path for $N = 16$ and k=4–7 over the 20 MCNC benchmark circuits.

On the other hand, the results for the average (over the entire MCNC benchmark suite) critical path (shown in Figure 12) show a consistent behavior, with the average delay values staying within 2% of the results obtained for the conventional cluster size $N = 10$ architecture without any shared LUT pairs.

3.7. Cluster Size $N = 16$. The results obtained for the cluster size $N = 16$, for the 20 MCNC benchmark circuits, have been presented in Figures 13 and 14. The results have been compared with the default cluster size $N = 16$ architecture, without any shared LUT pairs. For the experimentation with cluster size $N = 16$, we set the number of shared pairs equal to 5 ($p = 5$), while for the degree of sharing ($d = 3$ and 4) p was set to values 4 and 3, respectively. Based on the observations from Figure 8, it is clear that, for input sizes $k = 4$, 5, and 6, the number of utilized CLBs stays relatively constant up to $p = 5$, after which they start growing rapidly. The reason for decreasing the value of p has been discussed in Section 3.6.

The results in Figure 13 show the average savings in the total FPGA area, as the number of inputs and degree of sharing increase. Similar to the trend observed in Figure 11, the area savings for cluster size $N = 16$ increase progressively as the input size and degree of sharing grow from $k = 4$–7, from ~1.32% for $k = 4$ and $d = 2$, to ~7% for $k = 7$ and $d = 4$. Also the results for the average critical path delay (shown in Figure 14) follow a similar consistent pattern (as seen in Figure 12, for cluster size $N = 10$), staying within 1% of the results obtained for the conventional cluster size $N = 10$ architecture without any shared LUT pairs.

4. Conclusion

Most of the recently proposed FPGA architectures focus on replacing legacy LUTs with innovative, high coverage logic blocks. Although such logic blocks offer high area and performance gains for a particular benchmark suite, they are not generic enough to maintain quality of results over a wide range of circuits. In this paper, we have explored a novel FPGA architecture which allows sharing LUTs SRAM vectors between NPN-equivalent functions. To find NPN equivalence, a very high speed state-of-the-art Boolean matching algorithm has been employed. Furthermore, an efficient packing technique has also been introduced to cluster NPN-equivalent functions together inside a CLB. By using CLBs with shared LUTs (for cluster size $N = 16$, $k = 7$, $p = 3$, and $d = 4$), the configuration memory cells of logic blocks were

reduced by ~56% which resulted in area savings of up to ~7%, with a negligible penalty on the critical path delay (<1%).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This project is funded by NSTIP, Saudi Arabia. The authors acknowledge the support of STU (Science and Technology Unit), Umm Al-Qura University, Saudi Arabia.

References

[1] V. Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs," Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[2] I. Kuon and J. Rose, *Quantifying and Exploring the Gap between FPGAs and ASICs*, Springer Science and Business Media, 2010.

[3] V. Betz and J. Rose, "How much logic should go in an FPGA logic block," *Design & Test of Computers, IEEE*, vol. 15, no. 1, pp. 10–15, 1998.

[4] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," in *Proceedings of the ACM/SIGDA 8th International Symposium on FPGAs*, 2000.

[5] A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you do not really want 100% LUT utilization)," in *Proceedings of the ACM/SIGDA 7th International Symposium on FPGAs*, Montery, CA, USA, 1999.

[6] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic blocks to improve FPGA Speed and Density," in *Proceedings of the ACM/SIGDA 7th International Symposium on FPGAs*, pp. 203–213, Monterey, CA, USA, 2000.

[7] J. Rose and S. Brown, "Flexibility of interconnection structures in field-programmable gate arrays," *IEEE Journal of Solid State Circuit*, vol. 26, no. 3, 1991.

[8] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT," in *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, 2001.

[9] Y. Okamoto, Y. Ichinomiya, M. Amagasaki, M. Iida, and T. Sueyoshi, "COGRE: a configuration memory reduced reconfigurable logic cell architecture for area minimization," *Field Programmable Logic and Applications (FPL)*, pp. 304–309, 2010.

[10] B. K. I. Ahmadpour and H. Asadi, "An efficient reconfigurable architecture by characterizing most frequent logic functions," *Field Programmable Logic and Applications (FPL)*, 2015.

[11] Z. Zilic and Z. Vranesic, "Using decision diagrams to design ULMs for FPGAs," *IEEE Transactions on Computers*, vol. 47, no. 9, pp. 971–982, 1998.

[12] A. Ahari, B. Khaleghi, Z. Ebrahimi, H. Asadi, and M. B. Tahoori, "Towards dark silicon era in FPGAs using complementary hard logic design," *Field Programmable Logic and Applications (FPL)*, pp. 1–6, 2014.

[13] V. P. Correia and A. I. Reis, "Classifying n-input Boolean functions," *VII Workshop Iberchip*, 2001.

[14] S. Kimura, T. Horiyama, M. Nakanishi, and H. Kajihara, "Folding of logic functions and its application to look up table compaction," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '02)*, pp. 694–697, San Jose, CA, USA, 2002.

[15] J. Meyer and F. Kocan, "Sharing of SRAM tables among NPN-equivalent LUTs in SRAM-based FPGAs," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 15, p. 15, 2007.

[16] Q. Zhao, K. Yanagida, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "A logic cell architecture exploiting the shannon expansion for the reduction of configuration memory," *Field Programmable Logic and Applications (FPL)*, pp. 1–6, 2014.

[17] J. H. Anderson and Q. Wang, "Area-efficient FPGA logic elements: Architecture and synthesis," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pp. 369–375, Yokohama, Japan, 2011.

[18] J. Luu, J. Goeders, M. Wainberg et al., "VTR 7.0: Next generation architecture and CAD system for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, 2014.

[19] G. Lemieux, E. Lee, M. Tom et al., "Directional and single-driver wires in FPGA interconnect," in *Proceedings of the International Conference on Field Programmable Technology (ICFPT)*, pp. 41–48, Brisbane, NSW, Australia, 2004.

[20] Z. Huang, L. Wang, Y. Nasikovskiy et al., "Fast Boolean matching based on NPN classification," in *Proceedings of the International Conference on Field Programmable Technology (ICFPT)*, pp. 310–313, Kyoto, Japan, 2013.

[21] A. Asghar, M. M. Iqbal, W. Ahmed et al., "Exploring shared SRAM tables among NPN equivalent large LUTs in SRAM-based FPGAs," in *International Conference on Field Programmable Technology (ICFPT)*, pp. 229–232, Xi'an, China, 2016.

[22] Berkeley Logic Synthesis and Verification Group, ABC: System for Sequential Synthesis and Verification, http://www-cad.eecs.berkeley.edu/alanmi/abc.

[23] Y. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 1999.

[24] Intelligent FPGA Architecture Repository (iFAR), http://www.eecg.toronto.edu/vpr/architectures.

[25] Predictive Technology Model (PTM), http://www.eas.asu.edu/ptm.

[26] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," *IEEE Transactions on Very Large Scale Integration Systems (VLSI)*, vol. 12, no. 3, 2004.

[27] Xilinx Inc., Programmable Logic Data Book, 1996.