# Balancing Logic Utilization and Area Efficiency in FPGAs

Russell Tessier and Heather Giza

Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA. 01003.
`tessier@ecs.umass.edu`

**Abstract.** In this paper we outline a procedure to determine appropriate partitioning of programmable logic and interconnect area to minimize overall device area across a broad range of benchmark circuits. To validate our design approach, FPGA layout tools which target devices with less that 100% logic capacity have been developed to augment existing approaches that target fully-utilized devices. These tools have been applied to FPGA and reconfigurable computing benchmarks which range from simple state machines to pipelined datapaths. In general, it is shown that the minimum *area* point for architectures similar to those available from Xilinx Corporation falls below the 100% *logic* utilization point for many circuits.

## 1 Introduction

Traditionally, the capacity of FPGA devices has been completely identified by the quantity of logic gates available inside the devices. In practice, however, it is accepted that 100% logic utilization of FPGAs is frequently impractical due to a limited supply of programmable routing resources. Clearly, the individual nature of a specific logic design defines the amount of interconnect needed to complete device routing. If the routing allocated to a device is at a high level relative to its available logic, unused *routing* area will be wasted and the design can be defined as *logic limited.* If the level of routing resources is at a low level relative to its available logic, the logic device will be *routing-limited*, thus requiring the user to select an FPGA with a larger amount of routing and logic resources in order to successfully complete place and route. Since the additional logic resources will likely be unused, this leads to wasted *logic* area. An area-efficient FPGA family can be designed by allocating routing resources to a given logic capacity so that area wastage across a collection of designs with similar amounts of logic is minimized and the mapping for most designs is *balanced.*

The issue of balancing FPGA resources to minimize area was first explored by Dehon in [10]. In this previous work, the interconnect of reconfigurable devices is modelled as a hierarchical binary tree with individual LUTs located at tree leaves. A collection of benchmark designs was applied to devices of varying interconnect richness and it was determined that for an FPGA family the

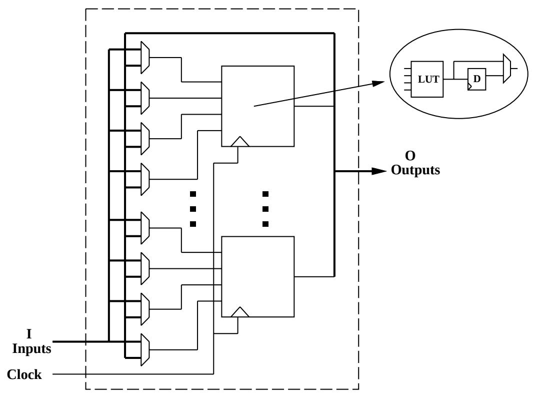

**Fig. 1.** Basic Logic Element and Logic Cluster

optimal area point supports 85% logic utilization across all designs. While this initial work clearly identifies area and logic utilization tradeoff issues, it has two significant limitations. In general, a binary tree is a limiting interconnect structure that leads to severe routing inefficiencies [16]. In this paper we consider lower-dimensional interconnect that more accurately reflects *island-style* routing structures found in XC4000 and Virtex devices from Xilinx Corporation [3]. Secondly, the previously-analyzed benchmarks are primarily oriented toward small state machines rather than the data paths commonly implemented in contemporary FPGAs. In this paper we consider a sampling of these benchmarks in conjunction with MCNC benchmarks and circuits from the RAW reconfigurable computing benchmark suite [5] to better represent the benchmark design space.

In performing our analysis of area efficiency it was often necessary to target designs to devices with less than 100% logic utilization. To perform this mapping accurately, new FPGA clustering and placement techniques were developed that specifically assume that some FPGA logic resources will be left unpopulated. It will be shown that these CAD techniques reduce overall required routing area by 40% versus previously-reported clustering and placement approaches when applied to designs mapped to devices with less than 100% logic utilization.

## 2 Background

In developing an area efficient FPGA design methodology, an effort has been made to match switch populations to existing commercial devices and to use available benchmark circuits from a spectrum of design suites.

### 2.1 Island-style FPGAs

While early FPGA architectures typically contained simple logic blocks containing one or two LUT/flip-flop pairs, more recent devices [2] [3] have grouped multiple LUT/FF pairs together into a single *cluster* to take advantage of design locality and to reduce FPGA place-and-route time. A key action in designing these architectural families has been the determination of logic cluster granularity. As previously described by Betz and Rose [7], if logic clusters contain insufficient logic resources, the amount of inter-cluster routing resources needed for routing will be great. Conversely, if clusters contain excessive amounts of logic, much of these resources will be wasted. Figure 1 shows a generalized model of a cluster-based FPGA device. Each cluster contains $N$ basic logic elements (BLEs), each possessing a single look-up table/flip-flop pair. The cluster has a total of $I$ inputs and $O$ outputs which connect cluster logic to the surrounding interconnection matrix. In [7] it was determined that the appropriate relationship between $N$ and $I$ is $I = 2N + 2$. To provide parallels to the Xilinx Virtex [3] architecture, a cluster size of $N = 4$ and cluster input count of $I = 10$ is used in experimentation. The routing structure of an *island-style* architecture is created by replicating a logic and routing *cell* in two dimensions to form a uniform, flat logic and routing substrate. The fraction of cluster I/Os that connect to tracks in each routing channel ($F_c = 0.3$) and the connectivity of each routing track to other tracks in a switchbox ($F_s = 3$) have been set to values determined by previous work [8].

Often, FPGA companies use the same logic and routing cell (with associated proportion of tracks per channel) to make numerous logic block arrays of differing logic block counts. If a logic design does not meet the routing constraints of a specific device in the family, it is often possible to meet routing constraints by migrating the design to a larger device in the family and leaving the added logic resources unused.

## 2.2 Design Requirements

For a given design, a known relationship exists between the amount of logic (or number of logic blocks) and the number of wires associated with the design. This relationship, Rent's Rule [12]:

$$\textbf{Rent's Rule}: \quad N = KG^p \tag{1}$$

where $N$ is the number of wires emanating from a region, $G$ is the number of circuit components (or logic blocks), $K$ is Rent's constant, and $p$ is Rent's exponent, characterizes the routing density in a circuit. Most circuits, except for linear arrays with primarily local communication, have been shown to have Rent exponents of $p > 0.5$ indicating that as a quantity of logic scales, the amount of interconnect emanating from it grows faster than its perimeter, which is directly proportional to $G^{0.5}$. As stated previously in [10], it is possible to characterize the relationship between a design and a target FPGA relative to their corresponding $p$ values assuming sufficient logic capacity is present in the FPGA. If $p_{interconnect} > p_{design}$ the design is effectively *logic limited* since some routing will be unused and if $p_{interconnect} < p_{design}$ the design is *routing limited* since some logic resources will have to be left unused in order for the design to route. Generally, FPGAs that have interconnect levels most closely aligned with the majority of target benchmarks will have the least area wastage.

While the Rent exponent $p$ of an FPGA based on a binary tree is generally easy to determine [10] given the centralized nature of hierarchical routing, the determination of $p$ for island-style arrays must be determined experimentally. In general, if a design with Rent exponent $p$ successfully routes on an FPGA device with no unused logic or routing resources (e.g. the design is balanced), the device may be characterized as having an interconnect capable of supporting other designs with Rent exponent $p$. It should be noted that full utilization of interconnect indicates that the device *track count* is the minimum needed for routing the array, not that every wire track is used. In Section 5 it will be shown that this procedure of calibrating the Rent exponent of given array sizes and track widths can be performed prior to experimentation to determine the capability of an array to route a specific group of designs (e.g. those with similar Rent exponents). It is interesting to note that while the *absolute* value of island-style track count relative to $p$ is difficult to determine with accuracy analytically, the *growth rate* of track count relative to logic block count in devices with Rent exponent $p$ can be determined analytically [14] through the use of average design wire length.

## 3 Related Work

With the exception of [9] and [10], most FPGA architectural evaluations [4] [8] have assumed that FPGA designers and consumers desire *full* device logic utilization for *all* designs, even at the cost of extreme amounts of routing area that is unused for *most* designs. As mentioned in [10], for most previous FPGA architectural evaluations, following assignment of design logic to programmable logic resources, device track counts and switch patterns are varied to find the lowest-cost solution from a routing area standpoint. In our evaluation, both routing *and* logic utilization are allowed to vary to permit a minimum overall area solution.

In [9], an FPGA architecture is described that allows logic blocks to be used either for logic or routing. While this approach allows for area tradeoffs, the fine-grained nature of the device architecture makes routing impractical for large, macro-based designs frequently implemented in practice today. As previously mentioned, in [10], a bifurcator-based binary tree model for routing is used to evaluate area utilization. While providing a flexible, scalable model for area experimentation, the bifurcator interconnect model is generally too restrictive for commercial development due to performance and locality limitations. A more complete discussion of previous work in FPGA architecture evaluation for island-style devices and others can be found in [8] and [10].

## 4 Experimental Methodology

In order to determine the area-minimizing ratio of logic to interconnect for an FPGA logic family, it is necessary to map a collection of benchmarks of approximately the same logic block count to a variety of FPGA arrays with varying logic block counts and interconnect richness (e.g. Rent exponents $p$). One mapping issue encountered in performing this evaluation was a lack of documented FPGA clustering and placement tools that can be applied to designs with less than 100% logic block utilization. Before describing our complete methodology, several novel approaches for mapping logic designs to FPGA devices with less than 100% utilization are described.

### 4.1 Clustering

A key aspect of mapping LUT-based designs to FPGAs with logic clusters is the process of clustering. In previous, full-utilization clustering approaches [7] [13], each cluster is packed as full as possible in an attempt to reduce the overall number of required device clusters. In our new approach, an attempt is made to *spread the logic evenly across all clusters in the device* while limiting the number of inputs that logically drive each cluster to be less than the number of pins physically available. The motivation for this approach is apparent if one considers the need for subsequent routing. Since each cluster input can drive any LUT input, underassigning logical inputs to physical cluster input pins gives a router much more flexibility in routing wires. For example, consider a cluster that has ten input pins, but only six that are to be used. The six logical inputs can be assigned to any of the ten available pins. If all ten inputs needed to be used, the number of possible input pin permutations would be greatly reduced. Another advantage of the modified clustering approach is that it distributes logic evenly across the chip. Generally, this helps the router avoid routing hot spots.

In the new clustering algorithm, the number of LUTs to be held in each cluster ($N_{high}$, $N_{low}$) is first determined. These utilization numbers reflect the overall LUT utilization of the device and differ by only one LUT. Following this step the number of device clusters that hold each quantity of LUTs ($C_{high}$ and $C_{low}$, respectively) is determined. Clustering is then performed for the two types of clusters with cluster inputs in each case, $I_{cluster}$, set to limit cluster fanin. Additional details about the clustering algorithm can be found in [15].

### 4.2 Placement

Simulated annealing is by far the most popular placement algorithm used for FPGAs [8]. This hill-climbing approach requires a cost function that closely models the costs likely to be faced during a subsequent routing phase. Prior to developing new clustering techniques, several new placement techniques were tested for use in conjunction with the original greedy clustering algorithm described in [7]. For designs targeted to devices with less than 100% logic utilization, greedy clustering leads to a number of both fully-populated clusters and some clusters that are completely empty. Both of the following placement techniques attempt to distribute the empty clusters inside the target device to minimize routing congestion. Each technique involves the use of a modified simulated annealing cost function that has been augmented to include costs in addition to wire length.

**Bin Utilization.** While wire length minimization has been shown to be an effective technique for promoting routability in fully-populated designs, the use of wire length alone in partially-populated devices can lead to routing congestion in one area of the device while other areas are completely empty. A way to overcome congestion in specific regions of the device is to penalize local region congestion in the annealing cost function.

To promote congestion-free routing, a bin-based placer was developed that considers the device as a collection of placement regions. Prior to placement, an *occupancy limit* for each region is set to be the percentage of populated clusters in the entire device. The net effect of a utilization factor is that populated clusters are spread evenly throughout the device. If the population of a bin exceeds the occupancy limit at a given time, a penalty factor is added to the annealing wire length cost function.

The efficiency of the placement approach is directly related to the size of the bin used. If the bin size is too small, not only is computation time increased, but also overall placement wirelength may be adversely affected. If the bin size is too large, the benefits of binning may be reduced. We have found that a bin size containing approximately 25-36 clusters leads to the best placement results.

**Non-linear Congestion.** An alternate binning approach, first described in [6], abandons the wire length cost model for simulated annealing in favor of a cost model based on wiring track demand within specific bins. In this case, the cost of the logic in each bin is characterized as:

$$Max\left(\frac{D_x}{\rho * S_x}, \left(\frac{D_x}{\rho * S_x}\right)^2\right) + Max\left(\frac{D_y}{\rho * S_y}, \left(\frac{D_y}{\rho * S_y}\right)^2\right) \qquad (2)$$

where $D_x$ ($D_y$) is the demand for horizontal (vertical) routing tracks within the region, $S_x$ ($S_y$) is the supply of available horizontal (vertical) tracks within the region and $\rho$ is a scaling factor. This approach is much more time consuming than the utilization-based binning approach since demand values must be updated by examining the routing bounding box of each net affected by a potential block move.

## 4.3 Experimental Procedure

To demonstrate how an area-efficient FPGA family can be determined, the ten benchmarks listed in Table 4.3 were mapped to a set of island-style FPGAs of various logic block counts and channel densities. The Rent exponents listed in Table 4.3 were determined through recursive bipartitioning using a KLFM

| Circuit | Source | LUTs | Rent Exp. ($p$) |
|---|---|---|---|
| switch | Ind | 1860 | .62 |
| r4000 | PREP | 1825 | .65 |
| alu4 | IWLS93 | 1522 | .63 |
| apex2 | IWLS93 | 1878 | .70 |
| ssp16 | RAW | 1798 | .54 |
| bsort | RAW | 1653 | .48 |
| spm4 | RAW | 1892 | .52 |
| bigkey | MCNC | 1707 | .56 |
| des | MCNC | 1591 | .57 |
| seq | MCNC | 1750 | .65 |

**Table 1.** Benchmark Design Statistics

mincut partitioner [11]. These similarly-sized benchmarks were taken from the MCNC benchmark suite [17], the RAW benchmark suite [5], and the PREP FPGA benchmark suite [1]. One design, a small network switch, was obtained from a commercial company.

The following steps were performed to determine the appropriate amount of routing tracks for a logic family to achieve minimum area across all benchmarks:

1. The Rent parameters of grids containing 22x22 logic blocks (clusters) were determined for assorted channel track counts using the procedure listed in Section 2.2. Since all benchmarks achieve exactly or nearly 100% logic utilization at this logic block count, the largest track count required for a grid of this size represents an upper area bound for these benchmarks.
2. Designs were mapped to grids with additional logic blocks compared to those used in step 1. Mapping was performed using the clustering and placing approaches outlined in Section 4. In many cases a design mapped to a larger logic grid required a lower track count to route successfully, thus requiring less routing area at the cost of additional logic area.
3. The minimum area point across all designs, track counts, and logic array sizes was determined through area summation of both logic block and routing switch transistor counts.

The VPR tool set [8] was used to perform all design routing with routing segment length distributions the same as those found in Xilinx Virtex devices. The **trans_count** tool from the University of Toronto was used to evaluate island-style FPGA area. As mentioned in Section 2, all clusters were assumed to contain four LUTs.

## 5 Results

The first step in the design procedure was to determine which of the clustering and placement algorithms were best suited to mapping designs to FPGAs with

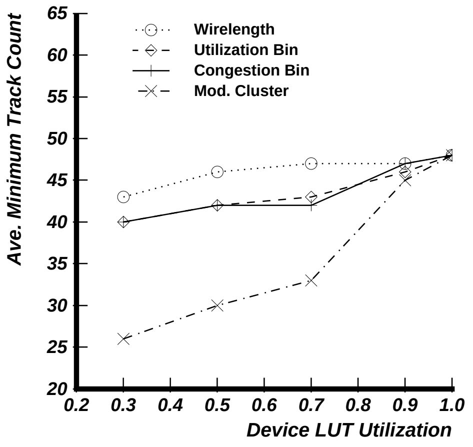

**Fig. 2.** Ave. minimum tracks based on device utilization

less than 100% logic utilization. All techniques, except the modified clustering approach described in Section 4, used the greedy clusterer described in [7]. Clusters created by the modified clustering algorithm were placed in the FPGA using simulated annealing based solely on wire length minimization. As can be seen in Figure 2, the modified clustering approach was most effective at distributing logic around the device for a variety of LUT logic utilization levels. Even though in many cases the total number of inter-cluster nets that needed to be routed *increased*, the additional routing flexibility obtained through reduced LUTs and inputs per cluster helped achieve lower-area placements. While utilization and congestion bin-based costs performed better than wire-length only based cost, the improvements were minimal.

The second part of the design analysis was to determine the number of clusters required per device to map the benchmark designs to devices with a fixed number of tracks per channel. As seen on the left side of Figure 3, as the number of available tracks per channel was reduced, the number of clusters required per device increased dramatically indicating low logic utilization per device for these cases. On the right side of the graph it can be seen that beyond a certain interconnect point (about 50 tracks per channel) adding extra tracks does not reduce average cluster count. This point indicates 100% routability for all benchmark designs. The dashed lines in the graph will be explained below.

The final step in the analysis was to add the transistor area consumed by logic clusters and routing for the design curves illustrated cumulatively in Figure 3 to determine the area for designs mapped to various levels of logic and interconnect. As shown in Figure 4, the minimum area point for all designs occurred for a track count of about 38. From experimentation, this value roughly corresponds to a Rent exponent $p$ of approximately 0.55. In Figure 3 it can be seen that a track count of 38 corresponds roughly to an average logic utilization of about 80%.

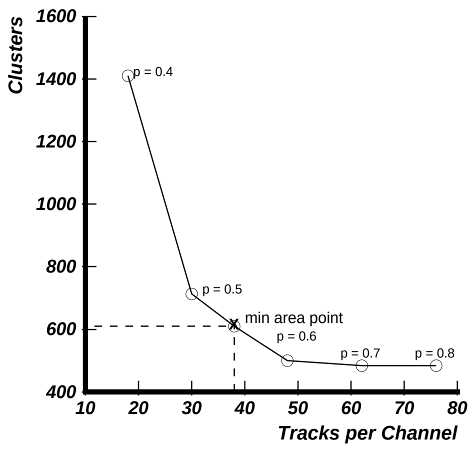

**Fig. 3.** Ave. clusters required for logic devices versus track counts

Two observations can be noted. First, a Rent exponent of 0.55 is in the range of the Rent values of the benchmarks, as one would expect. Secondly, the *p* value of 0.55 and utilization value of 80% are close to the 0.6 and 85% found by Dehon for the binary tree model [10].

## 6 Conclusions

In this paper we have outlined a procedure by which an area-efficient FPGA family can be designed. By evaluating a series of benchmark circuits, it is possible to determine routing track counts that will lead to reduced overall logic *and* routing area across all designs. An important step in this work was the analysis of several clustering and placement approaches to promote routability in FPGA designs with less than 100% logic utilization. While improved clustering techniques were found to be highly effective in reducing routing area, bin-based placement approaches were found to be less effective.

## References

1. *Prep Benchmark Suite*. www.prep.org, 1999.
2. *Altera Data Book*. Altera Corporation, 2000.
3. *Xilinx Corporation*. www.xilinx.com, 2000.
4. A. Agarwal and D. Lewis. Routing Architectures for Hierarchical Field Programmable Gate Arrays. In *Proceedings IEEE International Conference on Computer Design*, Oct. 1994.
5. J. Babb, M. Frank, V. Lee, E. Waingold, and R. Barua. The RAW Benchmark Suite: Computation Structures for General Purpose Computing. In *Proceedings, IEEE Workshop on FPGA-based Custom Computing Machines*, Napa, Ca, Apr. 1997.

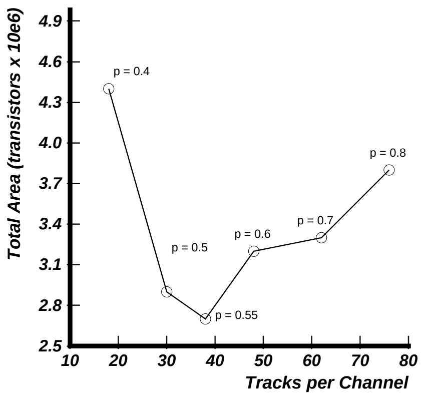

**Fig. 4.** Combined logic and routing area required for logic devices versus track counts

6. V. Betz and J. Rose. On Biased and Non-Uniform Global Routing Architectures and CAD Tools for FPGAs. *University of Toronto Department of Electrical Engineering, Technical Report*, June 1996.
7. V. Betz and J. Rose. Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size. In *Proceedings, IEEE Custom Integrated Circuits Conference*, pages 551–554, 1997.
8. V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs.* Kluwer Academic Publishers, Boston, Ma, 1999.
9. G. Borriello, C. Ebeling, S. Hauck, and S. Burns. The Triptych FPGA Architecture. *IEEE Transactions on VLSI*, pages 491–501, Dec. 1995.
10. A. Dehon. Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization). In *7th International Workshop on Field-Programmable Gate Arrays*, Monterey, Ca, Feb. 1999.
11. C. M. Fiduccia and R. M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Design Automation Conference*, May 1984.
12. B. Landman and R. Russo. On a Pin Versus Block Relationship For Partitions of Logic Graphs. *IEEE Transactions on Computers*, C-20(12), Dec. 1971.
13. A. Marquardt, V. Betz, and J. Rose. Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density. In *International Symposium on Field Programmable Gate Arrays*, Monterey, Ca., Feb. 1998.
14. R. Tessier. *Fast Place and Route Approaches for FPGAs: Chapter 6.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1999. available at www.ecs.umass.edu/ece/tessier/tessier.html.
15. R. Tessier and H. Giza. Balancing Logic Utilization and Area Efficiency in FPGAs. *University of Massachusetts Department of Electrical and Computer Engineering, Technical Report TR-CSE-00-5*, June 2000.
16. W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. Dehon. HSRA: High-Speed Synchronous Reconfigurable Array. In *7th International Workshop on Field-Programmable Gate Arrays*, Monterey, Ca, Feb. 1999.
17. S. Yang. Logic Synthesis and Optimization Benchmarks. *Microelectronics Centre of North Carolina Tech. Report*, 1991.