Designing Efficient Input Interconnect Blocks for LUT Clusters Using Counting and Entropy

WENYI FENG and SINAN KAPTANOGLU
Actel Corporation

In a cluster-based FPGA, the interconnect from external routing tracks and cluster feedbacks to the LUT inputs consumes significant area, and no consensus has emerged among different implementations (e.g., 1-level or 2-level). In this paper, we model this interconnect as a unified input interconnect block (IIB). We identify three types of IIBs and develop general combinatorial techniques to count the number of distinct functional configurations for them. We use entropy, defined as the logarithm of this count, to estimate an IIB's routing flexibility. This enables us to analytically evaluate different IIBs without the customary time-consuming place and route experiments. We show that both depopulated 1-level IIBs and VPR-style 2-level IIBs achieve high routing flexibility but lack area efficiency. We propose a novel class of highly efficient, yet still simple, IIBs that use substantially fewer switches with only a small degradation in routing flexibility. Experimental results verify the routability of these IIBs, and confirm that entropy is a good predictor of routability.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Style—*gate arrays*

General Terms: Design, Theory

Additional Key Words and Phrases: FPGAs, PLDs, counting, entropy, interconnect, LUT, cluster

ACM Reference Format:
Feng, W. and Kaptanoglu, S. 2008. Designing efficient input interconnect blocks for LUT cluster using counting and entropy. ACM Trans. Reconfig. Techn. Syst. 1, 1, Article 6 (March 2008), 28 pages. DOI = 10.1145/1331897.1331902. http://doi.acm.org/10.1145/1331897.1331902.

1. INTRODUCTION

Up to 90% of a Programmable Logic Device (PLD) chip is occupied by the programmable interconnect, including wires, switches and configuration bits. Due to their high complexity, routing architectures are usually designed through extensive experimentation.

Authors' address: Actel Corporation, 2061 Stierlin Court, Mountain View, CA 94043, email: {wenyi.feng; sinan.kaptanoglu}@actel.com.

An experimental approach evaluates a routing architecture by running place and route experiments with a suite of benchmark designs. This is an effective and maybe the ultimate way to verify a routing architecture. In academia, VPR [Betz et al. 1999] has been widely used for this purpose because it provides both a high-quality CAD flow and a parameterized routing architecture generator, which are convenient for architecture studies. However, the experimental approach can be time-consuming and the results obtained are limited by the CAD tool and the benchmark suite.

On the other hand, there is a lack of theoretical approaches to this problem, as pointed out in Hutton [2001].

This article proposes an analytical approach to evaluate and design a specific routing sub-structure to connect signals to LUT inputs, which we shall refer to as an input interconnect block (IIB) hereafter. The motivation of our approach is best described by a small example in Figure 1, which shows three different ways to connect eight routing tracks to two 2-LUTs. We might ask:

—Can we compare their routability without creating a complete routing architecture and running experiments? It is obvious that A is going to be better than B and C since A is a full crossbar. But which of the more economical alternatives, B and C, is better?

—Can we quantitatively predict the impact of these alternative IIBs on the usage of external routing resources (routing resources outside the IIB)? For instance, it is apparent that if we substitute B or C in place of A in a complete routing architecture, more external routing resources will likely be required to finish routing due to the reduced flexibility in the IIB. But can we predict how much more?

Previous work cannot address these questions. We answer them using a counting and entropy approach. We show that B and C have 312 and 256 distinct functional configurations respectively; hence, B is likely to be more routable than C. We also demonstrate that we can use entropy, defined as the logarithm of the number of distinct functional configurations, to quantitatively predict the impact of the IIB on external routing resource usage.

1.1 Review of Counting and Entropy Approach

The counting and entropy approach has been used to obtain theoretical lower bounds in switching interconnection networks. A classical example is a telephone switching network that is able to connect n inputs to n outputs in any permutation. Such a network contains at least $n!$ states, which leads to a lower bound of $\Omega(n\log n)$ on switches [Shannon 1950]. DeHon [1996] proposed a similar metric to characterize a programmable interconnect—a count of the number of distinct functional configurations. For example, a programmable interconnect which provides all n^{nk} possible interconnect configurations among n k-LUTs requires $\Omega(n\log n)$ configuration bits. In Feng and Greene [2006], the bound is reduced to $\Omega(n)$ when only configurations corresponding to good placement are taken into account.

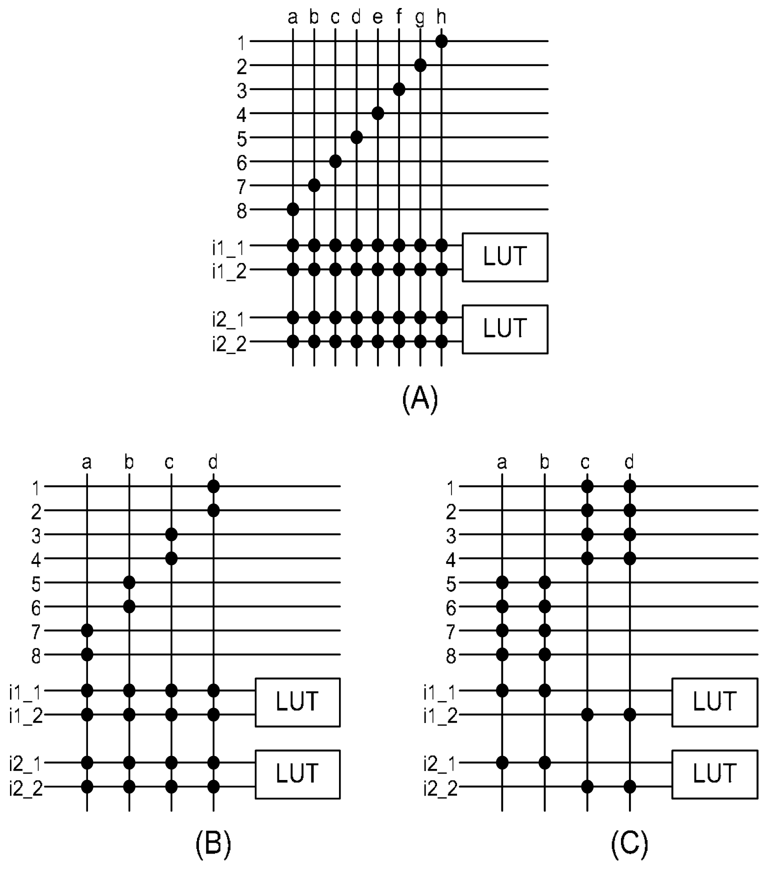


Fig. 1. Three ways to connect 8 routing tracks to 2 2-LUTs. (A) a full crossbar from 8 tracks to 4 LUT inputs; (B) 8 tracks drive 4 2-1 muxes first, then there is a full crossbar from 4 mux outputs to 4 LUT inputs; (C) 8 tracks drive 4 4-1 muxes first, then there is a 50% populated crossbar from 4 mux outputs to 4 LUT inputs. A, B and C have 784, 312 and 256 distinct functional configurations respectively.

When details of the interconnections are specified, the counting approach can be used to obtain the number of distinct functional configurations provided. Such a count can be used to predict routability. However, this approach is rarely used because counting is generally difficult except for certain trivial primitives [DeHon 1996]. One successful application of the approach on a non-trivial structure is presented in Chang et al. [1996]; They showed theoretically that a universal switch module has higher routing capacity than any other type of switch module, and verified experimentally its routability.

1.2 Review of IIB Design for LUT clusters

Modern FPGAs use clustering to improve the area and delay efficiency of the routing architecture. There exist different ways to connect signals to the LUT input muxes. In Xilinx Virtex architectures [Xilinx Inc. 2001], the routing tracks are connected directly to the input muxes (1-level IIB). In the VPR architecture [Betz et al. 1999] and the Altera Stratix architecture [Lewis et al. 2003], the routing tracks are connected to the input muxes via an intermediate level of muxes (2-level IIB).

Betz and Rose studied the VPR-style IIBs [Betz et al. 1999]. A VPR-style IIB has a sparsely populated first-level crossbar and a fully populated second-level crossbar (or intra-cluster crossbar). The fully populated intra-cluster crossbar is simple but takes no advantage of the logical equivalency of LUT inputs, a significant inefficiency. (According to Lemieux and Lewis [2004], the second-level crossbar alone can consume 24 to 33% of the transistor area in a cluster.)

Lemieux and Lewis [2004] improved the basic VPR-style IIB in two ways. They proposed an approach to generate highly routable sparse crossbars which can be applied to the first-level crossbar design in the VPR-style IIB. Furthermore, they showed that the intra-cluster full crossbar can be depopulated to achieve significant area reduction without performance degradation. A practical example is Stratix, which depopulates this crossbar by 50% [Lewis et al. 2003].

All such studies are mostly experimental, and they consider the first-level crossbar and the second-level crossbar separately. The reason is that to consider both crossbars simultaneously requires a much bigger solution space—too big to be handled efficiently by experimental approaches. Therefore, it is possible that some superior solutions are overlooked. Lemieux and Lewis [2004] noted this and suggested that future work should investigate joint optimization of both crossbars.

1.3 Contributions of This Work

This article extends the counting and entropy approach to study the IIB design problem. For 2-level IIBs, our approach naturally considers both crossbars simultaneously. The specific contributions of this article are as follows:

(1) We apply entropy, defined as the logarithm of the number of distinct functional configurations, to measuring the routing flexibility of an IIB. In addition, we provide theoretical arguments why there would be a linear tradeoff between an IIB's entropy and external routing resource usage (Section 2). We experimentally confirm this relationship (Section 5).

(2) We define three types of IIBs: 1-level IIBs, 2-level IIBs with fully populated second-level crossbars, and 2-level IIBs composed of disjoint sub-IIBs (Section 2). We develop general techniques to count the distinct functional configurations of such IIBs (Section 3).

(3) We propose a new class of efficient IIBs (Section 4). Compared to a depopulated 1-level IIB or a VPR-style 2-level IIB, these IIBs provide slightly lower entropy but use significantly fewer switches. In addition, unlike VPR-style IIBs, they do not limit the number of distinct external signals that can be routed to the LUTs (see Section 2.4). Their routability is experimentally verified (Section 5).

The rest of the article is organized as follows. The next section gives preliminaries and definitions. Section 3 develops the general counting techniques. The new class of efficient IIBs is proposed in Section 4. Theoretical and experimental results are provided in Section 5, followed by concluding remarks in Section 6.

2. PRELIMINARIES

A cluster-based FPGA uses a cluster as its basic building block, which is repeated to form an array. We assume each cluster contains N basic logic elements (BLEs), where each BLE contains a k-input LUT and a register (see BLE #N in Figure 2). The k inputs of a LUT are logically equivalent. One minor modeling difference from a VPR-style BLE [Betz et al. 1999] is that our BLE does not include the 2-1 output selection mux. This enables us to model a slightly less restrictive BLE, which represents more closely what commercial vendors use [Xilinx Inc. 2001; Leventis et al. 2003]. When using it to model the VPR-style BLE, we will treat the 2-1 output mux as part of the routing fabric.

The routing fabric is assumed to be a network of interconnected muxes. A size S mux has S inputs and one output, and can be programmed[1] such that any one of its S inputs can drive its output. Such a mux is said to have S switches.

In this article, we break the whole routing fabric into two parts: the *input interconnect blocks* (*IIB*), which are used to route signals from routing tracks and feedbacks (i.e., BLE outputs in the same cluster) to LUT inputs; and the *external routing*, which is used to route signals from BLE outputs to their destination clusters (or specifically, the routing tracks that drive the destination clusters). The external routing is a single connected structure for the whole device, while there is one IIB for each cluster. We call the muxes in the external routing *external routing muxes*, and typically they are the muxes that drive the routing tracks.

An IIB has M inputs and kN outputs (Figure 2). The kN outputs drive the LUT inputs. We assume that all M inputs are distinguishable and labeled from 1 to M, and all N LUTs are also distinguishable and labeled from 1 to N. The equivalent to our IIB in a VPR-style cluster would include the input portion of the C-block, the N 2-1 muxes inside VPR-style BLEs and the whole intracluster crossbar [Betz et al. 1999; Lemieux and Lewis 2004].

2.1 Distinct Functional Configuration

What is a distinct functional configuration of an IIB? The key is to define the meaning of a distinct function. Since the function of an IIB is to route the input signals to the LUT inputs, it follows naturally that a distinct function can be described by which signals are routed to the inputs of each LUT.

[1]Such a mux may have anywhere from $\lceil \log(S) \rceil$ to S configuration bits to be programmed, depending on its encoding scheme.

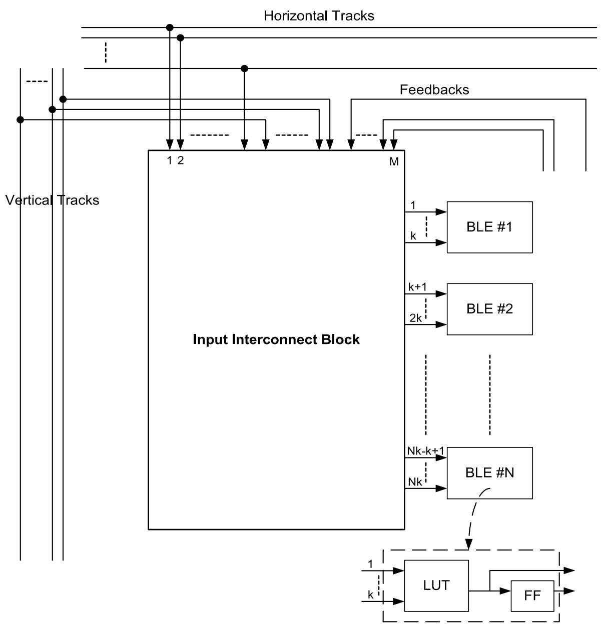


Fig. 2. Input interconnect block (IIB) with M inputs and kN outputs. The M inputs are from horizontal/vertical routing tracks as well as feedbacks. The routing tracks are driven by external routing muxes (not shown) outside the cluster. The feedbacks are driven by LUTs and FFs within the cluster. The k inputs to each LUT are logically equivalent.

Definition 1. *A routing requirement vector* (*RRV*) on the IIB is defined as an ordered list of N subsets of the M input signals available to the cluster. Each subset is a k-subset (containing exactly k distinct signals). The ith subset represents k signals to be routed to the ith LUT.

In this definition, we take LUT input equivalency into account. Each RRV represents a distinct function. The total number of RRVs for the IIB can be computed as $\binom{M}{k}^N$.

A configuration of an IIB is defined as a programming of all muxes in the IIB. In a configuration, each of the IIB's outputs is driven by one of its inputs. If, for each of the N LUTs, the k signals entering the LUT are distinct (i.e., comprise a k-subset), then we say the configuration realizes the corresponding RRV.

Definition 2. An RRV is said to be *routable* on an IIB, if there is at least one configuration of the IIB that realizes the RRV.

Therefore, counting the number of distinct functional configurations on an IIB is equivalent to counting the number of routable RRVs on the IIB.

It is quite possible that some RRV is not routable on an IIB. In fact, most IIBs discussed in this article only realize a tiny fraction of all possible RRVs. On the other hand, it is also very possible that some RRV can be realized by multiple configurations.

Example 1. For IIBs in Figure 1, ((1,5), (2,6)) is an RRV. This RRV requires router to route signals 1 and 5 to the first LUT, and signals 2 and 6 to the second LUT. For each LUT, we do not care about the exact input pin to which each signal is routed as long as both signals are routed to the LUT inputs since all LUT inputs are equivalent. For example, routing signal 1 to the first input (i1_1) and signal 5 to the second input (i1_2) is the same as routing signal 1 to i1_2 and signal 5 to i1_1. This RRV can be realized on IIB A and C, both by four configurations, but not on B. The four configurations on IIB A are (the signal paths are listed for each configuration, which can be easily translated into mux configuration): (1->h->i1_1, 5->d->i1_2, 2->g->i2_1, 6->c->i2_2), (1->h->i1_2, 5->d->i1_1, 2->g->i2_1, 6->c->i2_2), (1->h->i1_1, 5->d->i1_2, 2->g->i2_2, 6->c->i2_1), and (1->h->i1_2, 5->d->i1_1, 2->g->i2_2, 6->c->i2_1). The four configurations on IIB C are: (1->c->i1_2, 5->a->i1_1, 2->d->i2_2, 6->b->i2_1), (1->d->i1_2, 5->a->i1_1, 2->c->i2_2, 6->b->i2_1), (1->c->i1_2, 5->b->i1_1, 2->d->i2_2, 6->a->i2_1), and (1->d->i1_2, 5->b->i1_1, 2->c->i2_2, 6->a->i2_1).

2.2 Entropy

Definition 3. *The entropy of an IIB,* denoted as H, is defined as the base 2 logarithm of the number of routable RRVs on the IIB, that is, $H = \log(|\text{routable RRV set}|)$.

Why entropy? Our definition follows naturally from the one used in physics: entropy is the logarithm of the number of microstates. And with regard to IIB routability, a microstate corresponds to a distinct functional configuration of the IIB. Why use the logarithm? In one sense it doesn't matter; an IIB with a higher number of distinct functional configurations is likely to be more routable. Scaling by the logarithm is convenient, however, for the following reasons:

(1) Entropy is a more manageable quantity than the raw number of routable RRVs, which is usually huge.
(2) Entropy represents the minimum number of configuration bits needed to program an IIB. Therefore, using entropy to measure an IIB's flexibility gives flexibility a concrete meaning.
(3) An IIB's entropy is inversely correlated with the usage of external routing muxes required to route a design, and this correlation is likely to be linear. Consider a programmable interconnect consisting of IIBs and external routing muxes. Following the argument in Feng and Greene [2006], when a router routes a design, it needs to program the routing resources such that a certain minimal amount of information (corresponding to the post-placement interconnect entropy of the design) is stored. The information can only be stored in the IIBs or the external routing muxes. If we replace an initial IIB with a less flexible one having smaller entropy, then the router will be forced to program more external routing muxes to compensate for the entropy loss in the IIB. We estimate that the number of external routing muxes used is proportional to the entropy they contribute.

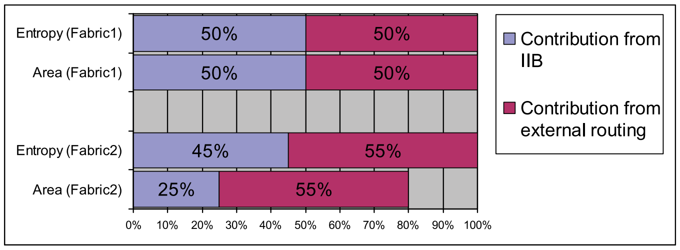


Fig. 3. How efficiently designed IIB can save routing area. In routing fabric 1, the external routing and the IIB contribute 50% each to both total entropy and total routing. In fabric 2, the IIB is improved to be half the area of the IIB in fabric 1 while still retain 90% of the entropy. To compensate for the entropy loss in IIB, the external routing in fabric 2 needs to increase 10% (to maintain the same level of routability). The net routing area saving from fabric 1 to fabric 2 is 20%.

Therefore, the entropy of the IIB should be linearly correlated to the usage of extra external routing muxes.

Given an IIB, if we can reduce its area but still keep the same amount of entropy (i.e., the same level of routability), then the new IIB is a pure improvement over the old one. However, it is often the case that when the IIB area is reduced, its entropy is also reduced. In such a case, only when the area decrease in the IIB exceeds the area increase in the corresponding external routing (which is required to make up for the entropy loss in the IIB), the whole routing fabric becomes a net win. Figure 3 uses an imaginary example to illustrate how an efficiently designed IIB can save total routing area.

From the example above, we also see that in addition to entropy of an IIB (used to measure its routing flexibility), we need another metric to measure its area efficiency. To obtain such a metric, first we use the total number of switches in an IIB to approximate its area; then we define *entropy per switch* of an IIB as its entropy divided by the total number of switches it uses (i.e., area). This metric is a measure of entropy per unit area, so it is a good metric of area efficiency. Also, the switch-count based area model allows us to ignore the underlying implementation details (e.g., SRAM or flash) in our discussion, while it is still accurate enough for us to do meaningful comparisons. Such an area model is commonly used in switching network study, and also used in Lemieux and Lewis [2004] for sparse crossbar designs.

Given the number of muxes and switches in an IIB, there are simple upper bounds on the IIB's entropy and entropy per switch.

THEOREM 1. *Let P and L be the number of muxes and switches in an IIB, respectively. Then the IIB's entropy and entropy per switch are at most $P \log (L/P)$ and $\log (L/P)/(L/P)$, respectively. The bounds are achieved only when each mux is of size L/P, and each configuration realizes a unique RRV.*[2]

[2]By extending the definition of a distinct functional configuration, the theorem applies to any network of interconnected muxes.

PROOF. Let the mux size be L_i for the ith mux. The total number of configurations is $\prod_{i=1}^{P} L_i$, which is an upper bound on the number of distinct functional configurations. So:

$$H \leq \log \prod_{i=1}^{P} L_i = \sum_{i=1}^{P} \log L_i \leq P \log \left(\left(\sum_{i=1}^{P} L_i \right) / P \right) = P \log(L/P) \qquad (1)$$

The second inequality is due to Jensen's inequality [Cover and Thomas 1991] for the concave function $\log(x)$, and the equality is achieved when all muxes are of the same size L/P. Dividing H by L, we also get the bound on entropy per switch. □

The function $f(x) = \log(x)/x$ (where x is an integer) is maximal at $x = 3$ and decreases thereafter. So the entropy per switch in any IIB is at most $\log(3)/3 = 0.528$. Also, the bigger the muxes are, the lower the bound on area efficiency goes.

2.3 Three Types of IIBs

An arbitrary IIB is too general for analysis. Instead, we will study three special types of IIBs. Figure 4 shows an example of each type.

In these drawings, we show two crossbars:

(1) *The first-level crossbar* has M inputs and I outputs, and is composed of I muxes (one per output). We call these muxes *L1-MUXes*.
(2) *The second-level crossbar* has I inputs and kN outputs, and is composed of kN muxes (one per output). We call these muxes *L2-MUXes*. (In the literature, these are also called *LUT IMUXes* since they drive LUT inputs; and this crossbar is also referred to as the *intra-cluster crossbar*.)

Each dot in a crossbar represents a switch. The size of a mux is equal to the number of dots lying on its corresponding output line in the crossbar.

For purposes of consistency, we draw a 1-level IIB (Figure 4(A)) the same way as a 2-level IIB, but with each L1-MUX only having one input (dummy mux). These M dots are not counted toward the total number of switches.

2.3.1 *Type-1 IIB*.

A type-1 IIB is a 1-level IIB, that is, the L2-MUXes are directly driven by the IIB input signals. It is an M input, kN output crossbar, and could be fully populated or sparsely populated.

Figure 4(A) shows a depopulated type-1 IIB.

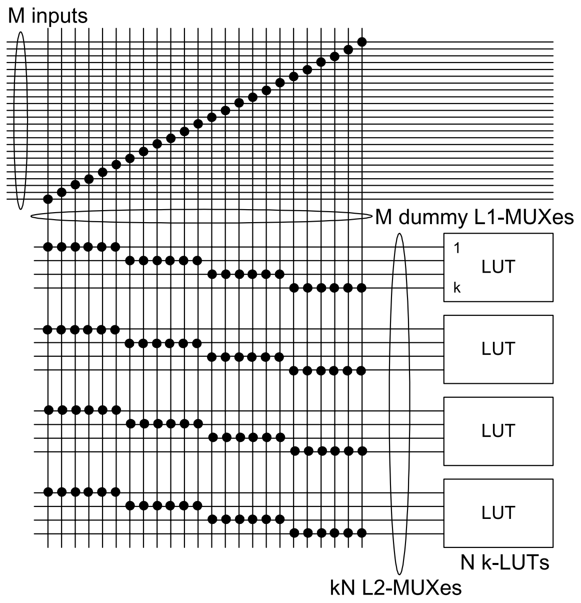
The Virtex-style IIB is of this type.

2.3.2 *Type-2 IIB*.

A type-2 IIB is a 2-level IIB with a fully populated second-level crossbar. The first-level crossbar is usually sparsely populated.

Figure 4(B) shows a type-2 IIB.

The VPR-style IIB belongs to this type with $I = k(N+1)/2 + N$, where $k(N+1)/2$ is the number of L1-MUXes driven by external input signals, and the remaining N by feedback signals.



(A) An example of type-1 IIB

Fig. 4. Examples of three types of IIBs.

2.3.3 *Type-3 IIB*. A type-3 IIB refers to a special kind of depopulated 2-level IIBs that can be decomposed into disjoint sub-IIBs.

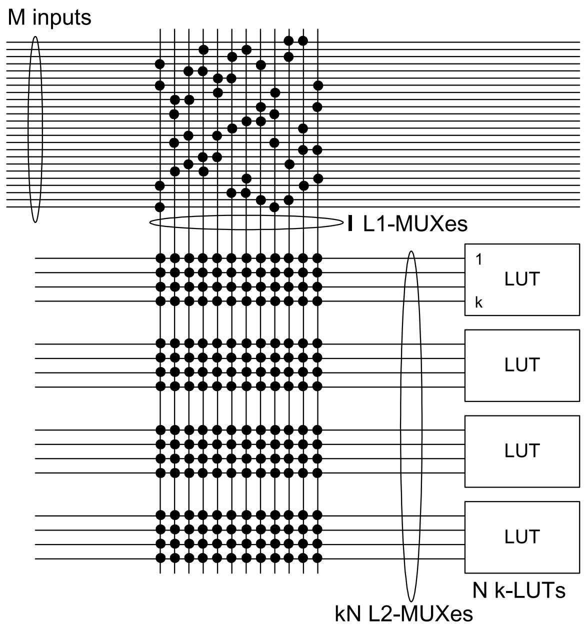
Definition 4. A 2-level IIB is a type-3 IIB if there is a partition (P_1, P_2, ..., P_C) of the L1-MUXes, and a partition (O_1, O_2, ..., O_C) of the L2-MUXes, such that

—L1-MUXes in P_i only drive L2-MUXes in O_i;
—Given any particular IIB input m and any particular LUT n, all paths (if any) between them must use muxes only in one set $P_i \cup O_i$.

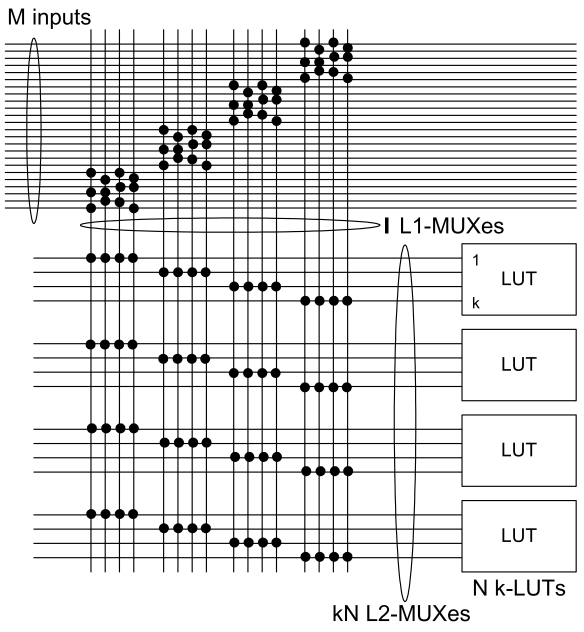
We also say that the 2-level IIB is composed of C disjoint sub-IIBs. The ith sub-IIB consists of P_i and O_i.
We point out two things following the above definition:

(1) The k L2-MUXes of a LUT may belong to different sub-IIBs; Figure 4(C) is such a case.
(2) Some of the M inputs may be shared by different sub-IIBs. For example, we can build an IIB by partitioning N LUTs into two groups, and building two independent 2-level sub-IIBs from M inputs to $kN/2$ L2-MUXes. Such an IIB is a type-3 IIB, and all M inputs are shared by both sub-IIBs.

Figure 4(C) shows a type-3 IIB. The whole IIB can be decomposed into k sub-IIBs, each with a set of its own inputs, a set of its own L1-MUXes and a



(B) An example of type-2 IIB



(C) An example of type-3 IIB

Fig. 4. *Continued*.

set of its own L2-MUXes (one from each LUT). Each sub-IIB is a type-2 IIB since the second-level crossbar of each sub-IIB is fully populated.

Not all 2-level IIBs with a depopulated second-level crossbar are type-3 IIBs. In particular, IIBs based solely on depopulating the intra-cluster crossbar of a VPR-style IIB, like those in Lemieux and Lewis [2004], are typically not type-3 IIBs. (Hence, such IIBs are not covered in this article.)

2.4 Input Bandwidth Limitation on IIBs

Input bandwidth is the maximal number of distinct external signals allowed to go into a cluster at the same time. This is a concept introduced in VPR-style architecture. Such a limitation is important for area reduction due to the full crossbar inside a VPR-style cluster. It is experimentally determined that $k(N+1)/2$ is typically enough to achieve 98% logic utilization [Betz et al. 1999; Ahmed and Rose 2004].

However, such a hard constraint is not at all natural and leads to its own problems. Certain commonly-used functions may require many unique signals going into a cluster. For example, a selective adder function (SEL? X+Y : X+Z), where SEL is a common signal, and X, Y and Z are all multiple bit signals, needs $3N+1$ unique signals per cluster if each 4-LUT can do one bit selective addition. With a $(2N+2)$ cluster input limitation, the burden is on the software to address such cases (e.g., by breaking the adder chain and underutilizing clusters).

The input bandwidth limitation is not an issue for 1-level IIBs [Betz et al. 1999].

3. COUNTING

In this section, we develop combinatorial techniques to count the number of distinct functional configurations for the three types of IIBs.

3.1 Type-1 IIB

This type is the easiest one because the M inputs directly drive LUT input muxes. The selection of input signals driving each LUT can be chosen independently from that of other LUTs. So the number of routable RRVs on the IIB is simply the product of the number of routable k-subsets for each LUT in isolation.

For a k-LUT, we can compute its routable k-subsets as follows. Let us use $x_1, x_2, \ldots, x_M$ to represent the M input signals of the IIB. For each LUT input mux, we write a summation formula with each input signal of the mux as a literal in the summation. Then we multiply all the k summation formulas together and expand it to a product-sum formula. The number of routable k-subsets is the number of unique product terms with k distinct literals. For example, for a 2-LUT with 2 input muxes driven by signals (1,2,4) and (1,3,4) respectively, we have $(x_1 + x_2 + x_4)(x_1 + x_3 + x_4) = x_1^2 + x_1x_3 + 2x_1x_4 + x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_4^2$. The number of unique product terms with 2 distinct literals is then 6, corresponding to 6 routable 2-subsets (1,2), (1,3), (1,4), (2,3), (2,4) and (3,4).

There are two special cases of type-1 IIBs:

—*Fully populated case.* This case has a fully populated crossbar between M inputs and kN outputs. It uses kNM switches and routes all $\binom{M}{k}^N$ RRVs. Figure 1(A) is such an IIB, and it routes all $\binom{8}{2}^2 = 784$ RRVs.

—*k-way depopulated case.* This case refers to Figure 4(A), in which both the M inputs and the kN L2-MUXes are partitioned evenly into k groups and each input group only drives one L2-MUX group. The L2-MUXes are partitioned in a way such that each group contains 1 mux from each LUT. This IIB uses NM switches and routes $(M/k)^{kN}$ RRVs. According to Theorem 1, this IIB is the most efficient among all type-1 IIBs with NM switches.

3.2 Type-2 IIB

Counting on this type is challenging due to the huge number of redundant configurations. Fortunately, because the second-level crossbar is fully populated, we can decouple the counting on the two crossbars.

THEOREM 2. *Let D_i be the number of unique i-subsets (of the M inputs) that can be routed on the first-level crossbar; let E_i be the number of ways to distribute i distinct signals to N distinct LUTs such that each LUT gets a k-subset (of the i signals) and each one of the i signals is used at least once by the LUTs. Then the total number of routable RRVs in the IIB is $\sum_{i=1}^{L} D_i E_i$.*

PROOF. We count routable RRVs separately according to the number of distinct signals they use. D_i is the number of distinct routable i-subsets (of the M inputs) on the first-level crossbar, and E_i is the number of ways to distribute these i signals to the LUTs on the second-level crossbar such that each one of the i signals is used by at least one LUT. Because the second-level crossbar is fully populated, the two crossbars are completely decoupled: any one of the D_i i-subsets from the first-level crossbar can be routed to the LUT inputs in E_i ways on the second-level crossbar. Therefore, the number of routable RRVs with i distinct signals is $D_i E_i$. Summing over all possible i values from 1 to I, we obtain the total number of routable RRVs. □

Now let us consider the computation of D_i and E_i respectively.

Computation of D_i. Because the first-level crossbar is not fully populated, only certain i-subsets can route on it. Hence, D_i depends not only on M and I, but also on the specific depopulation scheme. In principle, we can use the same multiplication and expansion approach (as in the previous section for LUT) to determine the D_is since all outputs of the first crossbar are equivalent. But such approach does not work for big I since the number of product terms grows exponentially with I. Lemieux and Lewis [2004] studied this problem, and there is no analytical solution for a general depopulated crossbar. Instead, they proposed to use Monte Carlo simulation to estimate the percentage of routable i-subsets. This approach is practical for the general case but may require long runtimes and can only produce an estimate. For certain special

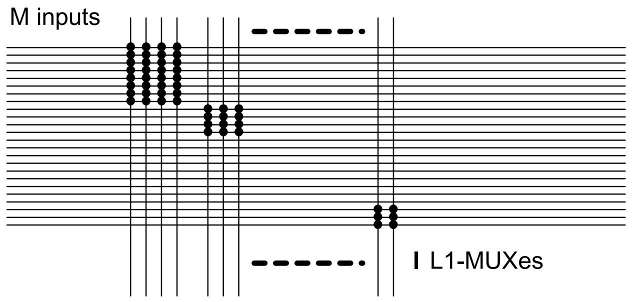


Fig. 5. A first-level crossbar composed of disjoint full crossbars.

cases, however, we can compute D_i exactly and conveniently using generating functions. A generating function [Brualdi 1998] for a sequence of numbers $(D_0, D_1, \ldots, D_i, \ldots)$ is a Taylor series with D_i as the coefficient of x^i:

$$D(x) = \sum D_i x^i \tag{2}$$

In our case, the D_i sequence is finite since D_i is 0 for $i > I$ by definition. Therefore, the generating function is a polynomial. Once we obtain the generating function, we obtain all D_is. The generating function of a full crossbar is straightforward. For example, for a 4-input, 4-output full crossbar, its generating function is $(1+x)^4 = 1+4x+6x^2+4x^3 + x^4$, since the number of 0 (1, 2, 3, 4) -subset out of the 4 inputs is 1 (4, 6, 4, 1) respectively. Similarly, the generating function of a 4-input, 2-output full crossbar is $(1+4x+6x^2)$.

The basic principle for our special non-full crossbar case computation is that if a crossbar can be partitioned into disjoint subcrossbars (i.e., there is no cross-connection among them), then its generating function is the multiplication of generating functions of all the subcrossbars. So D_is can be computed by the multiplication of polynomials. We describe two such special cases:

—*Special Case* 1. Each input drives just one mux. Let the mux sizes be $L_1, \ldots L_I$ respectively, then the generating function for the ith mux is $(1+L_i)$, and the generating function for the whole crossbar is $D(x) = \prod_{i=1}^{L}(1 + L_i x)$.

—*Special Case* 2. Inputs and outputs are partitioned into disjoint groups $(I_1, \ldots, I_T)$ and $(O_1, \ldots, O_T)$. There is a full crossbar from the inputs in I_i to the outputs in O_i. Figure 5 is an example. This is a more general form of case 1. The generating function for the ith sub-full-crossbar (with $|I_i|$ inputs, $|O_i|$ outputs) is $\sum_{j=0}^{|o_i|} \binom{|I_i|}{j} x^j$, hence, $D(x) = (\prod_{i=1}^{T} \sum_{j=0}^{|o_i|} \binom{|I_i|}{j} x^j)$.

We will use the above two special cases to count RRVs for the IIBs in Figure 1, as well as our proposed IIBs and other various experimental IIBs later in the article.

Computation of E_i. Since the second-level crossbar is fully populated, E_i depends only on i, N and k, and can be written as $E(i,N,k)$. It can be computed using the following recursive relation.

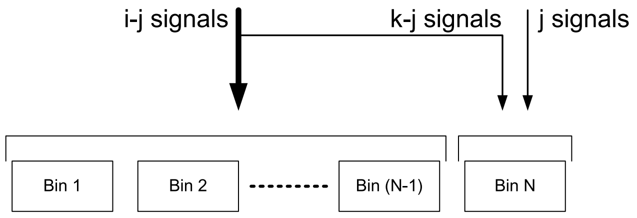


Fig. 6. A subcase in which the Nth bin uses j distinct signals and the first N-1 bins use $i - j$ distinct signals. The Nth bin also reuses $k - j$ signals out of $i - j$ signals used by the first N-1 bins to bring its signal total to k.

THEOREM 3.

$$E(i, N, k) = \begin{cases} \sum_{j=0}^{k} E(i-j, N-1, k) \binom{i}{j} \binom{i-j}{k-j}, & if\ N > 1 \\ 1, & if\ N = 1\ and\ i = k \\ 0, & if\ N = 1\ and\ i \neq k \end{cases}$$

PROOF. $E(i,N,k)$ can be seen as the number of ways to distribute i distinct signals to N distinct bins (LUTs) such that each bin gets k out of i signals (without repetition, order does not matter) and each of the i signals is used at least once by the bins. So when N is 1, $E(i,1,k)$ can be 1 only when $k = i$. This proves the case of N= 1.

In the case of $N > 1$, let us suppose we have obtained all $E(a,N\text{-}1,k)$ values for a = 1 to i. We can compute $E(i,N,k)$ by subcases according to the number of distinct signals (let it be j) used only by the Nth bin, see Figure 6. If the Nth bin uses j distinct signals, then the first N-1 bins use $i - j$. The value of j can vary from 0 (corresponding to the case that no distinct signal is used by the Nth bin) to k (corresponding to the case that k distinct signals are used by the Nth bin). For a specific j, the k signals used by the Nth bin consist of 2 parts: j distinct signals not used by the first N-1 bins, and $k - j$ signals used by the first N-1 bins. We can count the number of ways corresponding to j as follows: (1) we pick j out of i distinct signals and put them into the Nth bin; (2) we distribute the rest $(i - j)$ signals to the first N-1 bins; (3) we pick $(k - j)$ signals from those $(i - j)$ signals used by the first N-1 bins and give them to the Nth bin to bring its total number of signals to k. There are $\binom{i}{j}$ ways to do the first step, $E(i - j,N - 1,k)$ ways to do the second step, and $\binom{i-j}{k-j}$ ways to do the third step. Hence, $E(i - j,N - 1,k)\binom{i}{j}\binom{i-j}{k-j}$ is the number of ways to do the distribution such that the Nth bin uses j distinct signals and the rest use $(i - j)$ signals. Summing over all possible j values from 0 to k, we obtain the recursive relation. □

Note. When k = 1, the recursive relation becomes $E(i,N,1) = i{*}(E(i,N\text{-}1,1)+E(i\text{-}1,N\text{-}1,1))$, and can be solved as $E(i,N,1) = i!\ * S(N,i)$ where $S(N,i)$ is known as *Sterling's number of the second kind* [Brualdi 1998]. $S(N,i)$ represents the number of ways of partitioning N distinguishable elements into i indistinguishable non-empty sets.

Example 2. Figure 1(B) is a type-2 IIB. With the approach described above, we can compute the number of routable RRVs.

D_i computation. The first stage belongs to Special Case 1 for D_i computation, and has four disjoint 2-1 muxes. Therefore, $D(x) = (1+2x)^4 = 1+8x+24x^2+32x^3 + 16x^4$, and D_1=8, D_2=24, D_3=32, D_4=16.

E_i computation. (the step-by-step details are omitted)

—E_1 = 0, since each LUT requires two unique signals.
—E_2 = 1, since the two signals have to be both used by each LUT.
—E_3 = 6. The six ways are (assuming three signals are (a,b,c)): ((a,b)(a,c)), ((a,c)(a,b)), ((a,b)(b,c)), ((b,c)(a,b)), ((a,c)(b,c)) and ((b,c)(a,c)). Here ((a,b)(a,c)) means that the first LUT selects a and b, and the second LUT selects a and c.
—E_4 = 6. The six ways are (assuming four signals are (a,b,c,d)): ((a,b)(c,d)), ((c,d)(a,b)), ((a,c)(b,d)), ((b,d)(a,c)), ((a,d)(b,c)) and ((b,c)(a,d)).

Therefore, the total number of routable RRVs is: (8x0 + 24x1 + 32x6 + 16x6) = 312.

3.3 Type-3 IIB

THEOREM 4. *The number of routable RRVs on a type-3 IIB is the product of the number of routable RRVs on each disjoint sub-IIB.*

PROOF. We shall prove that a composition of C routable sub-IIB RRVs (one from each sub-IIB) is also a distinct routable RRV on the whole IIB using the definition of the type-3 IIB.

First, because each sub-IIB has its own set of L1-MUXes and L2-MUXes, the programming of each sub-IIB is independent. Therefore, we can compose C routable sub-IIB RRVs together by simply programming each sub-IIB to realize its RRV.

Second, such a composition will produce a distinct routable RRV for the whole IIB. We need to prove: (a) the k inputs of any LUT form a k-subset in the programming; (b) a different composition produces a different routable RRV. Each sub-IIB contains a subset of L2-MUXes. Therefore, if we look at one LUT, its k inputs could be from one sub-IIB or from several sub-IIBs. In the former case, the k inputs form a k-subset according to the RRV definition. In the latter case, the signals provided in each sub-IIB are different (by the RRV definition), and no signals from two different IIBs should be the same according to the second requirement in the definition of type-3 IIB. So the k inputs of the LUT must be different. This proves (a). Furthermore, if a sub-IIB (say Y) is programmed to route a different RRV, then at least one of its driven LUT (say Z) must receive some new input(s). Such an input difference on Z cannot be eliminated by programming other sub-IIBs since those inputs can reach Z only through Y. Therefore, such difference results in a different RRV for the whole IIB. This proves (b).

Combining the above arguments, we prove the theorem. □

The above theorem enables use to decompose the counting problem of a type-3 IIB into several smaller counting problems. Also, since it is just a divide-and-conquer approach, counting on a type-3 IIB relies on counting on each sub-IIB.

Example 3. Figure 1(C) is a type-3 IIB with two disjoint type-2 sub-IIBs. Each sub-IIB has M=4, I=2, N=2 and k=1.

The number of routable RRVs on each sub-IIB can be computed using the type-2 IIB counting approach. The first level is a 4X2 full crossbar, therefore $D(x) = (1 + 4x + 6x^2)$, and D_1=4, D_2=6. It is easy to see that E_1=1, E_2=2. So the number of routable RRVs on each sub-IIB is 4x1 + 6x2 = 16.

The number of routable RRVs on the whole IIB is therefore 16x16=256.

4. A NEW CLASS OF IIBS

Consider designing an IIB with M inputs and kN outputs. The goal is to build a highly flexible IIB with high area efficiency.

Type-1 IIBs typically lack area efficiency. Consider the k-way depopulated type-1 IIB. It uses MN switches and routes $(M/k)^{kN}$ RRVs. Such an IIB does not have any redundant configurations, but area efficiency is still low due to the big mux size M/k(area efficiency is a decreasing function of mux size).

Type-2 IIBs improve over type-1 IIBs due to the introduction of the L1-MUXes, which decouple the M inputs signals from the kN L2-MUXes. But the fully populated second-level crossbar severely reduces its efficiency.

This prompts us to propose a new class of IIBs. These IIBs are type-3 IIBs, which not only take advantage of LUT input equivalency, but also use L1-MUXes to greatly reduce the number of switches. Furthermore, these IIBs do not have a bandwidth limitation. Figure 7 shows pseudo code to build such IIBs, and Figure 8 shows an example. Each sub-IIB in the proposed IIB is a type-2 IIB, and the first-level crossbar of each sub-IIB belongs to special case 1 in the D_i computation. In the code, p is a parameter to control the tradeoff between routing flexibility and switch usage. As p decreases, the IIB uses fewer switches but also has lower routing flexibility. We do not allow p to go negative to guarantee that the IIB has no input bandwidth limitation.[3] For a given p, the IIB uses $M + k(N + p)N$ switches, and has entropy (as computed using Theorems 2, 3 and 4) of $k \log\left(\sum_{i=1}^{N}\binom{N+p}{i}\left(\frac{M}{k(N+p)}\right)^{i} i! S(N,i)\right)$, where $S(N,i)$ is Sterling's number of the second kind.

[3]If p is negative, then the number of L1-MUXes is $k(N+p)$, which is smaller than kN. This results in input bandwidth limitation. Furthermore, there is no simple way to check the input constraint (like that in VPR packing) due to the depopulation within the cluster [Lemieux and Lewis 2004].

```
      Input: $M$, $k$, $N$; and a non-negative integer $p$.

      Output: a type-3 IIB connecting $M$ inputs to $kN$ LUT
      inputs.

1.    Partition both $M$ inputs and $kN$ L2-MUXes into $k$
      groups. (See $k$-way depopulated type-1 IIB case)
2.    For $i = 1$ to $k$:
      /* create a type-2 IIB between each input group and
      L2-MUX group using ($N+p$) L1-MUXes */
      a.  Create $N+p$ L1-MUXes.
      b.  Evenly partition the $M/k$ inputs in the $i^{th}$ input
          group into $N+p$ sub-groups; connect each sub-
          group to an L1-MUX.
      c.  Create a full crossbar from the $N+p$ L1-MUX
          outputs to the $N$ muxes in the $i^{th}$ L2-MUX group.

/* The resulting IIB is a type-3 IIB with $k$ disjoint sub-IIBs.
Each sub-IIB is a type-2 IIB with $M/k$ inputs, $N+p$ L1-
MUXes and $N$ L2-MUXes. The total number of switches
used is $M+k(N+p)N$ */
```

Fig. 7. Proposed approach to build a 2-level IIB.

Input Partitioning. In Figure 7, the M inputs are first partitioned into k groups in step 1; each group is then further partitioned into $N + p$ subgroups in step 2.b. We classify the M inputs into different types according to the source of the inputs (e.g., the IIB in the next section has three input types: length-4 tracks, length-1 tracks and feedbacks). We require that the partitioning be done as evenly as possible in both steps: each partition gets not only the same (or almost the same) number of inputs, but also the same (or almost the same) number of inputs of each type. By doing so, we do not give any bias to any L2-MUX group (in step 1), or to any L1-MUX (in step 2.b). We believe this is best for routability. Also, feedback is a type by itself. This guarantees that the proposed IIBs do not have input bandwidth limitation, since the number of L1-MUXes is at least kN and none of them would be exclusively driven by the feedbacks.

The proposed IIBs can be considered as an extension of both a k-way depopulated type-1 IIB and a VPR-style type-2 IIB:

(a) They can be seen as extending a k-way depopulated type-1 IIB by adding L1-MUXes between each input group and L2-MUX group.
(b) They can also be seen as extending a VPR-style type-2 IIB by depopulating both crossbars and relaxing the input bandwidth limitation.

It should also be noted that the proposed IIBs work better for big clusters (big N and M values). For small clusters, the k-way depopulated type-1 IIB could be the best (the area efficiency of such an IIB is $\log(M/k)/(M/k)$, and is high when M/k is small).

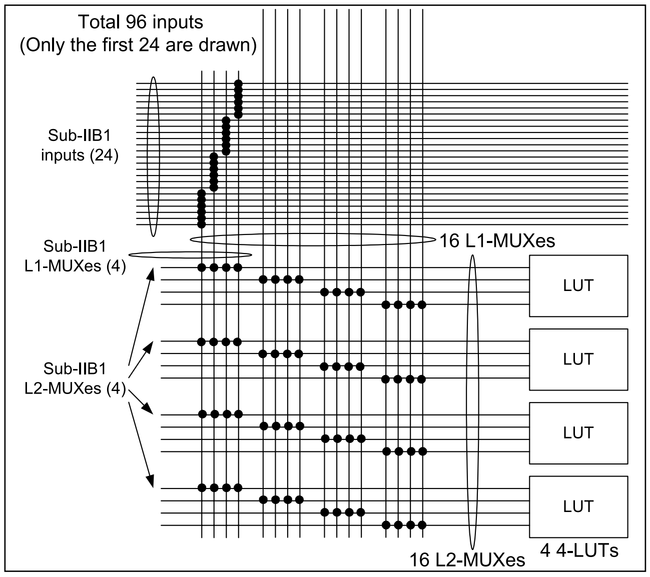


Fig. 8. A proposed IIB with M=96, N=4, k=4 and p=0. The number of L1-MUXes I=16. Only the first sub-IIB (M=24, I=4, N=4 and k=1) is completely shown.

5. RESULTS

To explore the effects of different IIBs on routing as well as to verify our entropy approach, we perform experiments using different IIBs in an Actel internal experimental FPGA architecture. A cluster in the architecture has eight 4-LUTs. Each routing channel has 48 length-12 tracks, 64 length-4 tracks and 8 length-1 tracks. All tracks are unidirectional. The numbers of tracks and other details of the external routing are experimentally chosen based on an initial IIB to guarantee that benchmark designs (96 designs ranging from a few hundred to 60K LUTs) can be routed comfortably on the architecture. We use the same architecture for all designs. Each one of the adjacent length-4 and length-1 tracks connects to the IIB. Each cluster also has 16 feedbacks (one LUT output and one FF output from each BLE). Three types of inputs (length-4, length-1 and feedback) are classified accordingly and used during input partition as described in Section 4. The IIB has the following parameters: M=160 (64x2 + 8x2 + 16), k=4 and N=8.

5.1 IIBs

We build the new proposed IIBs with different p values for routing experiments. In addition, we also build some more IIBs to evaluate the correlation between IIB entropy and external routing resource utilization in a wider range. As the routing results will show, these IIBs span a very wide range in term of IIB flexibility: from a barely routable IIB (LO-3) all the way to a full crossbar IIB (A). The IIBs used in routing experiments are summarized below.

—A: a fully populated 160X32 type-1 IIB.

—B: a 4-way depopulated type-1 IIB. It is also the initial IIB used to verify the routability of the routing architecture, and used as our comparison baseline.

—C, D, and E: three type-3 IIBs as proposed in Section 4 with p=12, 2 and 0, respectively.

—LO-1, LO-2, LO-3: Three IIBs with successively lower entropy than our proposed IIBs. They are built by using only 96 (out of 160 available) inputs with half of the length-4 inputs unused. LO-1 is similar to IIB B, and is a 4-way depopulated type-1 IIBs. Each LUT input mux size in LO-1 is 24. LO-2 and LO-3 are similar to IIB C, D and E, and are type-3 IIBs with four sub-IIBs. Each sub-IIB has 24 inputs and eight outputs. LO-2 has 12 L1-MUXes (i.e., p=4) and LO-3 has 8 L1-MUXes (i.e., p=0).

—HI-1, HI-2, HI-3, HI-4: four IIBs with successively higher entropy than our proposed IIBs.

  —HI-1: it is a type-3 IIB with 2 sub-IIBs. Each sub-IIB is a type-2 IIB with 80 inputs, 16 outputs (connecting to 2 inputs of each LUT), and 20 L1-MUXes. The first level of each sub-IIB is composed of 10 8X2 full crossbars.

  —HI-2: it is a type-2 IIB with 32 L1-MUXes. The first level is composed of 16 10X2 full crossbars.

  —HI-3: it is a type-2 IIB with 40 L1-MUXes. The first level is composed of 20 8X2 full crossbars.

  —HI-4: it is a type-2 IIB with 40 L1-MUXes. The first level is composed of 10 16X4 full crossbars.

These four IIBs are built in a way such that their entropy can be computed exactly using our counting approach, but they are not necessarily efficient. In particular, we build the first-level sparse crossbar out of disjoint smaller full crossbars. Such a crossbar is not as efficient as a randomly populated crossbar with the same number of switches. But since the purpose of these IIBs is to provide IIBs with higher entropy, their efficiency is secondary as long as their entropy can be computed exactly.

Two other IIBs are considered:

—VPR: a VPR-style type-2 IIB with 26 L1-MUXes (18 for external inputs, and 8 for feedbacks). The first-level crossbar is assumed to contain two disjoint parts: a 144$\rightarrow$18 crossbar with 3x144 = 432 switches for the routing tracks, and a 16$\rightarrow$8 crossbar composed of eight 2-1 muxes for the feedbacks (each BLE uses one 2-1 mux to feedback its FF or LUT output). Three fanouts per routing track is assumed for good routability. The second-level full crossbar uses 26x32 = 832 switches. So the total number of switches is 1280 (= 432+832+16).

—VLL: an improved IIB based on IIB VPR. Lemieux and Lewis proposed to improve the VPR-style IIB by providing spare cluster inputs and depopulating the second level crossbar. We assume two spare inputs are provided and the second level crossbar is 50% populated, consistent with what is reported in Lemieux and Lewis [2004]. As a result, the number of switches in the second level crossbar is reduced from 832 to 448 (=28x32/2). The number of

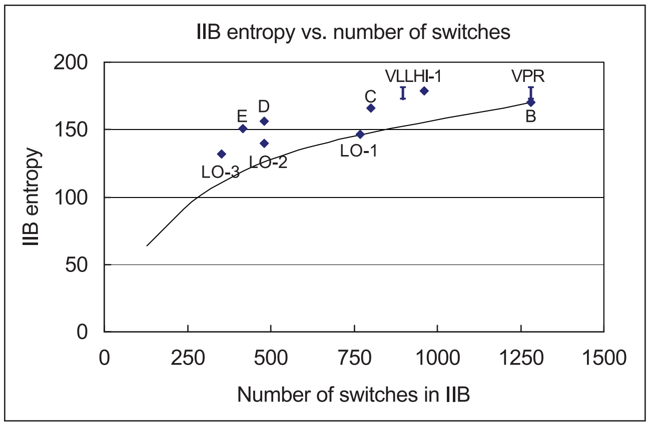


Fig. 9. IIB entropy vs. number of switches for IIBs with fewer than 1300 switches. The curve is for k-way depopulated type-1 IIBs when the IIB B is further depopulated by leaving some of the M inputs unused.

switches in the first level crossbar is assumed to be the same as that of IIB VPR. So the total number of switches is 896.

5.2 Theoretical Results: Comparison of Entropy and Switch Count

We compute the different results for all the IIBs and compare them in Table I and Figure 9. The VPR-style IIB has two numbers representing lower and upper bounds on its entropy (since we do not know how to compute D_i for its first-level crossbar). See the Appendix for how the upper and lower bounds are computed. We also assume the entropy for IIB VLL is unchanged from IIB VPR since it is shown that IIB VLL has similar routability as IIB VPR [Lemieux and Lewis 2004].

IIB A is the most inefficient one. It achieves 16% higher entropy than B but uses 3x more switches.

Compared to B, the proposed new IIBs (C, D and E) provide slightly less entropy but use significantly fewer switches. The smaller p is, the fewer switches are needed. From B to E, the number of switches is reduced by more than 2/3, while the overall entropy is reduced by only 11.6%. This shows that E is much more area efficient than B (as also indicated by entropy per switch, it is 0.362 for E and 0.133 for B).

Note that further depopulating a 1-level IIB (e.g., by leaving some of the M inputs unused) cannot achieve the same improvement as the proposed 2-level IIBs, as shown by the type-1 IIB curve in Figure 9. For example, LO-1 IIB is such a 1-level IIB with 64 inputs unused. It uses 768 switches and its entropy is 146.7, which is inferior to 2-level IIBs D and E in both switch count and overall entropy.

LO-2 and LO-3 are 2-level IIBs built by applying our proposed approach with a smaller set of inputs. They are better than IIB LO-1, their 1-level counterpart, in term of entropy per switch. However, LO-2 is not as good as IIB D: they both use 480 switches, but IIB D entropy per switch is 0.326, higher than

Table I. Comparison of different 160-input, 32-output IIBs. IIBs are sorted by entropy. LO-3, LO-2, and LO-1: 3 IIBs with lower entropy than proposed IIBs; E, D and C: proposed IIBs with p=0, 2 and 12 respectively; B: 4-way depopulated type-1 IIB; VPR: VPR-style type-2 IIB; VLL: improved IIB based on IIB VPR; HI-1, HI-2, HI-3, and HI-4: 4 IIBs with higher entropy than proposed IIBs; A: fully populated type-1 IIB. In the last column, the entropy values are normalized against B's.

IIB	Number of switches	Routable RRVs	IIB entropy	Entropy per switch	Normalized entropy
LO-3	352	5.46E+39	132.0	0.375	0.775
LO-2	480	1.38E+42	140.0	0.292	0.822
LO-1	768	1.47E+44	146.7	0.191	0.862
E	416	2.01E+45	150.5	0.362	0.884
D	480	1.26E+47	156.5	0.326	0.919
C	800	1.10E+50	166.2	0.208	0.976
B	1280	1.84E+51	170.3	0.133	1.000
VPR	1280	9.73E+51-3.04E+54	172.7-181.0	0.135-0.141	1.014-1.063
VLL	896	9.73E+51-3.04E+54	172.7-181.0	0.193-0.202	1.014-1.063
HI-1	960	6.83E+53	178.8	0.186	1.050
HI-2	1344	2.73E+55	184.2	0.137	1.081
HI-3	1600	1.10E+57	189.5	0.118	1.113
HI-4	1920	2.77E+58	194.1	0.101	1.140
A	5120	2.29E+59	197.2	0.039	1.158

0.292 for LO-2. IIB LO-3, despite having the highest entropy per switch value (0.375, which is slightly higher than 0.362 for IIB E), loses an additional 11% entropy due to the reduced number of switches (which, in turn, is due to the reduced number of inputs).

HI-1 to HI-4 are four high entropy IIBs. IIB HI-1 is superior to B in that it uses fewer switches and at the same time achieves higher entropy.

From the table, we can also see that the VPR-style IIB is most comparable to IIB B with the same number of switches and 1.4-6.3% higher entropy. In terms of area efficiency, both are about the same and far behind the proposed 2-level IIBs. Similarly, IIB VLL is quite comparable to IIB HI-1, one of the high entropy IIBs built for experiment. Both have a similar 50% depopulated second-level crossbar, but have a different first-level crossbar and use different numbers of L1-MUXes (28 in IIB VLL, versus 40 in IIB HI-1). As a result, HI-1 uses 64 more switches and its entropy is close to the upper bound of IIB VLL. Although they are better solutions than IIB VPR and IIB B, neither of them can provide the same area savings as our proposed IIBs D and E.

5.3 Experimental Results: Routability

We perform routing experiments using all IIBs except IIB VPR and VLL. The IIB VPR and VLL are not used because the input bandwidth constraint requires extra software support, which would further complicate the comparison.

The software flow consists of packer, placer, and router, none of which is timing driven. Packing is done without any input bandwidth limitation. For each design, we use the smallest square array in which the design fits. B is used as the initial IIB in the architecture for routing. After routing, we

Table II. Comparison of External Routing Mux Usage and Router Runtime of Different IIBs

IIB	IIB entropy	External Routing Mux Usage	Normalized Runtime
LO-3	132.0	44.81%	5.110
LO-2	140.0	42.48%	2.905
LO-1	146.7	40.72%	1.810
E	150.5	38.91%	2.257
D	156.5	37.57%	1.790
C	166.2	35.80%	1.253
B	170.3	35.11%	1.000
HI-1	178.8	32.42%	0.913
HI-2	184.2	30.51%	0.668
HI-3	189.5	30.06%	0.528
HI-4	194.1	29.71%	0.521
A	197.2	29.60%	1.423

collect the number of external routing muxes used, that is, length-1, length-4, and length-12 muxes. Then, we replace the initial IIB with each of the experimental IIBs, while leaving the rest of the routing fabric untouched. We keep the same placement and rerun the router. Then we again collect the external routing mux usage.

The typical literature approach uses minimal channel width to compare routing architectures. In this approach, new architectures with successively smaller channel width are created until a design fails to route. We do not use such an approach. Besides runtime considerations (due to the benchmark suite size), the main reason is that, for practical reasons like in Lemieux et al. [2004], we have a quantization constraint on the channel width. This constraint could completely hide the effect of the IIB change (as small as only 0.1% change on the routing resource usage), thus making our results inconclusive. Our approach is to plug different IIBs into the same complete routing fabric, then run the router and collect the external routing mux usage. We believe this is a more appropriate way to measure the small impact of IIB changes on routability.

Table II summarizes the results. The external routing mux usage is the percentage of external routing muxes used in the routing for all designs (i.e., the total number of external routing muxes used by all designs divided by the total number of available external routing muxes in the devices). For IIB LO-2 and LO-3, one design fails to route with 6 and 847 shorts, respectively. We still report routing mux usage by pretending no shorts have happened (i.e., if a mux is shorted and shared by T ($T > 1$) nets, the mux is counted T times in usage). For other IIBs, no design fails to route. We also report router runtime, which is normalized against baseline IIB B. The impact of different IIBs on routability is observed through the change in external routing mux usage and router runtime. From the table, we can see that as the entropy of the IIB decreases, more external routing muxes are used to finish routing, and also in general more time is needed to find routing solutions.

Figure 10 is a plot of external routing mux usage versus IIB entropy. Our hypothesis was that the lost entropy in the IIB should be compensated for

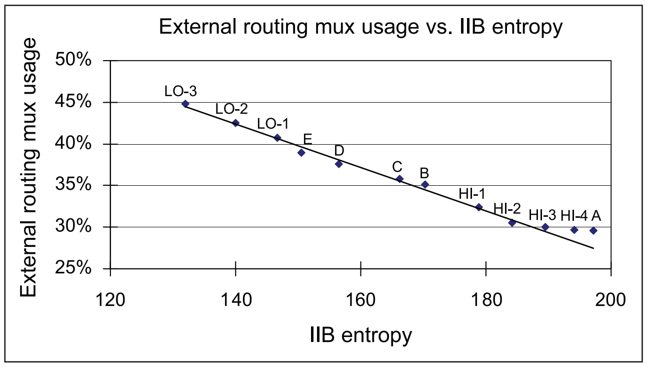


Fig. 10. Routing mux usage versus IIB entropy. It can be seen that as IIB entropy decreases, external routing mux usage increases. The relation is close to linear. From B to E, the external routing resource usage increases from 35.11% to 38.91%.

by the increased usage of external routing muxes, and the relation should be linear. The trend line in the figure is drawn for the points between LO-3 to HI-2, and shows the approximate linear relation. However, with further increase in entropy beyond IIB HI-2 (to HI-3, HI-4 and A), the routing resource usage decreases at a much smaller rate than the trend line indicates. It means that after the IIB entropy reaches a certain point, further increase of the entropy (i.e., IIB flexibility) does not help too much in saving external routing resource usage. This fact can be explained as follows. When IIB entropy is low, the router needs to use extra routing muxes (more than needed to cover the placement wire length) to meet the total entropy needs of the design. As IIB entropy increases, the extra amount resources needed will decrease and get closer to none. That is what happens with the last 3 IIBs (HI-3, HI-4, and A), and is consistent with previous studies that show diminishing returns of increasing the flexibility of the routing architecture [Betz et al. 1999; Lemieux and Lewis 2004].

Figure 11 is a plot of router runtime versus IIB entropy. The general trend is clear that router runtime increases as IIB entropy decreases (with two exceptions at LO_1 and A)[4]. This can be explained by the fact that as IIB entropy decreases, the routing solution space gets smaller, and it takes router longer runtime (and also more routing resources) to construct a solution. Such runtime increase is very small when IIB entropy is high, but becomes very sharp for low entropy IIBs (for example, from LO-1 to LO-2 and LO-3). Further decrease of IIB entropy beyond LO-3 will likely cause much bigger increase in runtime, and also more designs to fail. In fact, if the IIB entropy is reduced to a point such that the total entropy of the whole routing architecture is below the

[4]The first exception is that the router runtime of IIB A is much bigger than other high entropy IIBs. We believe it is due to a slowdown in router caused by the excessive number of switches in this IIB. The second exception is from IIB E to LO-1, which sees a runtime decrease while the entropy decreases. We believe this is mostly due to the 1-level IIB structure in LO-1 (versus 2-level in E), which has 1 fewer level to reach LUT inputs, and helps router runtime.

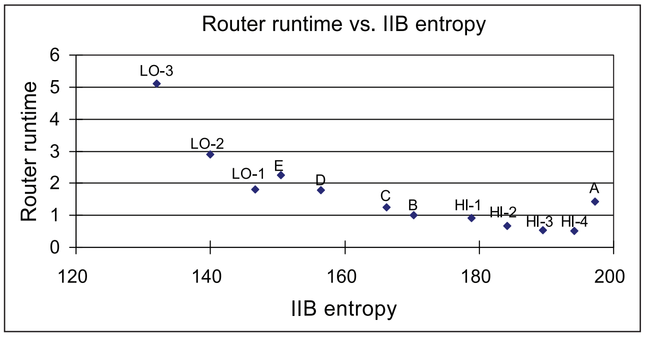


Fig. 11. Router runtime versus IIB entropy. It can be seen that as IIB entropy decreases, the time needed to find routing solution increases. From B to E, the runtime increases from 1.0x to 2.3x.

Table III. Area comparison. A non-zero entry in column 2 represents the scenario when the total external routing resource is adjusted such that external routing resource usage (column 5) remains the same as baseline B. All area percentages are against the total routing area in the baseline.

IIB	External Routing Adjusted Ratio	Number of Switches	IIB Entropy	External Routing Adjusted Usage	IIB Area	External Routing Area	Total Area	Area Saving
B	0%	1280	170.3	35.11%	57.0%	43.0%	100%	0%
E	0%	416	150.5	38.91%	29.0%	43.0%	72%	28%
E	11%	416	150.5	35.11%	29.0%	47.7%	77%	23%
VLL	−6%	896	181.0	35.11%	44.6%	40.5%	85%	15%

lower bound [Feng and Greene 2006], then designs will become theoretically unroutable. In such a case, even an optimal router with unlimited runtime can not find solutions.

5.4 Area Results

At last, we report the area saving results when using IIB E and VLL in place of IIB B in the routing architecture in Table III. The area estimation is based on Actel's flash technology where each switch is implemented using one flash cell.

When IIB B is used, the routing area breakdown is 57% IIB area and 43% external routing area. When IIB E is used in place of IIB B with no change to the external routing, the IIB area is reduced almost by half, resulting in a total routing area 72% of the original. Although this new routing architecture is slightly less routable (external routing resource usage increases from 35.11% to 38.91%), it still routes all designs. So we could use it without further change, resulting in a 28% net routing area saving. If necessary, we can also increase the available external routing resource by 11% to bring the external routing resource usage to the same level as before. This would still give us a 23% net routing area saving. These area saving results are based on our specific flash-based technology and could be different with other technologies. But it does show that the big reduction in switch count of the IIB results in significant area savings in the IIB.

We can also estimate roughly the potential area saving when IIB VLL is used in place of IIB B. We ignore the potential increase (about 2% [Betz et al. 1999]) in cluster count due to the decreased utilization resulting from the input bandwidth limitation. We assume its entropy is the maximal possible 181.0. The IIB area is estimated to be 44.6% of the original total routing area by a linear interpolation between IIB E and B according to the number of switches. The external routing area can be reduced by 2.5% of original total using the linear correlation between IIB entropy and external routing resource usage. The total area will be 85% of the original, resulting in a 15% net saving. Therefore, IIB VLL saves 8% less than IIB E.

6. SUMMARY AND FUTURE WORK

6.1 Summary

In this article we proposed a unified counting and entropy approach to analyze an IIB's routability. We experimentally verified the effectiveness of our approach. The major benefit of such an approach is that it provides a fast and reliable way to evaluate different IIBs without building a complete architecture and running time-consuming place and route experiments.

Using this new approach, we explored a much bigger solution space for IIB design than was possible before. The approach enables us to look at an IIB as a whole, even for a 2-level IIB. We analyzed and compared 3 types of IIBs. We proposed a novel class of highly efficient, yet still quite routable, 2-level IIBs based on type-3 IIB. Compared to VPR-style 2-level IIBs and improved VPR-style IIBs (as proposed in Lemieux and Lewis [2004]), our proposed IIBs have more sparsely populated crossbars and yet have no cluster input bandwidth limitation. Such jointly optimized solutions have not been explored by previous studies, which designed the two crossbars separately.

Our proposed IIBs can achieve big area savings. We showed that in Actel's flash-based implentation, we can save 28% of total routing area by simply replacing the initial IIB B with our proposed IIB E without changing the external routing fabric. It comes with a 3.8% increased usage (from 35.11% to 38.91%) of external routing muxes, and 2.3x router runtime. Alternatively, if we want to keep the usage of the external routing mux unchanged by increasing total external routing resource, we can still achieve a 23% net saving (with likely smaller than 2.3x router runtime.)

6.2 Future Work

There are limitations on our counting approach. Counting in general is a hard problem. We have only considered three special types of IIBs in this article. One limitation is that exact counting for the first level crossbar in a type-2 IIB is currently limited to 2 special cases, and the second special case is known to be nonoptimal. It would be useful if we can get meaningful bounds on D_i for other regular or randomly generated 1-level crossbars [Lemieux and

Lewis 2004], as it enables us to better measure the routing impact of additional switches in the first level crossbar. The other limitation is that our approaches can not be applied to IIBs other than the three types covered in the paper, like the IIBs proposed by Lemieux and Lewis [2004]. Finding ways to count on such IIBs (as well as others) will be useful.

One challenging future work is to consider the joint design of the IIB and the rest of routing network, including the connectivity from BLE outputs to routing tracks, and between routing tracks. Such joint design has the potential to deliver us better results, just as this paper has shown in the joint design of 2-level IIBs.

The area model used in this article is based on switch count. In a real implementation, a detail and accurate area model is often necessary and important. We have shown that our proposed IIBs can save 23% of routing area from a baseline 1-level IIB in a flash-based FPGA. It would be interesting to see how much saving can be achieved with our proposed IIBs in other implementations. Also, future work should investigate the timing impact of our new IIBs, which is largely ignored in this article.

APPENDIX

Estimate Upper and Lower Entropy Bound for the VPR-style IIB

The VPR-style IIB is a type-2 IIB. The 144→18 part in the first level crossbar is populated with 432 switches. It is neither of the two special cases for which we know how to obtain the exact D_i values, so we can not compute the exact entropy. Nevertheless, we use two extreme cases to bound H:

- *Upper-bound case*. The 144→18 crossbar is a perfect crossbar. No 144→18 crossbar with 432 switches is perfect, so it is an optimistic estimation. In this case, $D(x) = \left(\sum_{i=0}^{18} \binom{144}{i} x^i\right)(1+2x)^8$.
- *Lower-bound case*. The 144→18 crossbar is composed of 6 disjoint 24→3 full crossbars (special case 2 in D_i computation). Such crossbar can be improved using the switch moving algorithm in Lemieux and Lewis [2004]; so a real crossbar could only be better. Hence, it is a pessimistic estimation. In this case, $D(x) = \left(1 + \binom{24}{1}x + \binom{24}{2}x^2 + \binom{24}{3}x^3\right)^6 (1+2x)^8$.

  Then, we compute H for the above two cases using Theorem 2.

ACKNOWLEDGMENTS

The authors would like to thank Jonathan Greene for the discussions, and his great efforts to improve this article; and Fei Li, Kai Zhu, Eric Sather, Akshay Sharma, and the reviewers for their many helpful comments.

REFERENCES

AHMED, E. AND ROSE, J. 2004. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Trans. on VLSI, 12*, 288–298.

BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic, Norwell, MA.

BRUALDI, R. A. 1998. *Introductory Combinatorics*. Prentice Hall, Englewood Cliffs, NJ.

CHANG, Y., WONG, D. F., AND WONG, C. K. 1996. Universal switch modules for FPGA design. *ACM Trans. Des. Automat. Electron. Syst. 1*, 80–101.

COVER, T. AND THOMAS, J. 1991. *Elements of Information Theory*, Wiley, Hoboken, NJ.

DEHON, A. 1996. Entropy, counting, and programmable interconnect. In *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA 1996)*, 73–79.

FENG, W. AND GREENE, J. 2006. Post-placement interconnect entropy: How many configuration bits does a programmable logic device need? In *Proceedings of the System-Level Interconnect Prediction (SLIP 2006)*, 41–48.

HUTTON, M. 2001. Interconnect prediction for PLDs. In *Proceedings of the System-Level Interconnect Prediction (SLIP 2001)*, 125–131.

LEMIEUX, G., LEE, E., TOM, M., AND YU, A. 2004. Directional and single-driver wires in FPGA interconnect. In *Proceedings of the International Conference on Field-Programmable Technology (FPT 2004)*, 41–48.

LEMIEUX, G. AND LEWIS, D. 2004. *Design of Interconnection Networks for Programmable Logic*. Kluwer Academic Publishers, Norwell, MA.

LEVENTIS, P., CHAN, M., LEWIS, D., NOUBAN, B., POWELL, G., VEST, B., WONG, M., XIA, R., AND COSTELLO, J. 2003. Cyclone: A low-cost, high-performance FPGA. In *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference (CICC 2003)*. 49–52.

LEWIS, D., AHMED, E., BAECKLER G., BETZ, V., BOURGEAULT, M., CASHMAN, D., GALLOWAY, D., HUTTON, M., LANE, C., LEE, A., LEVENTIS, P., MARQUARDT, S., MCCLINTOCK, C., PADALIA, K., PEDERSEN, B., POWELL, G., RATCHEV, B., REDDY, S., SCHLEICHER, J., STEVENS, K., YUAN, R., CLIFF, R., AND ROSE, J. 2003. The Stratix logic and routing architecture. In *Proceedings of the International Symposium on Field Programmable Gate Arrays (FPGA 2003)*, 12–20.

SHANNON, C. E. 1950. Memory requirements in a telephone exchange. In *Bell Syst. Techn. J. 29*, 343–349.

XILINX INC. 2001. *Virtex-II Field-Programmable Gate Arrays*. San Jose, CA. Web site: http://www.xilinx.com.

Received May 2007; revised September 2007; accepted December 2007