

On the Tradeoff between Power and Flexibility of FPGA Clock Networks

JULIEN LAMOUREUX AND STEVEN J.E. WILTON
University of British Columbia

FPGA clock networks consume a significant amount of power since they toggle every clock cycle and must be flexible enough to implement the clocks for a wide range of different applications. The efficiency of FPGA clock networks can be improved by reducing this flexibility; however, reducing the flexibility introduces stricter constraints during the clustering and placement stages of the FPGA CAD flow. These constraints can reduce the overall efficiency of the final implementation. This paper examines the tradeoff between the power consumption and flexibility of FPGA clock networks.

Specifically, this paper makes three contributions. First, it presents a new parameterized clock network framework for describing and comparing FPGA clock networks. Second, it describes new clock-aware placement techniques that are needed to find a legal placement that satisfies the constraints imposed by the clock network. Finally, it performs an empirical study to examine the tradeoff between the power consumption of the clock network and the impact of the CAD constraints for a number of different clock networks with varying amounts of flexibility.

The results show that the techniques used to produce a legal placement can have a significant influence on power and the ability of the placer to find a legal solution. On average, circuits placed using the most effective techniques dissipate 5% less overall energy and were significantly more likely to be legal than circuits placed using other techniques. Moreover, the results show that the architecture of the clock network is also important. On average, FPGAs with an efficient clock network were up to 14.6% more energy efficient compared to other FPGAs.

Categories and Subject Descriptors: B.7.1 [**Hardware**]: Integrated Circuits – *Types and Design Styles*; *Gate Arrays*; B.7.2 [**Hardware**]: Integrated Circuits – *Design Aids*; *Placement and Routing*

General Terms: Algorithm, Design, Experimentation.

Additional Key Words and Phrases: FPGA, Clock Distribution Networks, Clock-Aware Placement, Low-power Design.

INTRODUCTION

With advancements in process technology, programmable architecture, and computer-aided design (CAD), field-programmable gate arrays (FPGAs) are now being used to implement and prototype large system-level applications. These applications often have several clock domains. In order to support applications with multiple clock domains, FPGA vendors incorporate complex clock distribution circuitry within their devices [Actel 2007; Altera 2005; Altera 2006; Xilinx 2007].

This research was supported by the Altera Corporation and the Natural Science and Engineering Council of Canada. Authors' addresses: Julien Lamoureux, Department of Computing, Imperial College London, SW7 2AZ; email: julienlamoureux@gmail.com; Steve Wilton, Department of Electrical and Computer Engineering, The University of British Columbia, V6T 1Z4; email: steve@ece.ubc.ca.

Permission to make digital/hard copy of part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date of appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2008 ACM 1073-0516/01/0300-0034 \$5.00

Designing a suitable clock distribution network for an FPGA is significantly more challenging than designing such a network for a fixed function chip such as an Application-Specific Integrated Circuit (ASIC). In an ASIC, the locations and skew requirements of each domain are known when the clock network is designed. In an FPGA, however, a single clock network that works well across many applications must be created. When the FPGA is designed, the number of clock domains the user will require, the clock signals that will be generated, the skew requirements of each domain, and where each domain will be located on the chip are all unknown. This forces FPGA vendors to create very complex yet flexible clock distribution circuitry.

This flexibility has a significant area and power overhead. Power is a particular concern, since the clock signals toggle every clock cycle and are connected to a large number of the flip-flops. Previous studies have indicated that in a typical FPGA, 19% of the total FPGA power is dissipated in the clock network [Tuan 2006]. The more flexible the clock network, the more parasitic capacitance on the clock nets, and the more routing switches traversed by each clock signal; this leads to increased power dissipation. Clearly, FPGA vendors must carefully balance the flexibility of their clock distribution networks and the power dissipated by these networks.

The clock distribution network also has an impact on the ability of the computer-aided design (CAD) tools to minimize the power and maximize clock frequency of a user circuit. FPGA clock networks typically are not flexible enough to supply *any* clock signal to *any* flip-flop. As we will discuss in this paper, typical networks allow only a subset of the clock signals to reach particular regions of the FPGA. This imposes additional constraints on the placement algorithm, as well as the clustering algorithm that groups logic elements into clusters. If the clock network is not flexible enough, these constraints could result in increased power dissipation and delay in a user circuit. Again, this balance must be considered by FPGA vendors as they design their clock distribution networks.

In this paper, we investigate the tradeoff between clock network flexibility and the power and speed of user circuits implemented on FPGAs. In particular, we make the following contributions:

1. We present a parameterized framework that describes a family of FPGA clock networks and encompasses the salient features of commercial FPGA clock networks. Such a framework is important as it allows us to reason about and explore clock networks.

2. We present new clock-aware placement techniques that satisfy the placement constraints imposed by the clock network. As described above, the topology of the clock distribution architecture implies constraints on where logic blocks can be placed, and on what logic blocks can be packed together into clusters. For a given clock distribution architecture, our placement algorithm finds a legal placement solution that meets these constraints. As a secondary goal, the algorithms try to find a solution that uses the clock distribution network efficiently. For example, it may be possible to group together logic blocks such that parts of each clock network can be "turned off" or remain unused, thereby saving significant power. Most clock distribution architectures provide for both global and local clocks; the placement algorithm also determines which user clocks are carried by the global distribution network and which user clocks are carried by a local distribution network. We consider several placement algorithms and compare their ability to meet the placement constraints as well as to minimize power by using the clock network efficiently. Our algorithms are implemented into the Versatile Place and Route (VPR) tool [Betz 1999], and are flexible enough to target an FPGA with any clock distribution network that fits within our parameterized framework.
3. We examine how the architecture of the clock network and the placement algorithm used affects the overall power, area, and delay of the FPGA. We consider both the cost of the clock network itself and the impact of the constraints imposed by the clock network. In doing so, we identify the key parameters in our clock framework that have the most significant impact. It is important to note that the approach that we take is empirical and therefore the results and conclusions we make regarding FPGA clock networks are inherently dependent on the clock-aware placement techniques that are used. With that in mind, however, we endeavor to make the clock-aware enhancements using standard and intuitive techniques and consider a number of different techniques to determine which are more suitable.

Together, the aim of these contributions is to provide insight into what makes a good FPGA clock distribution architecture.

Early versions of the parameterized clock network framework and clock-aware placement techniques were presented in [Lamoureux 2006] and [Lamoureux 2007], respectively. In this paper, we combine and expand these studies. Specifically, in this paper the assumptions we make regarding buffer sizing and sharing within the clock networks have been revisited to reflect current low-skew design techniques. Moreover,

the results have been expanded by including the impact of the clock network constraints on the clustering stage of the FPGA CAD flow. Finally, the empirical study has been made more concrete by incorporating a greater number of multiple clock-domain benchmark circuits.

This paper is organized as follows. Section 2 provides background on clock networks and previous work related to FPGA clock networks and CAD. Section 3 describes a parameterized framework used to describe and compare different FPGA clock networks. Section 4 describes new clock-aware placement techniques for FPGAs. Section 5 examines which clock-aware placement techniques perform the best in terms of power and speed. Section 6 then examines how FPGA clock networks affect overall FPGA power, area, and speed. Finally, Section 7 summarizes our conclusions.

BACKGROUND AND PREVIOUS WORK

This section provides background on clock networks and then describes previous work related to FPGA clock networks and clock-aware CAD.

Background

The primary goal when designing a clock network for any digital circuit (ASICs and FPGAs alike) is to minimize clock-skew, and the secondary aim is to minimize power and area. Many low-skew and low-power techniques for ASIC clock networks have been described in the literature. Buffered trees are the most common strategy for distributing clock signals [Friedman 2001]. Buffered trees have a root (clock source), a trunk, branches, and leaves (registers), and are driven by buffers at the trunk and/or along the branches of the tree, as illustrated in Fig. 1 (a). Another buffered-tree approach uses symmetry to minimize clock-skew, as illustrated in Fig. 1 (b). Symmetric buffered trees utilize a hierarchy of symmetric H-tree or X-tree structures to make the path from the source to each register the same length.

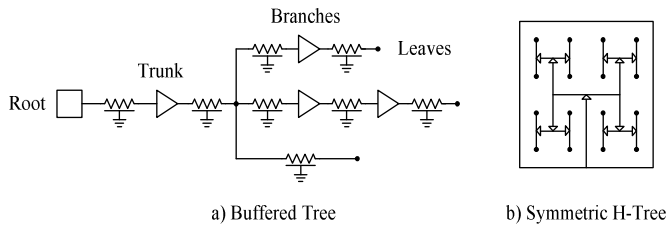


Fig. 1. Example clock distribution networks.

FPGA clock networks differ from ASIC clock networks since they are designed before the application is implemented. Thus, in addition to minimizing clock skew and

power, the FPGA clock network must be flexible enough to implement a wide range of different applications.

Commercial FPGAs

Commercial FPGAs, such as the Altera Stratix III [Altera 2006], the Actel ProASIC3 [Actel 2007], and the Xilinx Virtex 5 [Xilinx 2007] devices, support multiple local and global clock domains. In each of these devices, the FPGA is divided into regions, as illustrated in Fig. 2. The Stratix III has four quadrants that can be further subdivided into four sub-regions (per quadrant), the ProASIC3 has four quadrants, and the Virtex 5 has fixed size regions that are 20 logic rows high and span half the width of the FPGA. The Altera Stratix III provides 16 global clock signals, which can be connected to all the flip-flops on the FPGA, and 22 local clock networks in each of the four quadrants, which can be connected to any of the flip-flops within that quadrant. Similarly, the Actel ProASIC3 devices provide 6 global clocks and 3 local clocks per quadrant and the Xilinx Virtex 5 devices provide 32 global clocks and 10 local clocks. The global clocks in the Virtex 5 are not connected to flip-flops directly; instead, the global clocks drive local clocks within each region.

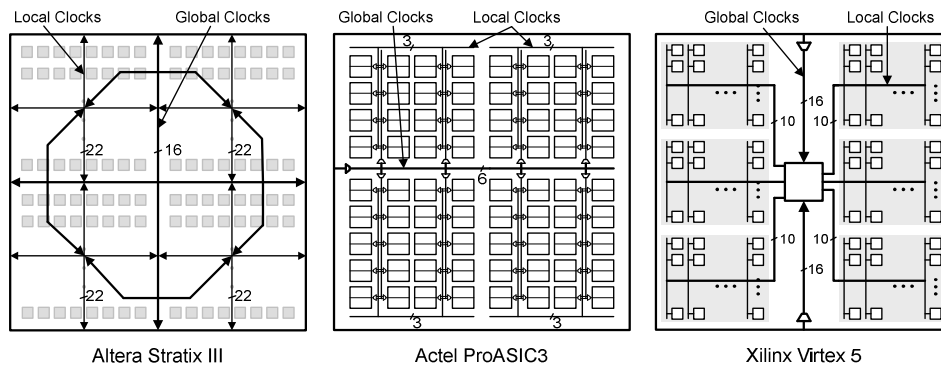


Fig. 2: Commercial FPGA clock networks.

Within the regions, the clocks are typically distributed to rows of logic blocks through a row multiplexer and rib routing channels. In the Stratix II devices [Altera 2005], each row multiplexer chooses 6 clocks from the 24 local/global clocks, and provides them to all the flip-flops in that row, as shown in Fig. 3. In ProASIC3 devices, the row multiplexers choose from 6 global, 3 local, and several internal signals. In the Virtex 5 devices, the row multiplexers choose between 10 local clocks.

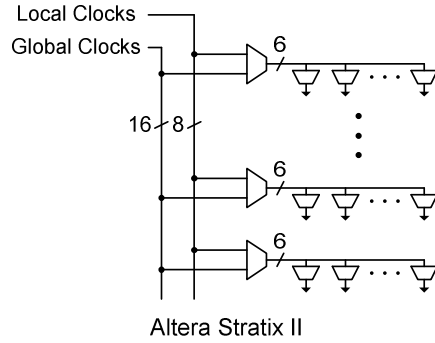


Fig. 3: Multiplexer structure of Stratix II clock networks.

The circuitry that drives the clock networks is similar for each of the three devices. As shown in Fig. 4, the local and global clock networks are driven by control blocks that select the clock signal and dynamically enables or disables the clock to reduce power consumption when the clock signal is not being used. The clock networks can be driven by an external source, an internal source, or by clock management circuitry which multiplies, divides and/or shifts an external source.

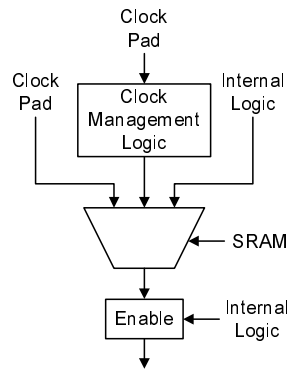


Fig. 4: Control block for local and global clock signals.

Commercial FPGAs have incorporated programmable clock distribution networks that support applications with multiple clocks and feature local and global clock regions for a number of years; however, to our knowledge, the associated clock-aware CAD techniques have not been disclosed.

Full Crossbar and Concentrator Networks

In this paper, we will employ both full crossbar and concentrator crossbar networks as building blocks to connect various stages of the clock distribution network together as described in Section 3. An $n \times m$ single-stage crossbar is a single stage network that connects n inputs to m outputs, as shown in Fig. 5. Fig. 5(a) shows a full crossbar network in which each output can be driven by any input. Such a crossbar requires $n \cdot m$

switches. Fig. 5(b) shows a concentrator network, in which any m -element set of the n input signals can be mapped to the m outputs (without regard for the order of the outputs). A concentrator built this way contains $m \cdot (n-m+1)$ switches [Nakamura 1982]. Sparse crossbar concentrators are most efficient when m is close to n (i.e. the network is close to square) or when m is very small (close to 1). A perfectly square crossbar concentrator only has n switches. As the crossbar concentrator becomes more rectangular, the number of switches approaches that of a full crossbar. Full and concentrator crossbars can also be implemented using fanin-based switches (multiplexers followed by buffers).

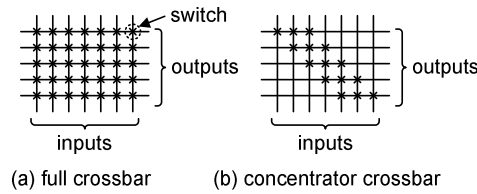


Fig. 5: Two examples of a 7×5 crossbar network.

As an example, a 6×3 concentrator crossbar network illustrated in Fig. 6 shows how concentrators are used to connect rib tracks to spine tracks of the clock networks described in this paper. The multiplexers are implemented with pass-transistors followed by cascaded buffered as described in Section 5.1

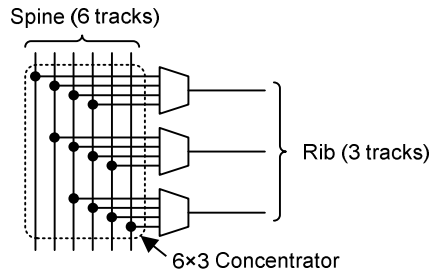


Fig. 6: Example of how concentrators are used in the clock network.

Previous Work

The existing literature on FPGAs has mostly assumed simplified FPGA clock architectures. The study described in [George 1999] examines the design of energy efficient FPGAs. For clock networks, the study proposes that edge triggered flip-flops are used within logic blocks to reduce the power dissipated by the clock network, since this reduces the toggle rates by a factor of two. In [Li 2003],[Poon 2005], FPGA power models that assume simple H-tree clock networks with buffers inserted at the end of each

branch are described. In both models, clock networks span the entire FPGA and are implemented using dedicated (non-configurable) resources. In both cases, the architecture the clock network is fixed.

Clock-Aware CAD

Clock-aware placement has been considered in several studies related to ASICs. As an example, the quadratic placer described in [Natesan 1996] minimizes clock-skew by biasing the placement to evenly disperse clocked cells. As a result, the clock tree generated after placement is more balanced. Another technique, described in [Edahiro 1996], minimizes clock-skew using a placement algorithm to optimally size and place clock buffers within the generated clock tree. Although useful for ASICs, these techniques are not applicable to FPGAs which have fixed (but configurable) clock networks.

FPGA Clock CAD

Only a few CAD studies have considered FPGA clock networks. In [Zhu 1997], clock-skew is minimized during placement by balancing the usage in each branch of the programmable clock network. This technique, however, is not necessary in recent FPGAs since the clock inputs to the logic blocks are buffered, which makes clock delay independent of usage. In [Brynjolfson 2000], dynamic clock management is applied to FPGAs to minimize power. The technique involves dynamically slowing clock frequencies when the bandwidth demand decreases. This technique can also be used in conjunction with dynamic voltage scaling to further reduce overall power. Finally, the T-VPack clustering tool, described in [Betz 1999], is clock-aware since it limits the number of clock signals that are used within each cluster, as specified by the user. Although the clusterer supports circuits with multiple clocks, the corresponding study focused on FPGAs with only one clock.

PARAMETERIZED FPGA CLOCK NETWORK FRAMEWORK

This section presents a parameterized framework for describing FPGA clock networks. The framework can be used to describe a broad range of clock networks and encompasses the salient features of the clock networks used in current and future FPGAs. This framework is important since it provides a basis for comparing new FPGA clock networks and clock-aware CAD techniques.

The framework assumes a clock network topology with three stages. The first stage programmably connects some number of clock sources (clock pads, PLL/DLL outputs, or internal signals) to the center of some number of clock regions. The second stage

programmably distributes the clocks to the logic blocks within each region. The third stage connects the clock inputs of each logic block to the flip-flops within that logic block. This topology is described in more detail in the following subsections.

Clock Sources

The source of the user clocks can be external, from a dedicated input pad, or internal, generated by a phase-locked loop (PLL), a delay-locked loop (DLL), or even the core of the FPGA. In all cases, we assume that these clock sources are distributed evenly around the periphery of the FPGA core. In our model, the number of potential clock sources is denoted n_{source} , meaning there are $n_{source}/4$ potential clock sources on each side of the FPGA.

Global and Local Clock Regions

The clock network has both local and global resources. The global clock resources distribute large user clocks across the entire chip (but not necessarily to every logic block, as described in Section 3.3). The local clock resources, on the other hand, distribute the smaller user clocks to individual regions of the FPGA. Although it is possible to distribute a large user clock to the entire chip by stitching together several local clocks, this would be less efficient than using a global clock.

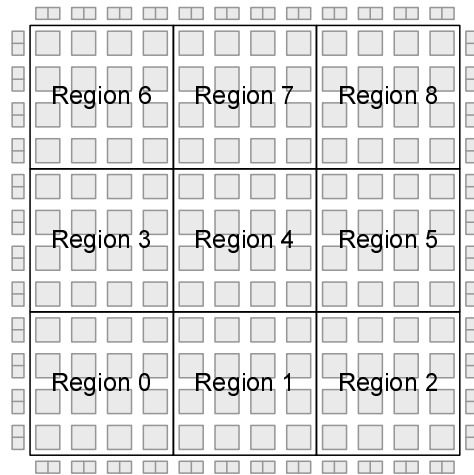


Fig. 7: Example FPGA with 3x3 clock regions.

To support local clocks, the FPGA fabric is broken down into a number of regions, each of which can be driven by a different set of clock sources (the same clock source can be connected to more than one region). The number of regions in the X-dimension is denoted by n_{x_region} , and the number of regions in the Y-dimension is denoted by n_{y_region} . The total number of regions is thus $n_{x_region} * n_{y_region}$. Fig. 7 shows an

example FPGA in which both nx_region and ny_region are both 3, which produces 9 clock regions.

First Network Stage

The first stage of the clock network programmably connects the clock sources on the periphery of the chip to the center of each region. The first stage consists of two parallel networks: one for the global clocks and one for the local clocks. We denote the total number of global clock signals as W_{global} . The global clock network selects $W_{global}/4$ signals from each of the $nsource/4$ potential clock sources on each side, as shown in Fig. 8. This selection is done using a concentrator network on each side of the chip (see Section 0). The use of a concentrator network guarantees that any set of $W_{global}/4$ signals can be selected from the $nsource/4$ sources on each side. This architecture does not, however, guarantee that any set of W_{global} signals can be selected from the $nsource$ potential sources; it would be impossible to select more than $W_{global}/4$ signals from any side. Relaxing this restriction would require an additional level of multiplexing (or larger concentrators and longer wires), which we do not include in our model.

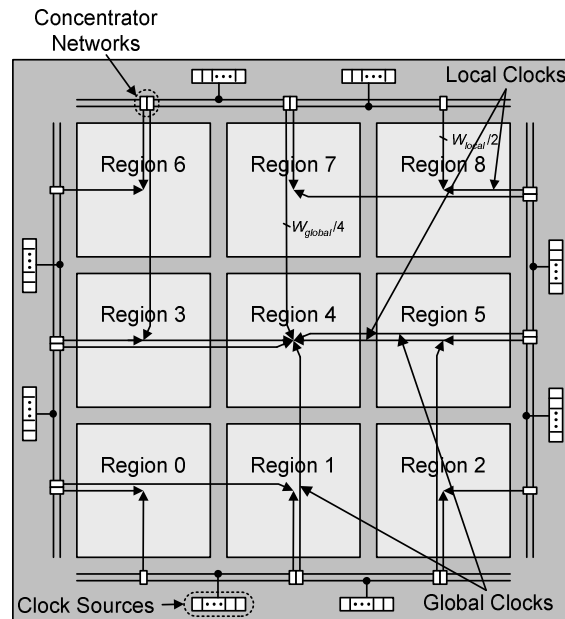


Fig. 8: Connections from the periphery to the center of regions (local clocks) and the center of the FPGA (global clocks).

All W_{global} signals are routed on dedicated wires to the center of the chip. Since the sources all come from the periphery of the chip, the connection between the sources and the center of the chip will introduce little or no skew. In an architecture in which some sources come from inside the chip, the drivers can be sized such that skew is minimized.

From the center of the chip, all W_{global} clocks are distributed to the center of all regions using a spine-and-ribs distribution network with n_{y_region} ribs, as shown in Fig. 9. Although an H-tree topology would have a lower skew, it is more difficult to mesh such a topology onto a tiled FPGA with an arbitrary number of rows and columns.

There is one local clock network per region. We denote the number of local clocks per region as W_{local} . The W_{local} signals are selected from the two sides of the FPGA that are closest to the region, as shown in Fig. 8 (if the number of regions in either dimension is odd, the selection of the “closest” side is arbitrary for regions in the middle). Half of the W_{local} signals are selected from the sources on each of the two sides using a concentrator network. The use of a concentrator network guarantees that any set of $W_{local}/2$ signals can be selected from the $n_{source}/4$ potential sources on each side. Driver sizing can be used to minimize the skew among the clocks connected to each region. Skew between regions is not as important, since global clocks will likely be used if a clock is to drive multiple regions.

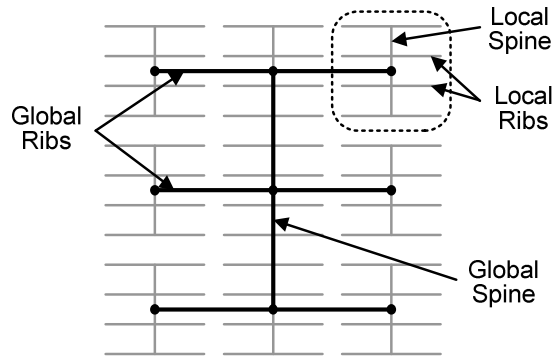


Fig. 9: Global clock connections from the center of the FPGA to the center of regions.

Second Network Stage

The second network stage programmably connects clock signals from the center of each region to the logic blocks within that region. There is one of these networks for each region.

The input to each second stage network consists of W_{global} global clocks and W_{local} local clocks from the first stage described in Section 0. These clocks are distributed using a spine-and-ribs topology as shown in Fig. 10. The spine contains $W_{global}+W_{local}$ wires. In each row, a concentrator network is used to select any set of W_{rib} clocks from the spine. These clocks are distributed to the logic blocks in that row through a local rib. Each logic block in the row connects to the rib through another concentrator network; the concentrator is used to select any set of W_{lb} clocks from the rib.

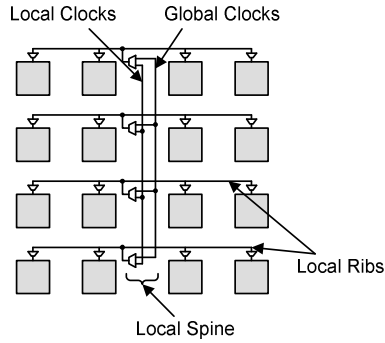


Fig. 10: Second stage of the clock network.

Third Network Stage

Finally, the third network stage programmably connects the W_{lb} logic block clocks to the N logic elements within the logic block. This is illustrated in Fig. 11. In order to provide flexibility to the clustering tool, we assume that the clock pins of the logic block are connected to the clock pins of the logic elements (within that logic block) using a full-crossbar network, such that each of the N logic elements can be clocked by any of the W_{lb} logic block clocks. Note that this is only stage of the clock network that uses full-crossbars; the other stages use concentrator crossbars to reduce the number of switches.

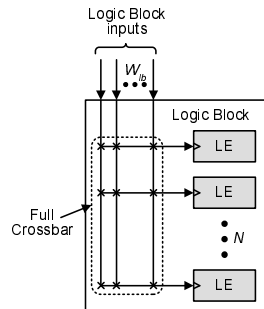


Fig. 11: Third stage of the clock network.

Table I summarizes the parameters of the FPGA clock network framework and includes corresponding values that are used in the empirical studies described in Section 5 and 6 of this paper.

Table I: Clock network parameters.

Parameter	Meaning	Baseline Value	Range in this Paper
nx_region	Number of clock regions in the X dimension	2	1 – 5
ny_region	Number of clock regions in the Y dimension	2	1 – 5
$nsource$	Number of potential clock sources (one quarter of these are on each side)	128	128
W_{global}	Number of global clocks that can be routed to the fabric at one time	52	0-52
W_{local}	Number of local clocks that can be routed to each region at one time	52	52
W_{rib}	Number of clocks that can be routed to all logic blocks in a single row of a region at one time.	10	6-14
W_{lb}	Number of clocks that can be connected to each logic block.	2	1-10

CLOCK-AWARE FPGA CAD

The previous section presented a parameterized framework to describe FPGA clock networks. This section describes new clock-aware placement techniques for finding a legal placement solution that satisfies all of the placement constraints imposed by the clock network.

Clock-Aware Placement

Placement can have a significant impact on power, delay, and routability. Placing logic blocks on a critical-path close together minimizes delay and placing logic blocks connected by many wires close together improves power and routability. Power can be further minimized by assigning more weight to connections with high toggle rates, as described in [Lamoureux 2005]. For applications with many clock domains, however, constraints that limit the number of clock nets within ribs (W_{rib}) and within a region (W_{local}) can interfere with these optimizations.

To investigate how clock networks constraints affect the power, delay, and routability during placement, the T-VPlace algorithm from [Betz 1999] was enhanced to make it clock-aware. T-VPlace is based on simulated annealing. The original cost function used by T-VPlace has two components. The first component is the *wiring cost*, which is the sum of the bounding box dimensions of all nets (not including clock nets). The second component is the *timing cost*, which is a weighted sum of the estimated delay of all nets. The cost of a swap is then:

$$\Delta C = \lambda \cdot \frac{\Delta \text{Timing Cost}}{\text{Previous Timing Cost}} + (1 - \lambda) \cdot \frac{\Delta \text{Wiring Cost}}{\text{Previous Wiring Cost}} + \quad (1)$$

The *PreviousTimingCost* and *PreviousWiringCost* terms in the expression are normalizing factors that are updated once every temperature change, and λ is a user-defined constant which determines the relative importance of the cost components.

Three enhancements were made to T-VPlace in order to make it clock-aware. First, a new term was added to the existing cost function to account for the cost of the clock. Second, a new processing step that determines which user clock nets use local clock network resources and which clock nets use global clock network resources was added. Third, the *random* initial placement approach was replaced with a new routine that finds a legal placement. Each enhancement is described below.

New Cost Function

The new cost function is the same as the original cost function, except that it has a new term to account for the cost of using clock network resources. Intuitively, the new term in the cost function minimizes the usage of clock network resources, which minimizes clock power and is key for finding legal placements. The new cost function is described by the following expression:

$$\Delta C = \lambda \cdot \frac{\Delta \text{Timing Cost}}{\text{Previous Timing Cost}} + (1 - \lambda) \cdot \frac{\Delta \text{Wiring Cost}}{\text{Previous Wiring Cost}} + \gamma \cdot \frac{\Delta \text{Clock Cost}}{\text{Previous Clock Cost}} \quad (2)$$

Like the two original terms, the new term is normalized by the cost of the clock from the previous iteration and is weighted by the factor γ . The best value for γ depends is found empirically by doing a parametric sweep to find value which produces the most efficient (and legal) placements. In this paper, we consider two different cost functions for the clock term. The best γ values were found to be 1.0 and 0.3 for first and second cost functions (see below), respectively.

The first cost function, which we call the *standard* cost function, is based on the amount of clock network resources needed to route all the user clock nets. Specifically, the cost function counts the number of clocks used in each rib, local region, and global region. Moreover, the cost of each resource type (rib, local, or global) is scaled by a constant (K_{rib} , K_{local} , or K_{global} , respectively) to reflect the capacitance of those resources.

Although straight-forward, the above cost function can be short-sighted when large user clock nets (with numerous pins) occupy more than one region. Unless all the pins are moved from a region, the cost of occupying that region does not change.

The second cost function, which we call the *gradual* cost function, changes when large nets are partially moved. The function scales the cost of adding an LE to a region based on how many other LEs in that region are connected to the same user clock net as that LE. Specifically, the incremental cost of adding an LE that uses clock net i to region j is described by the following expression.

$$\Delta\text{Clock Cost}(i,j) = K_j \cdot \frac{\text{max_pins}(i,j) - \text{pins}(i,j) + 1}{\text{max_pins}(i,j)}$$

where, (3)

$\text{pins}(i,j) \equiv$ number of pins of net i in region j

$\text{max_pins}(i,j) = \min(\#\text{pins of clock net } i, \#\text{LEs per region } j)$

In the expression, K_j is the weight factor for the region j , which is either K_{rib} , K_{local} , or K_{global} , depending on the type of region is being considered, $\text{pins}(i,j)$ is the number of pins of net i that are currently placed in region j , and $\text{max_pins}(i,j)$ is the maximum number of pins from net i that could be placed in region j . This is determined either by the number of pins on net i , if the entire fits within region j , or by the number LEs in region j , if the net has more pins than there are LEs in that region.

Intuitively, both cost functions encourage swaps that reduce the amount of clock network resources that are used. In the second cost function, however, the cost of moving a logic block to a region is smaller when other logic blocks in that region are connected to the same clock net. The goal of the gradual cost function is to prevent large clock nets from spreading out to more regions than necessary. However, since the function does not reflect the actual cost of the clock as directly as the first cost function, the overall results may not be minimized as intended. To determine which cost function is most suitable, we performed an empirical study which is described in Section 5.

Clock Resource Assignment

The second enhancement made to the placer was to make it able to determine which user clock nets should use the local clock network resources and which user clock nets should use the global clock network resources. Although this decision could be left to the user, it is more appropriate for the tool to make the assignment since it is convenient and the user may not be familiar with the underlying architecture of the clock network.

Global clock network resources are more expensive than the local clock network resources in terms of power since they are routed to the center of the FPGA before spanning to the center of the clock regions. Depending on the clock network and application, global clock network resources may also be in short supply. Therefore,

global clock networks should be reserved for large nets that do not fit within local regions or for nets that are inherently spread out.

We consider two approaches for assigning global clock network resources. The first approach assigns global resources *statically*, based on the size (fanout) of the clock nets. The advantage of this static approach is that it is quick and easy. The disadvantage is that it overlooks smaller nets that require global resources because they are inherently spread out. The second approach assigns global resources *dynamically* during placement based on how spread out the clock nets are. Fig. 12, Fig. 13, and Fig. 14 describe each assignment technique in more detail.

The static assignment routine is described in Fig. 12. It begins by determining how many clock nets use global clock network resources based on the number of clock nets in the application, the number of global clock network resources available, and a user-defined relaxation factor called RELAX_FAC. Intuitively, we are trying to use the fewest number of global clock resources as possible since they consume more power and should be reserved for clock domains that need to be spread out across more than one clock region. In Fig. 12, the *num_global_min* parameter is the absolute minimum number of global clock resources that would be needed to legally place the application. This value is determined by calculating how many application clocks nets would remain if each local clock network resource could be used to implement one of the clock nets. The placer, however, may not be able to find a placement that only uses local clock resources since application clock nets are often too large and need to be spread out over more than one clock region. The relaxation factor makes finding a legal solution easier by allowing more clock nets to be implemented on global clock resources. In our experiments we use a relaxation factor of 0.5. Once the number of global clock resources to be used (*num_global_relaxed*) has been decided, the routine assigns the global resources to the clock nets with the greatest number of pins (highest fanout).

```
assign_global_resources_statically () {  
    num_global_min = max( 0, num_clock_nets - num_local_regions *  
    num_local_clocks );  
    num_global_relaxed = min( num_clock_nets, num_global_clocks,  
    num_global_min +  
        num_clock_nets * RELAX_FAC );  
  
    /* biggest to smallest by number of pins*/  
    sorted_clock_nets = sort( clock_nets, num_pins );  
  
    for ( iclk = 0; iclk < num_global_relaxed; iclk++ )  
        use_global[sorted_clock_nets[iclk]] = TRUE;  
}
```

Fig. 12. Pseudo-code description of the static clock resource assignment technique.

The dynamic assignment technique described in Fig. 13 is applied during placement. Initially, all the user clock nets are assigned to use local clock network resources. Then, during the simulated annealing routine, clock nets can be *reassigned* to use global resources if the placer cannot find a legal placement. Specifically, the placer reassigns clock nets when the cost of the clock stops decreasing and the placement is still not legal. When clock nets are reassigned, the temperature is reset back to the initial temperature to find a placement solution given the new assignment. The following pseudo-code is based on T-VPlace, which is described in [Betz 1999].

The clock nets are *reassigned* using the routine described in Fig. 14. The routine begins by calculating how spread out each clock net is by calculating the *locality distance*, which counts how many clock pins would need to move in order to make the clock net local. After sorting each net by locality distance, it then assigns global clock network resources to half of the remaining local clock nets that have the highest locality distance.

```

placement_with_dynamic_assign () {

    assign_local_resources_to_all_clock_nets ();
    cost = random_placement();
    init_T = initial_temperature();
    t = init_T;

    while ( exit_criterion () == False ) {
        count = swap_count = 0;
        old_cost = cost;
        while ( inner_loop_criterion () == False ) {
            if (try_swap (t, cost) == 1) {
                swap_count++;
            }
            count++;
        }
        if ( is_placement_legal() == False &&
            swap_count < count / 2 && old_cost <= cost) {
            reassign_clocks ();
            t = init_T; /*reset temperature */
        }
        t = update_temp ();
    }
}

```

Fig. 13. Pseudo-code description of the dynamic clock resource assignment technique.

```

reassign_clock ( int num_global_available ) {
    num_not_local = 0;
    for (iclk=0; iclk<num_clock_nets; iclk++) {
        if (use_global[iclk] == FALSE) {
            locality_dist = calc_locality_distance (iclk);
            locality_dist[iclk] = locality_dist;
            if (locality_dist > 0) {
                num_not_local++;
            }
        }
    }
    /* sort biggest to smallest distance*/
    sorted_clock_nets = sort ( clock_nets, locality_dist );

    num_to_reassign = min( (num_not_local + 1) / 2, num_global_available )
    for (iclk=0; iclk<num_to_reassign; iclk++) {
        use_global[sorted_clock_nets[iclk]] = TRUE;
    }
}

```

Fig. 14. Pseudo-code description of the routine used to reassign clock nets.

Legalization

The final enhancement needed to make the placer clock-aware is to produce a placement that is legal. Legalization ensures that the number of different clock nets used in every region is less than or equal to the number of clock resources that are available in that region.

We consider two approaches. The first approach, called the *pre-placement* approach, finds a legal solution before placement. A legal solution is found using simulated annealing with the timing and wiring cost components turned off (leaving only the clock cost component turned on). If a legal placement is found, the actual placement is then performed with all three cost components turned on but only allowing legal swaps.

The second approach involves legalizing during the actual placement. The algorithm starts with a random placement and then uses simulated annealing with the timing, wiring, and clock costs (from Equation 2) turned on. To gradually legalize the placement, the clock cost component is modified to severely penalize swaps that make the placement either illegal or more illegal. In other words, illegal swaps are allowed but they have a higher cost. Explicitly, we multiplied the cost of using a rib, spine, or global routing resource that is not available by a constant value, called *Illegal_Factor*. Intuitively, a large value forces the placer to find a legal solution quickly but limits its ability to make major changes to the way the clock resources are used once a legal placement is found. In our experiments, we found 10 to be a suitable value for *Illegal_Factor*.

CLOCK-AWARE PLACEMENT RESULTS

This section begins by describing the experimental framework that is used in this paper and then compares the clock-aware placement techniques that were described in the previous section.

Experimental Framework

The same empirical framework is used in Section 5 and Section 6. A suite of benchmark circuits is implemented on a user-specified FPGA architecture using standard academic FPGA CAD tools. The CAD tools consist of the Emap technology mapper [Lamoureux 2005], the T-VPack clusterer [Betz 1999], the VPR placer (with clock-aware enhancements), and the VPR router [Betz 1999]. Note that the T-VPack does not need to be enhanced since it is already clock-aware. Finally, the power, area, and delay of each implementation are modeled using VPR for area and delay and the power model from [Poon 2005], which has been integrated into VPR.

The VPR models are very detailed, taking into account specific switch patterns, wire lengths, and transistor sizes. After generating a user-specified FPGA architecture, VPR places and routes a circuit on the FPGA and then models the power, area, and delay of that circuit. The area is estimated by summing the area of every transistor in the FPGA, including the routing, CLBs, and configuration memory. The delay is estimated using the Elmore delay model and detailed resistance and capacitance information obtained from the router. The power is modeled using the capacitance information from the router and externally generated switching activities to estimate dynamic, short-circuit, and leakage power. In this paper, the switching activities, which are required by the power model, are obtained using gate-level simulation and pseudo-random input vectors.

We enhanced the VPR power and area models to account for the parameterized clock network described in Section 0. Similar techniques to those used in VPR along with the following buffer sharing and sizing assumptions were used to model the clock networks:

1. Each switch is implemented using a transmission gate controlled by an SRAM cell. The transmission gate consists of one minimum sized NMOS transistor and one 2X PMOS transistor in parallel.
2. Shared buffers are used to drive all periphery, spine, rib, and CLB clock network wires.
3. Large cascaded buffers with four stages (1X, 4X, 16X, and 64X) are used to drive the periphery, spine, and rib wires and smaller cascaded buffers with three stages (1X, 4X, and 16X) are used to drive the CLB clock wires.

4. Large (64X) non-inverting repeaters are used to drive very long wires and are spaced by 40 FPGA tiles.
5. Unused clock networks are turned off to reduce power consumption.

An example of clock network switches of buffers is illustrated in Fig. 15. Note that these buffer sizing and sharing assumptions differ from the assumptions described in our earlier work [Lamoureux 2006] in which the buffers were smaller but not shared, and the repeaters spacing was significantly smaller, separated by only 1 FPGA tile. The new assumptions reduce the power, area, and skew of the clock network by reducing the number of buffers and repeaters and the overall amount of parasitic capacitance.

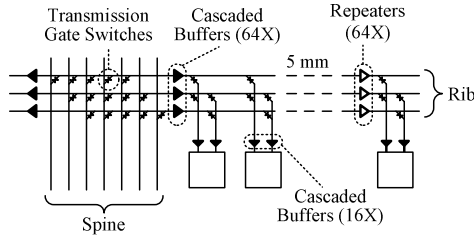


Fig. 15: Example of buffer sizing and sharing in the clock network.

Benchmark Circuits

In order to empirically investigate new FPGA clock network architectures and clock-aware CAD tools, benchmark circuits with multiple clock domains are needed. Existing academic benchmark circuits, however, are small and have only one clock. Moreover, since developing system-level circuits with multiple clock nets is labor intensive and expensive, commercial vendors are reluctant to release their intellectual property.

As a solution, we developed a technique to combine a number of single domain benchmark circuits to form larger multi-domain circuits that resemble system-level circuits from a place and route viewpoint. To connect the circuits, we begin by determining the number of primary inputs (N_{PI}) and primary outputs (N_{PO}) the meta-circuit should have using Rent's rule [Landman 1971]. Rent's rule relates the number of pins (N_p) to the number of gates (N_g) in a logic design using Expression 4.

$$N_p = N_{PI} + N_{PO} = K_p \cdot N_g^\beta \quad (4)$$

where β is the Rent's constant and K_p is a proportionality constant. The values of the two constants vary for different circuit types. β is lower for circuits with pin counts that do not increase quickly when the size of the circuit increases, such as static memory. K_p is lower for circuits with fewer pins in general. In [Bakolgu 1990], an empirical study

found that β varies between 0.12 and 0.63, and K_p varies between 0.82 and 6 for chip-level designs. For circuits implemented on gate arrays, β was 0.5 and K_p was 1.9. These are the values used in this paper.

After determining the number of primary inputs and output, all the sinks and sources of the meta-circuit are listed in arrays as illustrated in Fig 16. Each sink is then assigned a source using the algorithm in Fig. 17. Two synchronizing flip-flops are inserted between connections from IP outputs to IP inputs since the IP cores use different clocks.

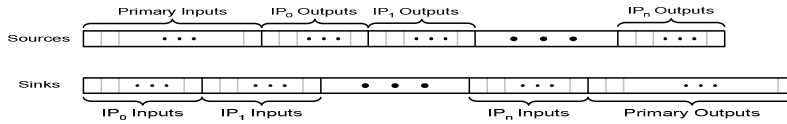


Fig. 16: Stitch connection scheme.

```

j = 0;
for (i = 0; i = Ni + NPO; i++) {
    sink = sinks[i];
    source = sources[j % (NO + NPI)];
    connections{sink} = source;
    j++;
}

```

Fig. 17: Benchmark stitching algorithm.

Using this technique and benchmark circuits from MCNC, several large benchmark circuits with multiple clock domains were generated. Table II lists the new benchmark circuits and provides additional details regarding the circuit size and number of clock domains. Specifically, column 2 specifies the total number of LUTs each benchmark circuit; columns 3 to 5 specify the number of flip-flops in the sub-circuits, between the sub-circuits, and in total, respectively; columns 6 and 7 specify the number of primary inputs in the sub-circuits and in the benchmark circuit, respectively; and columns 8 and 9 specify the number of primary outputs in the sub-circuits and in the benchmark circuit, respectively. Note also that the benchmark names are intended to provide some insight regarding the size of the clock domains. As an example, 1lrg40sml consists of one large circuit and 40 small circuits.

Table II. Benchmark circuit names and attributes.

Benchmark Circuit	# LUTs	# FFs			# PIs		# POs	
		Sub-circuit	Interface	Total	Sub-circuit	External	Sub-circuit	External
10lrg	14320	3316	1464	4780	831	101	887	158
10lrg_reordered	14320	3316	1464	4780	831	101	887	158
15med	5710	1367	994	2361	587	91	568	90
1lrg40sml	9452	2317	862	3179	569	138	559	138
2lrg4med80sml	15289	3403	5716	9119	3059	245	1704	174
30med	5210	492	1382	1874	795	112	592	92
30mixedsize	12537	1343	2056	3399	1163	150	993	133
30seq20comb	16398	1342	1642	2984	992	191	803	165
3lrg50sml	10574	2861	1572	4433	925	156	904	155
40mixedsize	12966	2758	1820	4578	1071	161	941	146
4lrg4med16sml	11139	3278	1748	5026	1000	126	1006	133
4lrg4med4sml	8873	2598	1270	3868	739	107	721	105
50mixedsize	16477	3685	1996	5681	1177	179	1155	177
5lrg35sml	11397	2222	1424	3646	771	60	940	231
60mixedsize	15246	1731	3486	5217	1900	198	1479	168
6lrg60sml	15838	3673	4364	8037	2388	218	1554	165
70s1423	13440	5180	2128	7308	1260	251	420	124
70sml	9301	1134	1264	2398	801	170	735	162
lotsaffs	7712	3123	632	3755	367	51	477	162
lotsaffs2	9309	3609	958	4567	597	118	616	138

FPGA Architecture and Assumptions

All experiments target island-style FPGAs implemented in a 180nm TSMC process. We use the baseline FPGA architecture described in [Betz 1999], which consists of logic blocks with 10 logic elements each and a segmented routing fabric with length 4 wires. For the clock network, we assume the baseline clock architecture described in Table I, which is similar to current commercial architectures.

For each experiment, the size of the FPGA is determined by the size of the benchmark circuit. Specifically, the size is determined by finding smallest 2-dimensional array of FPGA tiles ($n_x \times n_y$) with enough logic blocks to implement the benchmark circuit. For the general routing (not clock routing), the channel width is selected by finding the minimum routable channel width and then adding 20% to that channel width. These assumptions serve to model the case when FPGAs are highly utilized.

Placement Results

To compare the techniques for each of the three enhancements, we implemented eight different clock-aware placers (one for each possible combination). The eight placers are described in Table III. As an example, the first placer (Placer 1) uses the standard function for the clock term in the cost function, the static approach to assign global clock network resources, and the pre-placement approach to legalize the placement. Note that Placer 3 and Placer 7 assign global clock network resources dynamically during the pre-placement, while Placer 4 and Placer 8 assign global clock network resources dynamically during the actual placement since there is no pre-placement in the latter implementations.

Table III. Techniques used by each placer.

Placer	Cost Function	Assignment Technique	Legalization Technique
Placer 1 (P1)	standard	statically	pre-placement
Placer 2 (P2)	standard	statically	during placement
Placer 3 (P3)	standard	dynamic	pre-placement
Placer 4 (P4)	standard	dynamic	during placement
Placer 5 (P5)	gradual	statically	pre-placement
Placer 6 (P6)	gradual	statically	during placement
Placer 7 (P7)	gradual	dynamic	pre-placement
Placer 8 (P8)	gradual	dynamic	during placement

Table IV presents the overall energy per cycle dissipated by each benchmark circuit when implemented by the eight different clock-aware placers. The average is calculated using the geometric mean (rather than the arithmetic mean) to ensure that each benchmark circuit contributes evenly to the average, regardless of its size. Moreover, the averages only include benchmark circuits that were successfully implemented by every placer to make the results comparable.

A number of observations can be drawn from Table IV. First, the placer fails to find a legal solution in some cases. There is no guarantee that a legal placement will be found even when the placer starts by looking for a legal solution. In this experiment, placers P1 and P2 failed to find a legal placement for the 10lrg, 70s1423, and 8lrg benchmark circuits and placer P5 and P6 failed to find a legal placement for 70s1423. These placers all use the static approach to assign local and global clock resources. In terms of energy, the placers that use the static approach (P1, P2, P5, and P6) also performed worse than the other placers (P3, P4, P7, and P8), which use the dynamic approach. Intuitively, assigning global resources dynamically works better since more information is available

when the resources are being assigned; the placer can determine which nets are more spread out and minimize the number of global clock resources that are used.

Table IV. Overall energy per clock cycle.

Benchmark	Overall Energy (nJ)								
	Orig.	P1	P2	P3	P4	P5	P6	P7	P8
10lrg	6.63	-	-	6.70	6.41	6.74	6.34	6.45	6.55
10lrg_reordered	6.98	6.66	6.78	6.67	6.52	6.59	7.19	6.69	6.48
15med	2.91	-	2.95	2.85	2.84	2.89	2.89	2.81	2.81
1lrg40sml	4.50	4.61	4.49	4.44	4.35	4.48	4.49	4.49	4.34
2lrg4med80sml	12.29	11.70	13.01	11.07	12.18	11.44	13.52	11.29	11.71
30med	2.68	2.59	2.61	2.51	2.49	2.67	2.60	2.53	2.57
30mixedsize	7.24	6.57	6.63	6.53	6.36	6.54	6.60	6.32	6.65
30seq20comb	6.48	6.26	6.25	6.26	6.04	6.85	6.32	6.20	6.02
3lrg50sml	5.42	5.01	5.05	4.85	4.84	5.15	5.14	4.92	5.17
40mixedsize	7.79	7.66	7.80	7.95	7.51	7.63	7.84	7.55	7.54
4lrg4med16sml	6.34	6.44	6.53	6.46	6.36	6.33	6.60	6.20	6.25
4lrg4med4sml	4.49	4.46	4.50	4.43	4.40	4.49	4.39	4.42	4.37
4lrg	4.63	4.50	4.59	4.55	4.55	4.47	4.58	4.52	4.47
50mixedsize	9.98	9.62	9.92	9.64	9.45	9.59	9.77	9.68	9.95
5lrg35sml	6.25	6.11	6.20	6.37	6.09	6.10	6.76	5.95	5.95
60mixedsize	9.56	9.19	9.13	8.80	8.97	9.09	9.29	9.18	9.00
6lrg60sml	12.02	11.40	12.67	11.49	12.33	11.59	11.79	11.55	11.64
70s1423	10.99	-	-	10.34	10.77	-	-	10.32	10.97
70sml	5.07	5.24	5.03	4.70	4.68	5.11	4.89	4.78	4.75
8lrg	5.03	-	-	4.89	4.84	4.89	4.84	4.89	4.84
lotsaffs2	3.73	3.36	3.39	3.38	3.38	3.47	3.73	3.46	3.36
lotsaffs	2.32	2.19	2.25	2.21	2.22	2.24	2.28	2.21	2.24
Geomean	5.94	5.73	5.84	5.67	5.65	5.77	5.90	5.65	5.66
% Diff.	0.0	-3.5	-1.7	-4.6	-4.9	-2.9	-0.6	-4.9	-4.7

Another notable observation is that energy efficiency was similar when the standard and gradual cost functions were used. On average, the energy results of placers P1 to P4 correspond closely to the energy results of P5 to P8. However, the results suggest that the gradual cost function works better for finding a solution that is legal. Placers that use the standard function failed in 7 cases compared to only 2 cases for placers that use the gradual function.

Finally, the third observation is that legalizing pre-placement works are well as legalizing during placement. In fact, when static assignment is used, the pre-placement results were slightly better. The pre-placement does not seem to lock the placement in local minima, likely because the pre-placement goal of placing logic within individual clock domains together does not conflict with goals of the final placement.

To examine the placement techniques further, Table V compares the average energy of the clock and general purpose routing resources and the average critical-path delay of

each clock-aware placer to that of the original non-clock-aware placer from VPR. Note that the original placer ignores clock nets and any of the associated placement constraints. Therefore, the placement solutions produced by the original placer are not legal. This table serves to highlight the impact that the clock constraints have on each of the placers.

Table V. Impact of clock constraints for each placer.

Placer	Overall Energy (nJ)	% Diff	Clock Energy (nJ)	% Diff	Routing Energy (nJ)	% Diff	Tcrit (ns)	% Diff
Orig.	5.94	-	2.31	-	2.16	-	49.9	-
P1	5.73	-3.5	1.96	-15.1	2.33	7.8	49.8	-0.1
P2	5.84	-1.7	1.99	-14.2	2.41	11.7	50.4	1.1
P3	5.67	-4.6	1.89	-18.4	2.33	8.0	50.3	1.0
P4	5.65	-4.9	1.93	-16.6	2.28	5.7	50.2	0.8
P5	5.77	-2.9	2.17	-6.2	2.12	-1.9	50.2	0.6
P6	5.90	-0.6	2.17	-6.1	2.29	6.1	49.6	-0.5
P7	5.65	-4.9	2.03	-12.3	2.17	0.6	50.1	0.5
P8	5.66	-4.7	2.05	-11.5	2.18	1.1	49.8	-0.2

The table demonstrates that the clock constraints can have a significant impact on the energy dissipated by the clock networks and general purpose routing resources. As expected, the clock-aware implementations dissipate significantly less clock power than the original (non clock-aware) implementations since the clock-aware placers minimize clock usage. On the other hand, the table shows that the clock-aware implementations dissipate more power within the general purpose routing resources. This is especially true for placers P1 to P4, which use the standard clock cost function. This effect is reduced for placers P5 to P8, which use the gradual clock cost function. In terms of speed, the clock-aware placement techniques only have a small impact with average critical-path delay variations of approximately 1% in either direction. Finally, in this experiment, Placer 8 produces the best overall placements with the lowest overall energy per cycle and the second fastest average critical-path delay. Specifically, Placer 8 is 4.7% more energy efficient than the original VPR placer (which does not produce legal placements) with no increase in the average critical-path delay.

CLOCK NETWORK ARCHITECTURE RESULTS

In the previous section, we compared different clock-aware placement techniques. In this section we use the best clock-aware placer from the previous section (Placer 8) to compare how different clock network parameters affect overall power, area, and delay. Specifically, we consider four clock network parameters: W_{lb} , W_{rib} , W_{global} , and $nx(y)_{region}$. For each experiment, we vary one parameter at a time. Our goal is not to

exhaustively measure the cost of every possible combination of parameters, but rather to examine the impact that each clock network constraint has on the overall power, area, and critical-path delay of system-level applications.

Clocks per Logic Block (W_{lb})

We first consider the flexibility within the logic blocks by varying W_{lb} , the number of clocks per logic block. Intuitively, the larger this number, the simpler the task for the clustering algorithm (since the constraint on the number of clocks per logic block is not as severe); however, the larger W_{lb} , the more power-hungry the clock network will be. Specifically, we would expect that limiting the number of clocks per logic block would reduce the packing efficiency of the clusterer since this limits which blocks can be packed together. To examine this, we packed the benchmark circuits and varied the number of clock signals per logic block between 1 and 3. Table VI presents the results.

The table shows that increasing the number of clocks per logic block increases packing efficiency. However, the efficiency only increases by 1.2% when W_{lb} is increased from 1 to 3. The main reason the impact is small is that the efficiency is already close to 100% when W_{lb} is limited to 1, which leaves little room for improvement.

We next consider how W_{lb} affects the area of the clock network. Fig. 18 shows a breakdown of the clock network area relative to the overall FPGA area. We varied W_{lb} from 1 to 10 and used the baseline value from Table I for the remaining parameters. As shown, the area due to the logic block-to-logic element (LB-LE) switches increases significantly as W_{lb} increases. Recall that a full crossbar network was assumed within the logic block. Therefore, increasing W_{lb} by one adds one extra switch for every logic element in the FPGA.

The area due to the rib-to-logic block (RIB-LB) switches increases as W_{lb} increases from 1 to 6, and then decreases as W_{lb} increases from 7 to 10. This is because the number of switches in a concentrator network is smallest when the number of outputs is either very small or close to the number of inputs. Because of this, the incremental area cost of increasing W_{lb} when W_{lb} is more than half of W_{rib} (the number of clocks in each rib) is small.

Table VI: Logic utilization vs. number of clocks per logic block (W_{lb}).

Benchmark	# BLEs	$W_{lb} = 1$		$W_{lb} = 2$ (Baseline)		$W_{lb} = 3$	
		# CLBs	Packing Efficiency (%)	# CLBs	Packing Efficiency (%)	# CLBs	Packing Efficiency (%)
10lrg	15778	1587	99.4	1583	99.7	1582	99.7
10lrg_reordered	15780	1587	99.4	1582	99.7	1582	99.7
15med	6650	673	98.8	669	99.4	667	99.7
1lrg40sml	10600	1078	98.3	1064	99.6	1062	99.8
2lrg4med80sml	20810	2124	98.0	2092	99.5	2085	99.8
30med	6301	644	97.8	634	99.4	632	99.7
30mixedsize	14087	1427	98.7	1418	99.3	1415	99.6
30seq20comb	17708	1805	98.1	1788	99.0	1781	99.4
3lrg50sml	11832	1212	97.6	1189	99.5	1188	99.6
40mixedsize	14982	1518	98.7	1502	99.7	1500	99.9
4lrg4med16sml	12844	1297	99.0	1287	99.8	1285	100.0
4lrg4med4sml	10146	1022	99.3	1016	99.9	1016	99.9
4lrg	12885	1293	99.7	1291	99.8	1291	99.8
50mixedsize	18676	1889	98.9	1875	99.6	1870	99.9
5lrg35sml	12714	1293	98.3	1277	99.6	1272	100.0
60mixedsize	17775	1808	98.3	1789	99.4	1784	99.6
6lrg60sml	19765	2011	98.3	1983	99.7	1980	99.8
70s1423	15899	1622	98.0	1605	99.1	1598	99.5
70sml	10363	1071	96.8	1049	98.8	1043	99.4
8lrg	19133	1920	99.7	1920	99.7	1918	99.8
lotsaffs	10276	1046	98.2	1031	99.7	1029	99.9
lotsaffs2	8321	839	99.2	837	99.4	835	99.7
Average	-	-	98.5	-	99.5	-	99.7

Note that preliminary clock network area results presented in [Lamoureux 2006] differ from the area results in this paper since we use different buffer sizing and sharing assumptions. Although the conclusions we draw are similar, the area overhead of the clock network is smaller when the new assumptions are modeled compared to when the preliminary assumptions are modeled. As an example, the area of the baseline clock network described in Table I accounts of 4.0% of the overall area (in this paper) and 7.2% (in [Lamoureux 2006]).

Finally, we consider how the W_{lb} constraint affects the energy per clock cycle and the critical-path delay. Table VII presents the overall energy for each circuit and

Table VIII gives a breakdown of this energy and shows how the critical-path delay is affected when the W_{lb} is varied from 1 to 3.

The results indicate that increasing the number of clocks per logic block actually increases (rather than decreases) the overall energy and critical-path delay. Intuitively, increasing W_{lb} should decrease energy and delay since the clusterer can pack logic elements more efficiently when logic elements with different clocks can be packed together. However, both tend to increase. Upon further inspection, the average critical-path delay increases because flip-flops from different clock domains are sometimes packed in the same logic block (during clustering) and the corresponding clock domains are placed in different regions of the FPGA (during placement). When this occurs, flip-flops end up being placed far apart, which leads to increased critical-path delay. This issue could likely be resolved after placement by moving or duplicating logic elements that are causing the delay increase, as described in [Schabas 2003]; however, this feature is not supported in our experimental CAD flow. The overall energy increases for two reasons. First, the added logic block clocks add a significant amount of parasitic capacitance the rib tracks in the clock network. Second, the greater availability of logic block clocks increases the usage of logic block and rib clock resources. Increasing W_{lb} from 1 to 2 increases clock energy by 24.5%, which overshadows the 7.0% energy savings obtained in the routing resources.

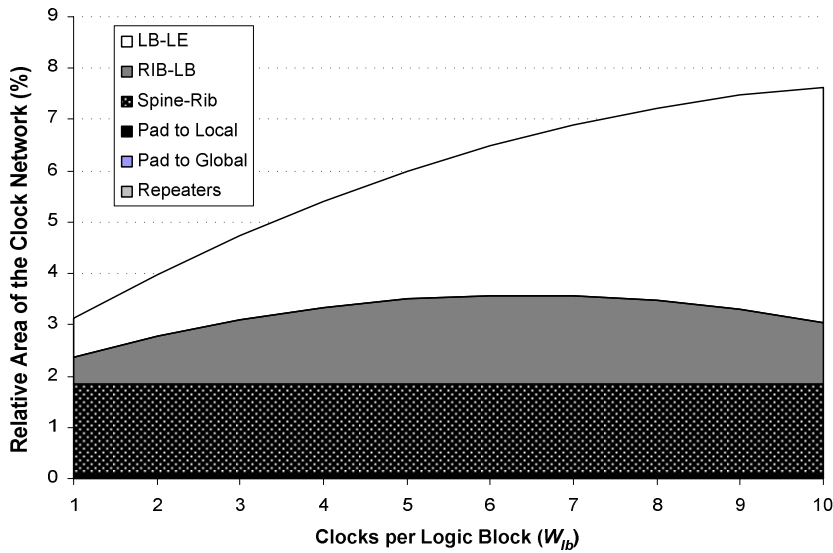


Fig. 18: Clock network area vs. W_{lb} .

Table VII. Overall energy per cycle vs. number of clocks per logic block (W_{lb}).

Benchmark	Overall Energy (nJ)		
	$W_{lb}=1$	$W_{lb}=2^*$	$W_{lb}=3$
10lrg	6.11	6.55	6.74
10lrg_reordered	6.29	6.48	6.58
15med	2.65	2.81	2.89
1lrg40sml	4.43	4.34	4.28
2lrg4med80sml	10.08	11.71	12.28
30med	2.38	2.57	2.67
30mixedsize	6.03	6.65	6.61
30seq20comb	6.64	6.02	6.29
3lrg50sml	4.60	5.17	5.42
40mixedsize	7.08	7.54	7.83
4lrg4med16sml	5.99	6.25	6.33
4lrg4med4sml	4.30	4.37	4.52
4lrg	4.37	4.47	4.40
50mixedsize	9.46	9.95	10.44
5lrg35sml	5.45	5.95	6.14
60mixedsize	8.16	9.00	9.34
6lrg60sml	10.28	11.64	13.69
70s1423	9.01	10.97	11.53
70sml	4.79	4.75	6.25
8lrg	5.33	4.84	4.89
lotsaffs2	3.06	3.36	3.46
lotsaffs	2.11	2.24	2.25
Geomean	5.34	5.65	5.90
% Diff.	-5.45	0.00	4.37

Table VIII: Energy and critical-path delay breakdown vs. clocks per logic block (W_{lb}).

W_{lb}	Overall Energy (nJ)	% Diff	Clock Energy (nJ)	% Diff	Routing Energy (nJ)	% Diff	T_{crit} (ns)	% Diff
1	5.34	-5.5	1.51	-24.5	2.38	7.0	49.2	-1.4
2*	5.65	-	2.01	-	2.22	-	49.9	-
3	5.90	4.4	2.14	6.4	2.25	1.4	50.5	1.2

Clocks per Rib (W_{rib})

We next consider the impact of varying the number of wires in each rib within a region. Intuitively, the placement tool has to ensure that the total number of clocks used by all logic blocks lying on a single rib is no larger than W_{rib} . The higher this value, the easier the placement task becomes, however, a larger value of W_{rib} means the clock network will be larger and consume more power.

Fig. 19 illustrates the clock area when W_{rib} is varied from 6 to 14 and the baseline values from Table I are used for the remaining parameters. The figure shows that the clock network area increases significantly as W_{rib} increases. This is due to an increase in the number of the rib-to-logic block (Rib-LB) switches and spine-to-rib (Spine-Rib) switches. The area of remaining clock network connections remains unchanged and is mostly due to logic block-to-logic element (LB-LE) switches.

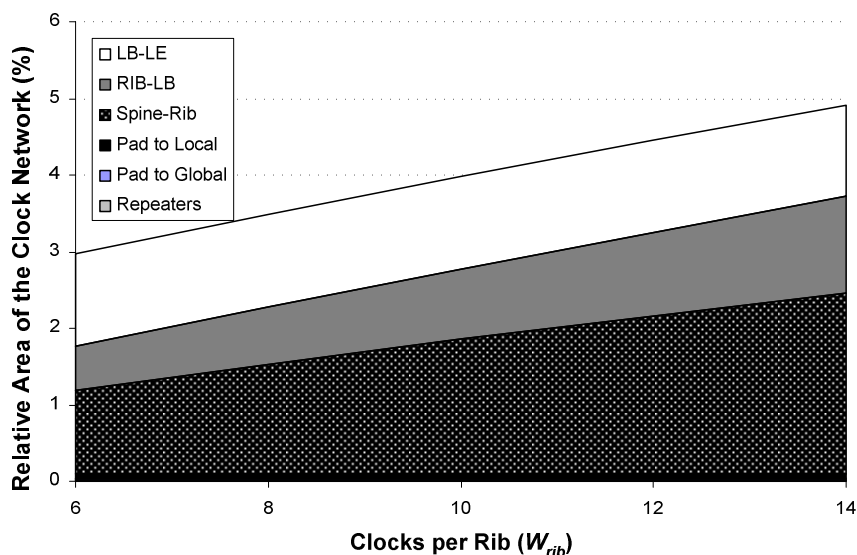


Fig. 19: Clock network area vs. W_{rib} .

Table IX presents the overall energy per cycle when W_{rib} is varied from 6 to 14. The results from this table show that the overall energy decreases only slightly when W_{rib} is increased. In most cases, the majority of the savings are gained when W_{rib} is increased by 1 or 2 tracks beyond the minimum W_{rib} that produces a legal solution. Correspondingly, **Table X** Table X, which gives a breakdown of energy dissipation and critical-path delay with respect to W_{rib} , shows that the energy savings obtained when W_{rib} is increased originate from the savings obtained in the general purpose routing. Unlike the previous case (when W_{lb} was increased), the energy savings obtained in the general purpose routing are not overshadowed by the increase in clock energy dissipation. In terms of critical-path delay, Table X shows that W_{rib} does not have a significant impact on the critical-path delay.

Table IX. Overall energy per cycle vs. the number of clocks per rib (W_{rib}).

Benchmark	Overall Energy (nJ)								
	$W_{rib}=6$	$W_{rib}=7$	$W_{rib}=8$	$W_{rib}=9$	$W_{rib}=10^*$	$W_{rib}=11$	$W_{rib}=12$	$W_{rib}=13$	$W_{rib}=14$
10lrg	6.51	6.46	6.43	6.44	6.55	6.54	6.55	6.55	6.55
10lrg_reordered	6.44	6.44	6.47	6.43	6.48	6.49	6.49	6.48	6.49
15med	2.86	2.86	2.81	2.89	2.81	2.87	2.85	2.84	2.81
1lrg40sml	4.45	4.53	4.39	4.32	4.34	4.35	4.38	4.48	4.45
2lrg4med80sml	-	-	-	-	11.71	11.70	11.70	11.70	11.61
30med	2.59	2.55	2.62	2.59	2.57	2.55	2.58	2.58	2.63
30mixedsize	6.71	6.56	6.56	6.46	6.65	6.38	6.41	6.90	6.29
30seq20comb	6.21	6.16	6.66	6.09	6.02	6.00	6.10	6.16	6.04
3lrg50sml	5.18	4.93	4.90	5.11	5.17	4.96	4.98	5.00	5.00
40mixedsize	7.65	7.63	7.65	7.55	7.54	7.57	7.64	7.46	7.51
4lrg4med16sml	6.29	6.49	6.79	6.35	6.25	6.20	6.17	6.22	6.23
4lrg4med4sml	4.39	4.45	4.46	4.41	4.37	4.41	4.41	4.41	4.41
4lrg	4.47	4.47	4.47	4.47	4.47	4.48	4.48	4.48	4.48
50mixedsize	10.18	9.61	9.53	9.39	9.95	9.49	9.70	9.55	9.49
5lrg35sml	5.97	6.56	6.50	5.92	5.95	6.08	5.97	5.99	5.97
60mixedsize	9.33	8.86	8.92	8.87	9.00	9.19	8.89	8.96	8.91
6lrg60sml	-	-	11.74	11.48	11.64	11.55	11.66	11.60	12.17
70s1423	-	-	-	11.47	10.97	10.75	10.61	10.50	10.51
70sml	4.93	4.79	4.79	4.79	4.75	4.74	4.81	4.76	4.88
8lrg	4.80	4.84	4.85	4.83	4.84	4.83	4.85	4.84	4.84
lotsaffs2	3.57	3.41	3.42	3.44	3.36	3.30	3.32	3.37	3.39
lotsaffs	2.21	2.19	2.20	2.23	2.24	2.28	2.20	2.20	2.19
Geomean	5.12	5.08	5.11	5.04	5.05	5.04	5.03	5.06	5.03
% Diff.	1.3	0.6	1.1	-0.3	0.0	-0.4	-0.4	0.1	-0.4

Table X. Energy and critical-path delay breakdown vs. clocks per rib (W_{rib}).

W_{rib}	Overall Energy (nJ)	% Diff	Clock Energy (nJ)	% Diff	Routing Energy (nJ)	% Diff	T_{crit} (ns)	% Diff
6	5.12	1.3	1.71	-1.5	2.11	4.7	51.1	0.3
7	5.08	0.6	1.72	-1.1	2.07	2.4	51.4	0.9
8	5.11	1.1	1.72	-0.8	2.08	3.3	51.4	1.0
9	5.04	-0.3	1.73	-0.5	2.01	-0.2	51.3	0.7
10*	5.05	-	1.74	-	2.02	-	50.9	-
11	5.04	-0.4	1.73	-0.5	2.00	-0.6	51.2	0.5
12	5.03	-0.4	1.74	0.1	1.99	-1.3	51.3	0.7
13	5.06	0.1	1.74	0.2	2.01	-0.2	51.0	0.2
14	5.03	-0.4	1.74	-0.1	2.00	-1.1	51.3	0.7

Global Channel Width (W_{global})

In this section, we consider the impact of changing the number of clock wires in the spine of the regions ($W_{local} + W_{global}$) on area, power, and delay. Intuitively, a wider spine means more clocks can be distributed within a given region, which makes placement easier but increases the area and power consumption of the clock network.

Fig. 20 illustrates the clock network area when $W_{local} + W_{global}$ is varied from 40 to 120, keeping a 1:1 ratio between W_{local} and W_{global} (matching the baseline architecture in Table I). The graph shows that increasing number clock wires in the spine does increase the area the clock network, but not as significantly as did increasing the W_{lb} and W_{rib} parameters. Most of this area increase comes from the increase in the number of spine-to-rib (Spine-Rib) switches. The number of switches between the clock sources and the spine (Pad to Local and Pad to Global) also increases, but this area is negligible, and is not apparent in the figure.

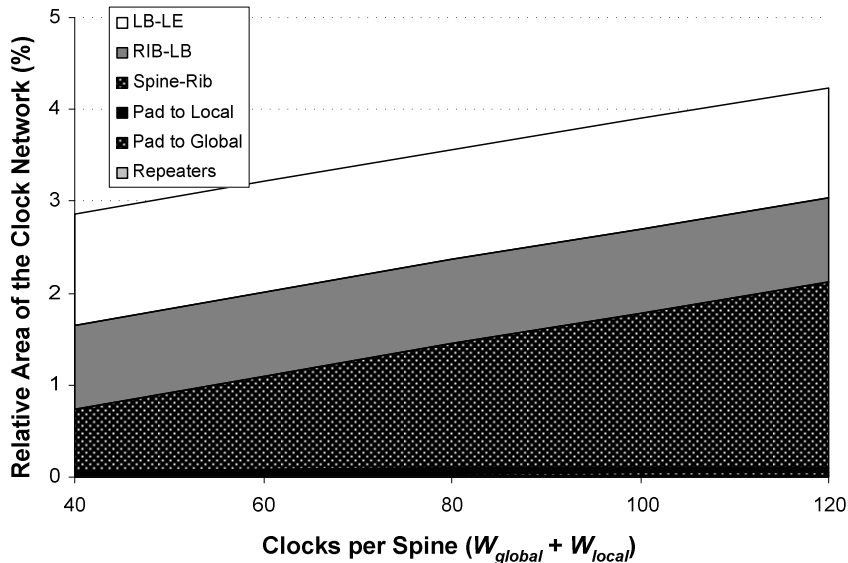


Fig. 20: Clock network area vs. $W_{global} + W_{local}$.

Table XI presents the average overall energy, clock energy, routing energy, and critical-path delay when the number of global clocks (W_{global}) is varied between 0 and 40 clocks and the baseline values are used for the remaining parameters. Results when W_{global} is 0 and 4 are not shown since legal placements could not be found for every circuit. This emphasizes the importance of providing enough global clock network resources. The table also shows that (as long as a legal solution can be found) the number of global clock network resources only has a small impact on the overall energy, clock energy, routing energy, and critical-path delay. Intuitively, the impact on energy is

small since most of the power dissipated by the clock network is dissipated in the ribs and the logic blocks (not the global clock network resources). Therefore, since W_{global} has only a small impact on area, power, and speed, clock networks should be designed with enough global clock resources to ensure that all target applications can be legally placed. These results follow a general trend that clock network flexibility is beneficial and less expensive close to the clock sources (clock pads or internal sources) and become less beneficial and more expensive close to the clock sinks (flip-flops).

Table XI. Energy and delay vs. global channel width (W_{global}).

W_{global}	Overall Energy (nJ)	% Diff	Clock Energy (nJ)	% Diff	Routing Energy (nJ)	% Diff	T_{crit} (ns)	% Diff
52 (baseline)	5.23	0.0	1.81	0.0	2.07	0.0	51.9	0.0
8	5.23	0.0	1.81	0.2	2.08	0.1	52.0	0.2
12	5.25	0.4	1.82	0.5	2.08	0.5	51.9	-0.1
16	5.23	0.1	1.83	0.9	2.06	-0.6	51.8	-0.2
20	5.26	0.5	1.83	1.1	2.08	0.5	51.6	-0.6
24	5.29	1.2	1.83	1.2	2.11	1.8	51.7	-0.4
28	5.27	0.8	1.83	1.2	2.09	0.9	51.6	-0.7
32	5.27	0.8	1.83	1.1	2.10	1.1	51.7	-0.4
36	5.26	0.7	1.83	0.9	2.09	0.9	51.8	-0.4
40	5.27	0.7	1.83	1.1	2.09	0.9	51.8	-0.4

Number of Clock Regions (nx_region , ny_region)

Finally, this section examines how the number of clock regions affects power, area, and delay. As described in Section 0, we assume the FPGA is broken down into $nx_region \times ny_region$ regions, each of which contains its own set of local clocks. A clock network with many clock regions is suitable for implementing applications with many clock domains. In the best case, we can map each domain of a user circuit to a single region of the FPGA. Intuitively, this would save power because each clock would only be routed to minimum number of clock regions.

Fig. 21 shows a breakdown of the clock network when the number of clock regions is 1, 4, 9, and 16 and the baseline values from Table I are used for the remaining parameters. The figure illustrates that increasing the number of clock regions increases the area fairly significantly and that most of the area increase is due to the increase in the number of spine-to-rib (Spine-Rib) switches.

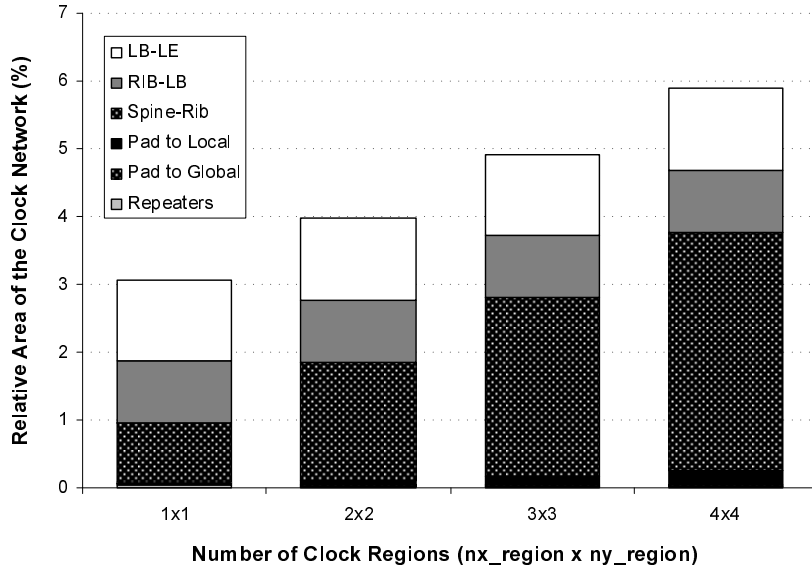


Fig. 21: Clock network area vs. number of clock regions.

Note, however, that as we increase the number of regions and therefore reduce their size, the number of clock domains of the user circuit that are mapped to each region decreases. In the case where each user clock can be placed within one clock region, the number of clocks required in each is inversely proportional to the number of regions. To examine this effect, Fig. 22 shows a breakdown of the clock network when the number of clock regions is 1, 4, 9, and 16, W_{local} is 64, 16, 7, and 4, W_{global} is 16, and the baseline values from Table I are used for the remaining parameters.

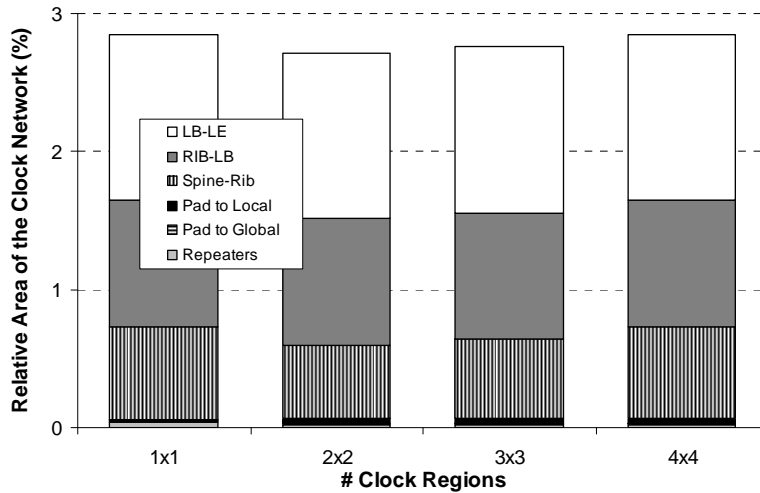


Fig. 22: Clock network area vs. number of clock regions (with adjusted W_{local}).

Fig. 22 shows that the area of the clock network does not increase significantly when the number of clock regions is increased, as long as the number of clocks in the spines is roughly inversely proportional to the number of clock regions.

Finally, we consider how the number of clock regions affects the overall energy. Table XII and Table XIII present the energy results for FPGAs with 1, 4, 9, and 16 clock regions and baseline values are used for the remaining parameters. Note that in the following experiments, the size of the clock regions vary with circuit size. In commercial devices, the size of the clock regions are usually fixed (to facilitate layout) and the number of clock regions vary with FPGA size. However, since our benchmark circuits do not vary widely in size, we believe the conclusions are consistent.

The table reports two different geometric means. The first value is averaged over every benchmark and second value is averaged over the benchmark circuits with valid results in each of the four experiments. Intuitively, increasing the number of clock regions increases the amount of circuitry needed to physically implement the local connections between the clock sources on the periphery of the FPGA to center of each region. At the same time, however, increasing the number of clock regions decreases the size of each region, which reduces the number of tracks used within the clock regions and ribs.

Table XII shows that increasing the number of clock regions significantly reduces overall energy, with average savings of 14.6% when a clock network with 25 regions is used instead of the baseline clock network, which has 4 regions. Moreover, the table also shows that reducing the number of clock regions from 4 to 1 increases overall energy by nearly 30% and makes finding a legal placement more difficult, with 5 failed placements. In each of these cases, the placer failed to find a legal placement since the demand for rib tracks increased beyond the baseline value. Finally, Table XIII shows that the overall energy savings are due in most part to the clock energy savings and that increasing the number of clock regions does not have a significant impact on the average critical path delay.

Table XII. Overall energy vs. the number of regions ($nx_region \times ny_region$).

Benchmark	Overall Energy (nJ)				
	1 (1×1)	4* (2×2)	9 (3×3)	16 (4×4)	25 (5×5)
10lrg	8.21	6.55	7.55	5.71	5.54
10lrg_reordered	8.42	6.48	5.93	5.99	5.62
15med	3.51	2.81	2.73	2.59	2.60
1lrg40sml	5.66	4.34	3.99	3.89	3.85
2lrg4med80sml	-	11.71	10.15	10.04	9.33
30med	3.22	2.57	2.38	2.36	2.31
30mixedsize	8.06	6.65	6.05	5.83	5.61
30seq20comb	8.46	6.02	5.61	5.21	5.21
3lrg50sml	-	5.17	4.36	4.31	4.10
40mixedsize	9.41	7.54	6.89	6.72	6.47
4lrg4med16sml	8.02	6.25	5.88	5.41	5.44
4lrg4med4sml	5.63	4.37	4.06	4.02	3.94
4lrg	6.00	4.47	4.11	3.95	3.99
50mixedsize	12.60	9.95	8.72	8.32	7.94
5lrg35sml	7.69	5.95	5.55	5.42	5.19
60mixedsize	-	9.00	8.16	7.76	7.63
6lrg60sml	-	11.64	10.11	10.02	10.02
70s1423	-	10.97	9.46	9.49	9.10
70sml	-	4.75	4.36	4.34	4.10
8lrg	6.24	4.84	4.26	4.26	4.26
Lotsaffs	4.57	3.36	2.97	2.84	2.72
Lotsaffs2	3.03	2.24	1.96	1.84	1.77
Geomean	5.83	5.65	5.16	4.96	4.82
% Diff	-	0.0	-8.6	-12.2	-14.6
Geomean (Valid)	6.37	4.93	4.60	4.35	4.24
% Diff (Valid)	29.2	0.0	-6.8	-11.8	-14.1

Table XIII: Clock energy, routing energy, and critical-path delay vs. W_{rib} .

# Regions	Overall Energy (nJ)	% Diff	Clock Energy (nJ)	% Diff	Routing Energy (nJ)	% Diff	T_{crit} (ns)	% Diff
4 (2 × 2)	5.65	-	2.10	-	1.98	-	55.3	-
9 (3 × 3)	5.16	-8.6	1.55	-26.5	2.08	4.8	55.2	-0.2
16 (4 × 4)	4.96	-12.2	1.39	-33.9	2.01	1.2	54.9	-0.6
25 (5 × 5)	4.82	-14.6	1.25	-40.6	2.01	1.2	55.0	-0.5

CONCLUSIONS AND FUTURE WORK

This paper presented a new framework for describing FPGA clock networks, described new clock-aware placement techniques, and examined how the FPGA clock

networks and the constraints they impose on the CAD tools affect the overall power, area, and speed of FPGAs.

The framework, which describes a wide range of FPGA clock networks, is key in this research since it provides a basis for comparing new FPGA clock networks and clock-aware CAD techniques. The challenge in creating such a framework was to make it flexible enough to describe as many different "reasonable" clock network architectures as possible, and yet be as concise as possible. In our model, we describe a clock network using seven parameters.

After describing the framework, new clock-aware placement techniques that satisfy the placement constraints imposed on the clock network were described. Several useful clock-aware placement techniques were found. Specifically, we found that simulated annealing can be used to satisfy the placement constraints imposed by the clock network (to legalize the placement). Moreover, we found that the cost function used to minimize the usage of clock network resources and legalize the placement is important. We introduced a gradual cost function which produced implementations that are as energy efficient as those produced using a standard costing approach but aided in finding legal placements. Finally, for FPGAs with local and global clock network resources, we found that using a dynamic approach to assign global clock resources improves overall results and improved the likelihood of finding legal placements.

Using these clock-aware techniques, an empirical study was then performed to examine the impact of the constraints imposed by the clock network. A number of important observations were made. First, the clock network should be more flexible near the clock sources and less flexible near the logic element in order to minimize power. The results showed that adding flexibility near clock sources (near the pads) only slightly increases area and has little effect on overall energy, but makes finding legal placements significantly more straightforward. Moreover, adding flexibility near the logic elements significantly increases area, has little effect on overall energy, and can have a negative impact on critical-path delay. Another important observation is that dividing FPGA clock networks into smaller regions only slightly increases area, but significantly reduces overall energy when implementing applications with many clock domains. On average, FPGAs that have clock networks with 2×2 clock regions dissipated 14.6% more power than FPGAs that have clock networks with 4×4 clock regions for the benchmark circuits in our study.

REFERENCES

- ACTEL. 2007. *ProASIC3 flash family FPGAs datasheet: device architecture* (Jan).
- ALTERA. 2005. *Stratix II Device Handbook 1*, Chapter 2 (March).
- ALTERA. 2006. *Stratix III device handbook 1*, Chapter 6 (Nov).
- BAKOLGU, H.B, 1990. Section 9.4, Machine Organization and Rent's Rule, in: Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley, Reading MA.
- BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. Architecture and CAD for deep-submicron FPGAs. *Kluwer Academic Publishers*.
- BRYNJOLFSON, I., AND ZILIC, Z. 2000. Dynamic clock management for low-power applications in FPGAs. *IEEE Custom Integrated Circuits Conference (CICC)*. 139-142.
- EDAHIRO, M., AND LIPTON, R.J. 1996. Clock buffer placement algorithm for wire-delay-dominated timing model. *Great Lakes Symp. on VLSI*. 143-147.
- FRIEDMAN, E.G. 2001. Clock distribution networks in synchronous digital integrated circuits. *Proc. of the IEEE* 89, 5 (May). 665-692.
- GEORGE, V., ZANG, H., AND RABAEY, J. 1999. The design of a low-energy FPGAs. *Intl. Symp. on Low-Power Electronic Design (ISLPED)*. 188-193.
- LAMOUREUX, J., AND WILTON, S.J.E. 2005. On the interaction between power-aware computer-aided design algorithms for field-programmable gate arrays. *Journal of Low Power Electronics (JOLPE)* 1, 2. 119-132.
- LAMOUREUX, J., AND WILTON, S.J.E. 2006. FPGA clock network architecture: flexibility vs. area and power. *ACM/SIGDA Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, 101-108.
- LAMOUREUX, J., AND WILTON, S.J.E. 2007. Clock-aware placement for FPGAs. To appear in the *Intl. Conf. on Field-Programmable Logic and Applications (FPL)*.
- LANDMAN, B.S., AND RUSSO, R.L. 1971. On a pin versus block relationship for partitions of logic graphs. *IEEE Trans. on Computers C-20*, 12. 1469-1479.
- LI, F., CHEN, D., HE, L., AND CONG, J. 2003. Architecture evaluation for power-efficient FPGAs. *Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*. 175-184.
- NAKAMURA, S., AND MASSON, G.M. 1982. Lower bounds on crosspoints in concentrators. *IEEE Trans. on Computers* 31, 12. 1173-1178.
- POON, K.K.W., AND WILTON, S.J.E. 2005. A detailed power model for field-programmable gate arrays. *ACM Trans. on Design Automation of Electronic Systems (TODAES)* 10, 2 (April). 279-302.
- NATESAN, V., AND BHATIA, D. 1996. Clock-skew constrained cell placement. *Intl. Conf. on VLSI Design*. 146-149.
- SCHABAS, K., AND BROWN, S.D. 2003. Using logic duplication to improve performance in FPGAs. *ACM/SIGDA Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*, 136-142.
- TUAN, T., KAO, S., RAHMAN, A., DAS, S., AND TRIMBERGER, S. 2006. A 90nm low-power FPGA for battery-powered applications. *ACM/SIGDA Intl. Symp. on Field-Programmable Gate Arrays (FPGA)*. 3-11.
- XILINX. 2007. *Virtex-5 user guide*, Chapter 2 (Feb).
- ZHU, K., AND WONG, D.F. 1997. Clock skew minimization during FPGA placement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits* 16, 4 (April). 376-385.