Clock-Aware FPGA Placement Contest

Stephen Yang, Chandra Mulpuri, Sainath Reddy, Meghraj Kalase,
Srinivasan Dasasathyan, Mehrdad E. Dehkordi, Marvin Tom, Rajat Aggarwal
Xilinx Inc. 2100 Logic Drive San Jose, CA 95124
{stepheny,chandim,sainath,meghraj,sda,mehrdad,marvint,rajata}@xilinx.com

ABSTRACT

Modern FPGA device contains complex clocking architecture on top of FPGA logic fabric. To best utilize FPGA clocking architecture, both FPGA designers and EDA tool developers need to understand the clocking architecture and design best methodology/algorithm for various design styles. Clock legalization and clock aware placement become one of the key factors in FPGA design flow. They can greatly influence FPGA design performance and routability. FPGA placement problem can get very difficult with clock legalization constraints. This year's contest is a continuous challenge based on last year's routability driven placement. Contestants need to design best-in-class clock aware placement approach to excel in the contest.

Keywords

FPGA; Placement; Clock; Legalization; Routability

1. INTRODUCTION

As modern FPGA architectures continue to evolve and designs become more complex, FPGA placement remains to be one of the most challenging problems in FPGA design flow [1]. Today's FPGA architecture imposes complicated layout rules during placement stage. The benefit is that designers can ignore the layout details and focus on logical and functional aspect. The tool developers, however, need to improve placement, routing and optimization algorithms to best achieve design goals while meeting architecture constraints. Clock legalization rule, among all the architectural constrains, has major impact on layout quality including design timing performance and routability.

Routability-driven FPGA placement contest held in ISPD 2016 [2] successfully attracted attention from academic research groups. 19 teams registered for the contest and 12 teams submitted final version of the FPGA placement tool. FPGA placement problem was well studied. New algorithms were tested on academic format benchmarks based on modern FPGA architecture. A number of FPGA placement papers have been published [8][9][10][11].

ISPD'17, March 19–22, 2017, Portland, OR, USA.
 ISBN 978-1-4503-4696-2/17/03...$15.00.
DOI: http://dx.doi.org/10.1145/3036669.3038241

The contest of this year is an extension of ISPD 2016 contest. The introduction of clock-aware concept gives placement problem a new challenge. The best placement algorithm needs to find the balance between getting the appropriate clock legalization constraints and optimizing the basic placement quality.

2. FPGA ARCHITECTURE

2.1 FPGA Programmable Blocks

Xilinx FPGAs [3], an example of which is illustrated in Figure 1, consist of an array of programmable blocks of different types, including general logic (CLB), memory (BRAM) and multiplier (DSP) blocks, surrounded by a programmable routing fabric (interconnect) that allows these blocks to be connected via horizontal and vertical routing channels. This array is surrounded by programmable input/output blocks (IO) that interface the chip to the outside world.

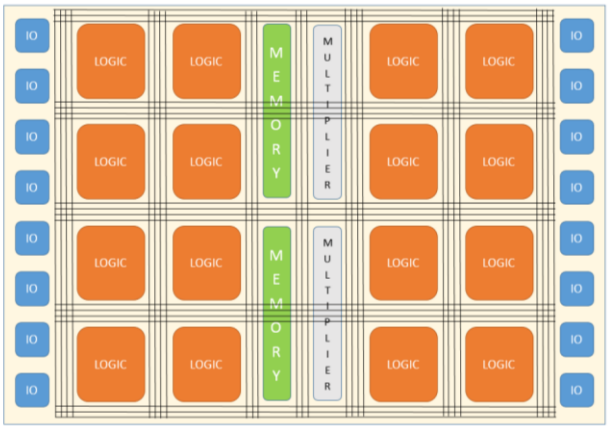

Figure 1. Example of Xilinx FPGA Architecture

This array has a configuration memory (SRAM) beneath it, which, when loaded with appropriate bits, programs the blocks and the interconnects to behave a certain way, as illustrated in Figure 2.

Given a logic design that the user wants to implement on the FPGA, the Xilinx Implementation Tool flow (Vivado) converts the design into the appropriate set of configuration bits (bitstream) which is loaded onto the SRAM to make the FPGA behave as the design. There are usually multiple steps involved in this tool flow, the main ones being, Synthesis, Placement, Routing, and Bitstream generation.

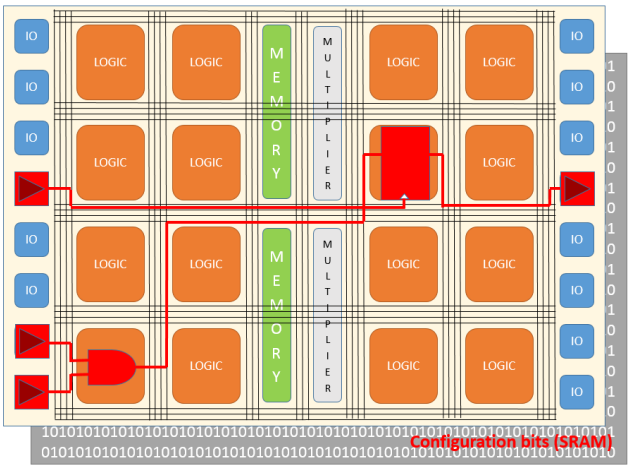

Figure 2. Example of Programming the Xilinx FPGA

Synthesis tool infers the design logic in terms of the logic blocks available within the FPGA. Placement tool places these inferred logic blocks on the various sites of physical logic blocks present in the FPGA. Routing tool connects up the pins of these physical logic blocks using the programmable interconnect routing structures in the FPGA. Bitstream generation tool then proceeds to generate the set of configuration bits that program these logic blocks and interconnect routing structures to behave as the design intended.

The general logic block (also referred to as the configurable logic block, or CLB), is the main resource for implementing general-purpose combinatorial and sequential circuits. The CLB is made up of the logic elements themselves, which are grouped together into a slice. These logic elements are of the type lookup tables (LUTs) or sequential elements (FFs). Each CLB contains one slice. Each slice provides sixteen LUTs and sixteen flip-flops. The slices and their CLBs are arranged in columns throughout the device. There are, however, certain restrictions pertaining to how these LUTs and FFs can be used within each slice. These are explained in detail in the "Placement Evaluation Flow" section under "Legalization Rules" subsection.

In the specific Xilinx FPGA we're targeting for this contest, the XCVU095-ffva2104-es2 device, we have 67,200 CLB/SLICE locations, 880 usable IO locations, 770 DSP locations, and 1730 BRAM locations. More information on this device, and the architecture in general, can be obtained from [5].

2.2 Clocking Architecture

Xilinx UltraScale Architecture introduces a new ASIC-like clocking architecture to the FPGA world. One main feature of this new architecture is the abundance of clocking resources. For example, the biggest device can accommodate more than 600 total clocking buffers. The architecture also introduces a mesh-like routing structure for routing clocks from clock sources all the way to all loads. Such routing structure allows the software tools to make smart choices of how the clocks are placed and routed in a way that have not been feasible in any other FPGA architecture.

The clock placement problem can be stated as the problem of assigning clocking components of a design to compatible clocking resources on a device. In the simplest form, clocking components consist of clock sources and clock loads. Clock sources are components that generate clock signals and/or derive dedicated clock nets using dedicated clocking trees. Clock loads are sequential components that capture data with respect to the input clock signal.

Clock source placement is usually done early in the placement flow along with general IO placement, and it heavily depends on architectural rules imposed by the device constraints. Clock load placement, specifically for non-IO clock loads, is taken care by the general placement flow. This usually starts with a global placement of all placeable components, where an approximate location is found for each component. This is followed by a detailed placement, where a legal placement is created and each component is assigned to a physical site on the device. At early stages in the global placement flow, the clock loads are partitioned based on their placement at the time. The Clock load partitioning is driven by clocking architectural constraints. Without a correct clock load partitioning the final placement solution could be illegal, i.e., no routing solution would be available.
The clock placement and partitioning approach explained above is independent of the how the clocks are routed. But this approach is not enough to create legal clocking solutions. A clock partitioning solution that combines the problems of clock partitioning and clock routing, is needed to produce legal clocking solution and optimize clocking network for better skew, hold requirement, and insertion delay.

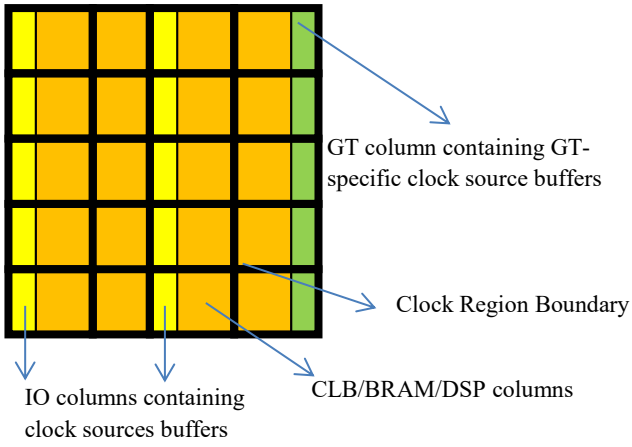

Figure 3. Clock Region Boundaries in an UltraScale device

Each FPGA device in this architecture is divided into multiple clock regions. A clock region includes all synchronous elements--Configurable Logic Block (CLB), I/O, high speed transceivers (GT), DSP, block RAM, and so on-- in an area spanning one I/O bank, with a horizontal clock row (HROW) in its center. Figure 3 shows clock region divisions for one of the UltraScale devices.

This particular device is divided into a 4x5 rectangular grid of 20 clock regions. Note that some clock regions may contain an IO bank or a GT quad. Clock source buffers are inside the IO and GT columns. So clocks can only be sourced from such clock regions.

The clock routing structure consists of a two-layer network of routing tracks as detailed below:

- A routing network consisting of 24 horizontal and 24 vertical tracks

- There is a one-to-one bidirectional connection between any two horizontal routing and vertical routing tracks in each clock region. For example, for a clock using horizontal routing track 0, it can switch to vertical routing track 0 at their intersection in one clock region and back to horizontal routing track 0 in another clock region.
- There is no vertical routing track in IO or GT columns.

- A distribution network, also consisting of 24 horizontal and 24 vertical tracks
  - There is a one-to-one unidirectional connection (from vertical to horizontal) between any two horizontal distribution and vertical distribution tracks in each clock region. For example, for a clock using vertical distribution track 0, it can switch to horizontal distribution track 0 at any possible intersection.
  - There is no vertical distribution track in GT columns.

There is no path from distribution back to routing tracks. So once a clock is on the distribution network it can only go to the leaf level nodes. From routing (horizontal or vertical) to distribution network clocks need to hop onto vertical distribution first. There is a one-to-one connection from every routing (horizontal or vertical) to its corresponding vertical distribution track.

The clock can be distributed from the sources in one of two ways. They can go onto routing tracks which take the clock to a particular sub-region without going to any loads and then go onto the distribution tracks. This is used to move the root for all the loads to be at a location beneficial from a skew perspective. Alternatively, they can go straight onto the distribution tracks. This would be to reduce insertion delay or that point being the root is most beneficial for skew. Once on the distribution tracks, the clock travels vertically and taps off at various horizontal segments. Before driving the horizontal segment it would go through a programmable delay and clock enable circuit. From the horizontal distribution it can feed the leaf clocks.

Each clock segment can be driven at either end or by a driver within the segment. Each of those drivers therefore would be tri-stable. This allows the clock network to be segmented at each fabric sub-region boundary. By having the clock only use segments as needed, it allows the tracks to be reused.

The clock routing structure consists of a two-layer network of routing tracks as detailed below:

- A routing network consisting of 24 horizontal and 24 vertical tracks
- A distribution network, also consisting of 24 horizontal and 24 vertical tracks
- Each clock region has 24 horizontal routing (HR) and 24 vertical routing (VR) tracks
- Each clock region has 24 horizontal distribution (HD) and 24 vertical distribution (VD) tracks
- There is a one-to-one bidirectional connection between any two HR and VR tracks in each clock region. For example, for a clock using HR track 0, it can switch to VR track 0 at their intersection in one clock region and back to HR track 0 in another clock region
- There is a one-to-one unidirectional connection from VD to HD in each clock region. For example, for a clock using VD track 0, it can switch HD track 0
- Once on HD, clock only drives HD tracks on neighboring clock regions or clock loads in that region. There is no way back on HR/VR/VD tracks
- From HR/VR to distribution network clocks need to hop onto VD first. There is a one-to-one connection from every routing (horizontal or vertical) to its corresponding VD track
- All tracks are segmented at clock region boundaries, therefore two clocks can use the same track provided that their loads are in non-intersecting rectangular clock region areas
- Each clock net should use a single clock track and a single clock root
- Each global clock buffer has a dedicated clock track that can only be driven by that clock buffer. The Y coordinate of the site where the clock buffer is placed at can be used to specify the track number for that given site. So for BUFGCE_XmYn the clock track number will be n%24
- Within a clock region, global clock buffer locations can be changed without affecting design legality

2.3 Clock Placement Problem

Place all clock sources and clock loads and partition the clock loads into partitions containing one or more clock regions, such that

- Number of global clocks in each clock region is at most 24 clocks.
- Within each clock region, each half column has at most 12 clocks.
- Each clock region has enough resources to accommodate all clock loads assigned to that region.
- If needed, all loads of each clock should be constrained to a continuous rectangular area consisting of one or more clock regions.

3. BENCHMARKS

The benchmarks for ISPD 2017 clock placement contest have been generated using an internal netlist-generation tool based on Generate NetList (GNL [7]). The tool allows us to create netlists which varies in features such as number of components, interconnection, number of control sets, number of clocks. Additionally, it provides control over the type of components (primitives) used in the netlist. For ISPD 2017 benchmarks, we have restricted the primitives to be Look-Up-Tables (LUTs), Flip-Flops (FFs), DSP blocks (DSPs),

Design Name	#Luts (Util)	#Flops (Util)	#RAMB36	#DSPs	#IOs	Rent	#Clocks
design5	215K(40%)	236K(22%)	170(10%)	75(10%)	300	0.6	30
design6	242K(45%)	270K(25%)	255(15%)	112(15%)	300	0.6	33
design7	268K(50%)	300K928%)	340(20%)	150(20%)	300	0.6	36
design8	295K(55%)	325K(30%)	425(25%)	187(25%)	300	0.6	39
design9	322K(60%)	354K(33%)	510(30%)	225(30%)	400	0.63	42
design10	350K(65%)	384K(36%)	595(35%)	262(35%)	400	0.63	45
design11	376K(70%)	414K(38%)	680(40%)	300(40%)	400	0.63	48
design12	392K(73%)	431K(40%)	765(45%)	337(45%)	400	0.63	51
design13	408K(76%)	449K(42%)	850(50%)	375(50%)	400	0.63	54
design14	424K(79%)	450K(43%)	900(53%)	397(53%)	400	0.63	55
design15	440K(82%)	484K(45%)	950(56%)	420(56%)	400	0.63	56
design16	456K(85%)	503K(47%)	1000(59%)	442(59%)	400	0.63	57

Table 1. Benchmark statistics

*Number in parenthesis indicates the utilization as percentage of available resources in the FPGA [6]

and Block RAMs (BRAMs). We have varied number of primitives of different types, interconnection complexity, number of clocks to create netlist of varied complexity. The target device chosen was xcvu095, part of the Virtex UltraScale [4] family.

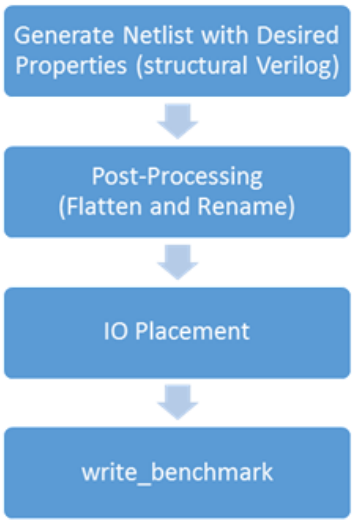

Figure 4. Benchmark Generation Flow

The following properties of the netlist were varied among the ISPD benchmarks.

- Number of instance: We have created benchmarks that utilize between 40% and 85% of the available LUTs. Proportionally we have also varied the number of DSPs, BRAMs, and FFs to create medium to highly utilized designs.
- Rent exponent: Interconnection complexity were varied to create netlists of different Rent exponents. This is important to test the routability aspect of the placement solution.
- Number of clock: Further complex designs were created by varying the number of clocks from 30 to 57.
- Number of Resets: FPGA architecture limits the number of unique reset nets per Slice. By varying the number of resets we test how well the placer can support such restrictions.

Figure 4 explains the flow used for generating these benchmarks. First, we generate structural Verilog using our netlist-generation tool. The input to this tool is a configuration file, which specifies the desired parameters in the netlist. The structural Verilog file is post-processed to create a flattened design, without any hierarchies. Along with dissolving hierarchies, we also rename the instances and nets in this step. Next, we run Vivado placer to place IO ports of the design. Finally, we write the benchmark in Bookshelf format. The Bookshelf format list the instances in the design in a ".nodes" file and their interconnection in a ".nets" file. It also writes IO placement in a ".pl" file. Library cells are separately listed in a "*.lib" file.

4. PLACEMENT EVALUATION

The placement evaluation flow is similar to ISPD 2016 contest [2]. The major difference is on the clock legalization check. Total wirelength is the main evaluation metric. Clock skew and timing are not part of the metrics in this contest.

4.1 Placement Interface

Contestants are expected to write the output of their placement tool in a specific (.pl) file format. Placer's output placement file should contain locations of all the instances in the design. The location of an instance has three fields: x-coord, y-coord (to determine the SITE) and BEL (index within the SITE). Figure 5 shows the BEL number for LUTs/FFs placed inside a SLICE SITE.

For BRAM and DSP instances, since there are no BELs within a SITE, the BEL index remains 0.

Figure 5. BEL offsets within a SLICE

The placement output (.pl) file, will be given as an input to Xilinx Vivado tool using the flow.tcl file, which is available as part of each benchmarks archive. Vivado Placer will then read these instance placements, and check for legal placement on every instance. In case of illegal placement, Vivado Placer will error out with a reason behind the illegality for each instance. If the placement is legal, Vivado router starts and completes routing, or report unroutable design. If routing completes successfully, the following message indicates total routed wirelength: "Total Routed Wirelength: xxxxx (Vertical xxxx, Horizontal xxxx)". In case of unroutable placement, the following message shows up: "CRITICAL WARNING: [Route 35-162] xxxx signals failed to route due to routing congestion."

4.2 Legalization Rules

Each SLICE site provides sixteen LUTs and sixteen FFs. There are, however, certain restrictions pertaining to how these LUTs and FFs can be used within each SLICE.

4.2.1 Clock Legalization Rules

- Number of global clocks in each clock region is at most 24 clocks.
- Within each clock region, each half column has at most 12 clocks.
- Each clock region has enough resources to accommodate all clock loads assigned to that region.

4.2.2 Using LUTs in a SLICE:

- The 16 LUTs within SLICE are conceptual LUTs that can only be fully used under certain conditions:
- When implementing a 6-input LUT with one output, one can only use LUT 1 (leaving LUT 0 unused) or LUT 3 (leaving LUT 2 unused) or ... or LUT 15 (leaving LUT 14 unused)
- When implementing two 5-input LUTs with separate outputs but common inputs, one can use {LUT 0, LUT 1} or {LUT 2, LUT 3} or ... or {LUT 14, LUT 15}
- The above rule of coming LUTs with separate outputs but common inputs, holds good for 5-input LUTs (as mentioned above) or fewer input LUTs as well
- When implementing two 3-input (or fewer input) LUTs together (irrespective of common inputs), one can use {LUT 0, LUT 1} or {LUT 2, LUT 3} or ... or {LUT 14, LUT 15}

4.2.3 Using FFs in a SLICE:

- There are 16 FFs per SLICE (two per LUT pair), and all can be used fully under certain conditions:
- All FFs can take independent inputs from outside the SLICE, or outputs of their corresponding LUT pair (FF 0 can take LUT 0 or LUT 1 output as input, ..., FF 15 can take LUT 14 or LUT 15 output as input)
- All can be configured as either edge-triggered D-type flip-flops or level-sensitive latches. The latch option is by top or bottom half of the SLICE (0 to 7, and 8 to 15). If the latch option is selected on a FF, all eight FFs in that half must be either used as latches or left unused. When configured as a latch, the latch is transparent when the clock input (CLK) is high.
- There are two clock inputs (CLK) and two set/reset inputs (SR) to every SLICE for the FFs. Each clock or set/reset input is dedicated to eight of the sixteen FFs, split by top and bottom halves (0 to 7, and 8 to 15). FF pairs ({0,1} or {2,3} or ... or {14,15}) share the same clock and set/reset signals. The clock and set/reset signals have programmable polarity at their slice inputs, allowing any inversion to be automatically absorbed into the CLB.
- There are four clock enables (CE) per SLICE. The clock enables are split both by top and bottom halves, and by the two FFs per LUT-pair. Thus, the CEs are independent for: {FF 0, FF 2, FF 4, FF 6}, {FF 1, FF 3, FF 5, FF 7}, {FF 8, FF 10, FF 12, FF 14}, {FF 9, FF 11, FF 13, FF 15}. When one storage element has CE enabled, the other three storage elements in the group must also have CE enabled. The CE is always active High at the slice, but can be inverted in the source logic.
- The two SR set/reset inputs to a SLICE can be programmed to be synchronous or asynchronous. The set/reset signal can be programmed to be a set or reset, but not both, for any individual FF. The configuration options for the SR set and reset functionality of a register or latch are: No set or reset, Synchronous set (FDSE primitive), Synchronous reset (FDRE primitive), Asynchronous set (preset) (FDPE primitive), Asynchronous reset (clear) (FDCE primitive). The SR set/reset input can be ignored for groups of four flip-flops (the same groups as controlled by the CE inputs). When one FF has SR enabled, the other three FFs in the group must also have SR enabled.
- The choice of set or reset can be controlled individually for each FF in a SLICE. The choice of synchronous (SYNC) or asynchronous (ASYNC) set/reset (SYNC_ATTR) is

controlled in groups of eight FFs, individually for the two separate SR inputs.

Some of these FF Packing rules are illustrated in Figure 6.

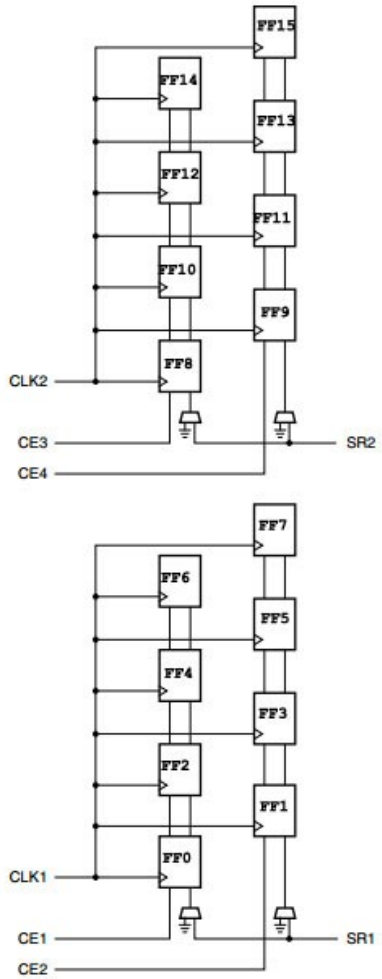

Figure 6. Flip Flop control signals connectivity within a SLICE

More information on the CLB composition can be obtained from [5]

4.3 Evaluation Metrics

- For each design in the benchmark suite, the placers will be ranked based on the contest evaluation metric. The final rank for a placer will be the sum of the individual ranks on all the circuits. The placer with the smallest total rank wins the contest.
- The placement runtime must be 12 hours or shorter.
- The placement must be legal in terms of logic legalization.
- The placement must be legal in terms of clock legalization rules described in Section 2.
- The placement has to be routed by Vivado router, and the router has to complete the job within 12 hours. Routing is regarded as failed if it takes more than 12 hours to complete.
- PlacementScore=RoutedWirelength*(1 + Runtime_Factor)
  - Vivado router reports total routed wirelength. This is the base of the score.
  - Total placement and routing runtime will be used in computing P&R_Runtime_Factor;
  - Runtime_Factor= -(Runtime - Median_Runtime) / 10.0
  - Runtime factor is between -10% and +10%
- The failed place/route job will get the lowest rank on this design. In the presence of failures on multiple placers, the break-tie factors are (in order): placer failure, logic legalization failure, clock legalization failure, router failure.

5. ACKNOWLEDGMENTS

The authors would like to thank Dr. Sudip Nag, Dr. Padmini Gopalakrishnan and Dr. Sabya Das for their support.

6. REFERENCES

[1] R. Aggarwal, FPGA Place and Route Challenges, *International Symposium on Physical Design. 2014*

[2] S. Yang, A. Gayasen, C. Mulpuri, S. Reddy, and R. Aggarwal, Routability-Driven FPGA Placement Contest, *International Symposium on Physical Design, 2016*

[3] Xilinx, "UltraScale Architecture", http://www.xilinx.com/products/technology/ultrascale.html

[4] Xilinx, "Virtex UltraScale FPGAs", http://www.xilinx.com/publications/prod_mktg/ultrascalevirtex-product-table

[5] Xilinx, "UltraScale Architecture Configurable Logic Block User Guide", http://www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf

[6] Xilinx, "UltraScale Architecture and Product Overview", http://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[7] GNL: http://users.elis.ugent.be/~dstrooba/gnl/

[8] W. Li, S. Dhah, and D. Z. Pan, "UTPlaceF: A Routability-Driven FPGA Placer with Physical and Congestion Aware Packing", *International Conference on Computer-Aided Design (ICCAD), 2016*

[9] C.W. Pui, G. Chen, W.K. Chow, K.C. Lam, J. Kuang, P. Tu, H. Zhang, F.Y. Young, B. Yu, "RippleFPGA: A Routability-Driven Placement for Large-Scale Heterogeneous FPGAs", *International Conference on Computer-Aided Design (ICCAD), 2016*

[10] R. Pattison, Z. Abuowaimer, S. Areibi, G. Grewal, A. Vannelli, "GPlace – A Congestion-aware Placement tool for UltraScale FPGAs ", *International Conference on Computer-Aided Design (ICCAD), 2016*

[11] S. Dhah, S. Adya, L. Singhal, M. A. Iyer and D. Z. Pan "Detailed Placement for Modern FPGAs using 2D Dynamic Programming", *International Conference on Computer-Aided Design (ICCAD), 2016*