International Journal of Engineering & Technology, 7 (2.11) (2018) 31-35

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Optimization of Physically-Aware Synthesis for Digital Implementation Flow

Leo E. Geralla[1*], Melvin Joey de Guzman[2], Jefferson A. Hora[3]

[1,3]Mindanao State University-Iligan Institute of Technology, Iligan City, Philippines
[2]Lattice Semiconductor Corporation, Muntinlupa City, Philippines
**Corresponding author E-mail: lgeralla@gmail.com*

Abstract

Synthesis is very important to have a high-quality implementation of every design. However, more accurate results could not be achieved if we will not consider the expected effects of routing delay introduced by placement and routing. This delay causes the poor timing correlation between the logical-only synthesis and Place and Route. . Now, tools with physical aware synthesis allow the user to integrate the physical information much early in the process. While such technique is readily available in the tools itself, there is no established flow to utilize the use of physical aware synthesis to the whole ASIC design process. Moreover, there's lack of in-depth experimental analysis, specifically on commercially available designs, on the correlation of physically aware synthesis to the subsequent steps in the backend of the whole design process such as the place and route (PnR) and Timing Closure (STA). With this study, optimal flow for synthesis run is achieved through several experimental setups. Effects in place and route (PNR), and Static Timing Analysis (STA) is also observed and documented. Two different physically aware synthesis methodologies are proven to have improved timing correlation between the synthesis and PNR results. Power after signoff also improved significantly. Total runtime from synthesis to timing closure reduces because of much lesser violations in the first iteration alone.

Keywords: *Physically-aware Synthesis; synthesis; PAM; PAS; PLE*

1. Introduction

Two major problems with technology scaling, according to Jaggi Balasubramanian [1], are the increased in the complexity of design and the challenges need to overcome with the interconnect design. As the process technology shrinks, the delay equation is dominated by wires and it will only get worse with each process design. Alpert, C.J. Karandikar, S.K., et. al, said through their paper [2] that, however, technology scaling caused the increasing delay of wire relative to gate delay. Consequently, a design that meets timing requirements in the traditional synthesis alone will not closed much likely once its physical footprints and attributes is realized , due to increased significant of wire delays. Routing congestion becomes more and more difficult to handle. Routing congestion is a recognized problem in traditional RTL synthesis design. For technologies of 0.25 um and below, interconnect capacitances becomes more dominant with respect to gate capacitance. Resolution of routing congestion requires additional routing spaces. During physical design, nets are routed manually one-by-one in an attempt to bypass the congested area. As a result, designers opted to long wiring detours and increased overall wire length and delay [15]. Thus, the die size grows and the timing worsens as those nets are forced to bypass the congested area. Routing congestion can be categorized into global and local [16]. Global interconnect congestion occurs when there are a lot of chip-level or inter-block wires that need to cross an area. Local congestion stems from two sources: placement density and logic-induced congestion. The former is caused by the placement of the cells which are too close. The latter is due to logic structure and cell selection. Some of these problems can be mitigated by efforts to explore better floorplans or with physically aware logic synthesis. However, most of logic-induced congestion is difficult to eliminate without modifying the design itself. Hence, by Lin Zhong and Niraj K. Jha [14] suggests an HLS methodology that detects congestion early, annotates root causes in terms of the input description (SystemC) and lets designers perform the design modifications manually to eliminate congestion sources. To decrease the routing congestion and improve circuit's performance, Wu J., Ma C., and Huang propose a routing congestion aware high level synthesis method, which is based on simulated annealing (SA) algorithm. The method has two loops, the inner loop is a practical iteration, and the outer loop is a gradually cooling process. After the high level synthesis, Wu J., Ma C., and Huang accept or reject the new result according to the cost function value which is obtained by fore-placement. Experimental results show that their method is more effective than the traditional method and CRSF (Congestion driven re-synthesis after floor planning) algorithm. [17] Another way of addressing the congestion problem in synthesis was also presented by Pandini D., Pileggi L., and Strojwas [15] in their paper. They focus on first addressing the problem of congestion minimization in logic synthesis during technology mapping. It is only at this stage that it is possible to exploit placement information to capture the connectivity of the technology independent representation of the circuit consisting of base functions, and then explore the trade-offs between area (and/or delay) and congestion minimization when a fixed amount of routing resources are available.

Tasuoka M, Watanabe R.,Otsuka T., and Hasegawa T., [13] propose a novel design flow by integrating a High-level synthesis (HLS) tool with physically aware logic synthesis technology to a

precisely detect logic-induced congestion and its root causes in the systemC model. The novelty of their approach comes from the combination of three technology ideas which includes the following the integration of high level synthesis and congestion analysis based on the physical attributes that is consistent with the backend physical view, localization of congestion violations to a small number of RTL instances such as multiplexers (MUXes) and demultiplexers (DEMUXes) and annotation technology provison in high level synthesis flow that relates RTL objects to SystemC [13]. Another paper by Lin Zhong and Niraj K. Jha [16] brings the concept of physically aware high level synthesis by providing a comprehensive treatment of interconnect-aware high level synthesis for low power. Interconnects (wires, buffers, clock distribution networks, multiplexers and busses) consume a significant fraction of total circuit power. Lin Zhong and N.K Jha demonstrated the importance of optimizing on-chip interconnects for power during high-level synthesis. They developed a binding algorithm which not only reduces power consumption in functional units and registers in the resultant register-transfer level (RTL) architecture, but also optimizes interconnects for power. Physical design information is taken into account making it a physically-aware innovation [14].

Synthesis part is very important to have a high-quality implementation of every design. However, more accurate results could not be achieved if we will not consider the expected effects of routing delay introduced by placement and routing. This delay causes the poor timing correlation between the logical-only synthesis andPlace and Route, Alpert, C.J, et.al added. [3].Currently, several electronic design companies introduce physically aware synthesis to address the complications caused by increasing wire delays. New products from several vendors are being used for the great improvement over the simple RTL logic synthesis used in a "legacy" ASIC design flow as illustrated by Moxon Design [9]. Parimi N. from Cadence [4] even said that there so much pressure in design engineers in a team to deliver faster, smaller devices in shorter times which opted them to develop an integrated methodology to incorporate physical design information during synthesis. Previously, wire-load models that are used by tools created by EDA companies are fan-out based. Using this old technique for shrinking technologies, Balasubramanian J. [1] said that it will only lead to significant differences of the Quality of Results (QoR) of the synthesis and implementation tools. Now, tools with physical aware synthesis allow the user to integrate the physical information much early in the process. While such technique is readily available in the tools itself, there is no established flow to utilize the use of physical aware synthesis to the whole ASIC design process. Moreover, there's lack of in-depth experimental analysis, specifically on commercially available designs, on the correlation of physically aware synthesis to the subsequent steps in the backend of the whole design process such as the place and route (PnR) and Timing Closure (STA).

Now, the challenge of this paper is to present a flow that will allow designer to integrate physically aware innovations of the tool into the Methodology design kit (MDK) with optimum settings to utilize the better effects of physically aware innovation to synthesis and other subsequent steps of the design process. Moreover, this will provide thorough analysis on results of using physically-aware netlist in place and route and static timing closure.

2. Logic Synthesis

2.1 Definition

It is a process by which an abstract form of desired circuit behaviour, mostly in form of RTL (Register transfer level) is turned into a design implementation in terms of logic gates. ,It is usually done by EDA tool.

Synthesis tool, after mapping the RTL to gates, also do the minimal amount of timing analysis to see if the mapped design is meeting the timing requirements. (Important thing to note is, synthesis tools are not aware of wire delays, they only know of gate delays). After the synthesis there are a couple of things that are normally done before passing the netlist to backend (Place and Route). [5] [6]

2.2 Synthesis Gap

As process technology shrink, changes in the characteristics of wire interconnects make it much more difficult to achieve optimal timing and closure. Previously, delays are significantly driven by standard cells itself and the fan-outs. As we all know, using fan-out based wireload models to provide wire delay information leads to significant differences in Quality of Results (QoR) between the synthesis and the implementation tools. [7]

Now, synthesis stage alone requires to be aware of the physical aspects of the designs, such as design rules, placement and routing, for better delay calculations and at the same time should also reduce the effort needed in achieving timing closure for the backend designers.

3. Physically-Aware Synthesis

With smaller process geometries and increasing chip complexity, the traditional separation of logic synthesis and physical design leads to many iterations for design closure. Thus, the need to incorporate physical effects into gate level logic synthesis by including process technology data and physical floor planning data, such as cell placement and metal routing layer information [10].

Different techniques have been developed to enhance the traditional synthesis flow. This leads to the early development of physically aware synthesis. Physically aware synthesis makes use of the actual design and physical library information. It also calculates wire delays for different logic structure in the design resulting to a better correlation with the results of place and route. There are techniques developed that are common in most EDA tools according to Gopi Kudva [11]. Three of these are the Physical Layout Estimation (PLE), Physically-aware Mapping (PAM) and Physically-aware Structuring (PAS).

3.1. Physical Layout Estimation

The most basic technique that can be used to run a physically-aware synthesis is the physical layout estimation (PLE). When enabling PLE, parasitic resistance and capacitance values from LEF libraries are being used to calculate more realistic wire delays instead of the values from the synthesis technology library. Today, PLE is only innovation being used in some semiconductor companies [8][12].

3.2. Physically-Aware Mapping

While PLE is the basic technique for having physically aware netlist, another technique focuses more on improving timing with increased correlation. Specifically, with PAM, optimized generic gates and macros are initially placed. The placed generic gates and macros are optimized through RTL level optimization, including datapath optimization. Also, routes and congestion for the placed generic gates and macros are estimated taking into account the physical constraints such as placement and routing blockages. Using those estimated routes, parasitic extraction such as the resistance and capacitance is being done using a unique extraction method [8][11].

3.3. Physically-Aware Structuring

Another technique but focuses on congestion issues and better placement is the physically-aware structuring. Specifically, PAS optimizes binary/one-hot multiplexer (mux) selection. It also targets high-congestion structures such as cross bars, memory-

connected multiplexer chains and barrel shifters. Additionally, if there's a large mux, it is being decompose into a set of smaller mux, and each of which can have the same decode logic. Moreover, decoding logic intelligently partitioned using physical input pin knowledge. Lastly, generated decode islands were congestion aware because of smarter select line sharing and duplication. [8][11]

3.4 Physically-Aware Synthesis Flow

Some additional steps in the traditional synthesis methodology are added to come up with the proposed physically-aware synthesis as shown in Fig. 1. After reading the general setup and the user input files such as the timing constraints, synthesis libraries, etc, and elaboration of the design, floorplan was also being fed into the synthesis tool before the generic synthesis even started. This now lays the basic foundation flow for physically-aware synthesis and is used in creating the different physical flow to obtain optimal timing results which will be shown in methodology section of this paper.

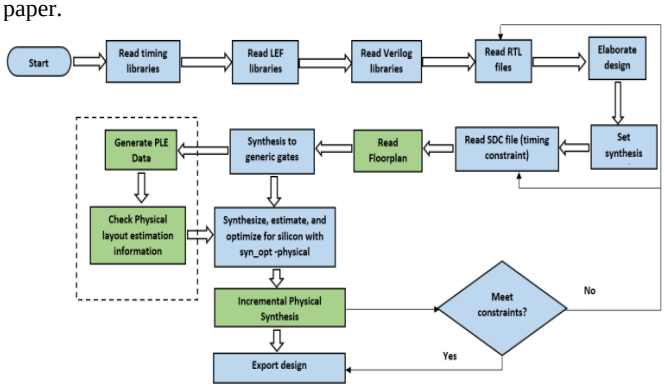

Fig. 1. Proposed Physically-Aware Synthesis Flow

4. Methodology

This section will discuss the methods to be used for the implementation of the study.

4.1 Process Flow

New products by several vendors are being used for improvement of the logic design process flow over time. Proposed process flow shown in Figure 2 showed integration of physically-aware synthesis in a digital design flow. Physical synthesis design flows such as shown in Figure 1 incorporate physical effect into logic synthesis by including process technology data and physical floor planning data.

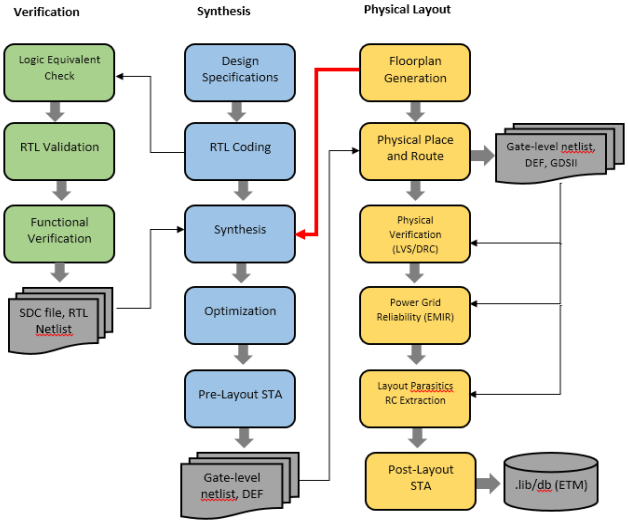

Fig. 2. Proposed Logic Design Process Flow

4.2 Tools

Simulations are limited to the use of specific tools in doing the proposed Logic Design Process Flow as shown in Figure 1 which includes the following processes: synthesis, place and route (PNR), layout parasitic extraction (lpe) and static timing closure (STA).Genus was used for synthesis simulations and Genus-physical tool was used for physical synthesis. For place and route, encounter digital implementation (EDI) tool was utilized. To confirm physical results, Tempus was used for post-simulation results and static timing analysis(STA). In addition, Quantus QRC was used for low parasitic extraction which was needed STA analysis.

4.3 Synthesis Tool

Genusis the tool used in this study. Genus physical is an innovation from cadence that introduces different techniques to have a physically-aware synthesis netlists. There are different attributes and setups available for designers to use when having synthesis run. That is why having the best setup flow in order to optimize the use of physically aware synthesis is needed. Optimum setting for the physically-aware synthesis run affects the results in each stage of the whole ASIC design flow. [7]

4.4 Methodology Design Kit

Methodology design kits (MDK) are an automated and reference scripts to perform certain specific processes such as synthesis, place and route, and static timing closure (STA). A methodology of a sample MDK from Lattice is shown in Figure 3. TheMDK takes designers input such as the RTL codes, Synopsys Design Constraints (SDC) file, and technology libraries, etc, in performing those processes to meet different design specifications specific in each process.

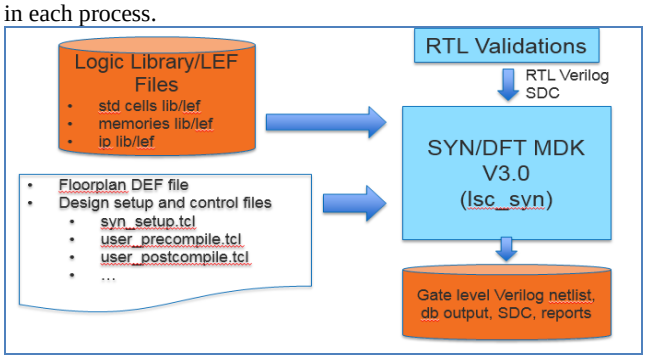

Fig. 3. Lattice Synthesis Methodology with MDK [13]

4.5 Attributes Affecting Physical Flow

An attribute is a setting that controls how the tool operates on objects during different processes, for example, like synthesis and technology mapping. Objects are anything that the tool can manipulate such as libraries, designs, constraints, sub designs, etc. Few attributes that will affect physical flow were analyzed and listed. The types of attributes we used basically define the five (5) physically-aware setups we used in this study. [8] Syntax of the attributes used is shown is Figure 4.

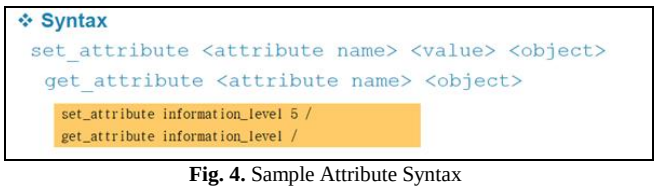

Fig. 4. Sample Attribute Syntax

4.6 Physically-aware Methodologies

The basic physically-aware synthesis flow shown in figure 1 was enhanced through the addition of different physical attributes in the setup codes and main setup files. Five physical setups were finalized and used in this study. First physical setup is the default flow as generated by the tool used by Lattice. Second physical setup was created using the existing physical default values of the synthesis tool vendor. These two methodologies were used to test if it's considerable to use default values. Proving this as false would help us conclude that physically-aware flow will not benefit if the flow is not considered well in order to achieve optimal timing results. After carefully studying and testing different attributes, we coded three physically-aware setups.
The third physical setup has flow that considered no legalize placement while the fourth one considered legalized placement during the synthesis run. Final physical setup focuses only on meeting timing requirements. Different setups were simultaneously run during the whole duration of the experimental setup.

4.7 Design Under Test

A configurable (CFG) logic block for FPGA is used in this project. CFG block is the fundamental component of an FPGA, allowing the user to virtually implement any logical functionality within the chip.
These blocks contain the logic for the FPGA. In the large-grain architecture used by all FPGA vendors today, these CLBs contain enough logic to create a small state machine. The block contains RAM for creating arbitrary combinatorial logic functions, also known as lookup tables (LUTs). It also contains flip-flops for clocked storage elements, along with multiplexers in order to route the logic within the block and to and from external resources. The multiplexers also allow polarity selection and reset and clear input selection.[9]

5. Results

5.1 Synthesis

The slack associated with each connection or path is the difference between the required time and the arrival time. Negative slack implies that a path is too slow if the circuit is to work at a certain speed/frequency making it a violating path:

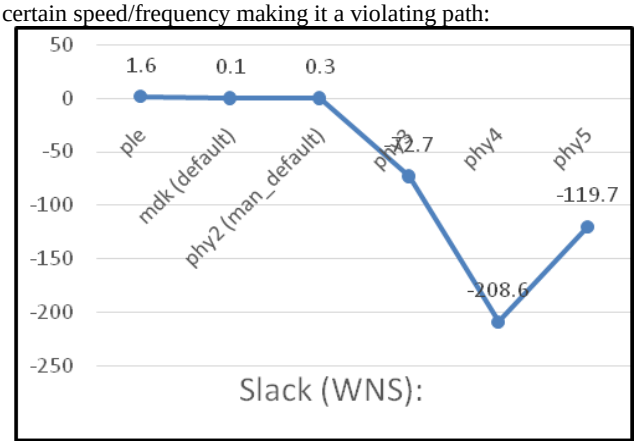

Fig. 5. Worst Negative Sack For all Paths (Synthesis)

Figure 5 showed the worst negative slack (WNS) time in nanoseconds in all the path of the design under test. Physical synthesis results is expected to be lower since it has considered physical attributes making it more pessimistic. Though negative slack is observed in three (3) test cases, the criteria that is needed for the synthesis simulation to pass the test is to have a positive slack in register-to-register path for the synthesis results.

Through designer's point of view, since the design is a block of a whole design, negative slack from other paths apart from the reg-to-reg path may be resolve in the integration of all blocks. Worst negative slack (WNS) for register to register path is shown in Figure 6. Figure 5 showed that all test cases have positive slack time in all test cases. There is also a decrease in slack, as expected, due to the less ideal estimation contributed by the physical attributes in our simulation input.

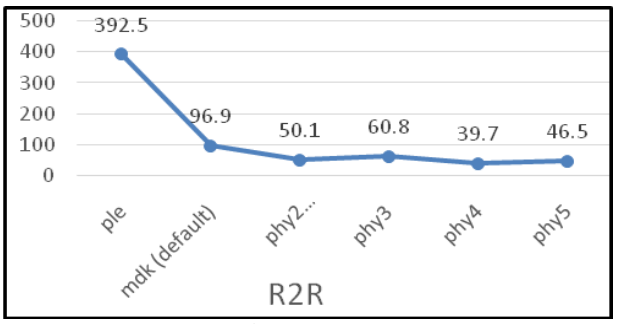

Fig. 6. Worst Negative Slack for Register-to-Register Paths (Synthesis)

5.2 Place And Route (PNR)

5.2.1 Timing Setup

Expectedly, more violations were introduced and the timing slack became more negative. This was evidently observed in Figure 7 and Figure 8 which showed the Timing setup results in terms of Worst Negative Slack, Total Negative Slack (TNS) and the number of violating paths of the design under test after the PNR signoff.

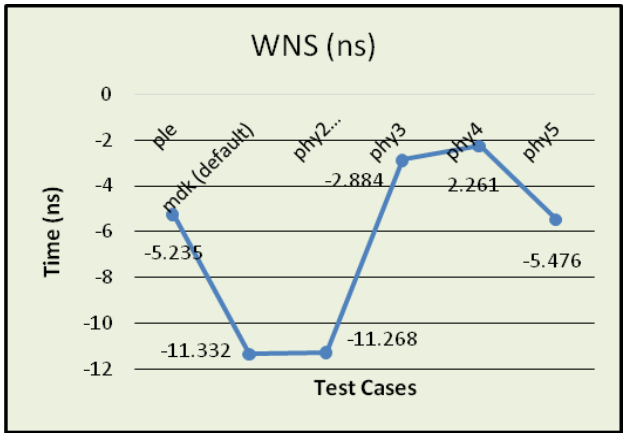

Fig. 7. Worst Negative Sack For all Paths (PNR Signoff)

For timing setup, Physical 4 still has the least number of violating paths of 144 followed by physical 3 with 205 violations. All other physical setups including the current methodology, PLE, has violating paths of more than 1000. Moreover, Physical 4 also has the least WNS slack of -2.261ns followed by Physical 3 with -2.884ns. Both were better than the PLE with WNS slack time of -5.235ns. All other physical test cases has worst timing setup results.

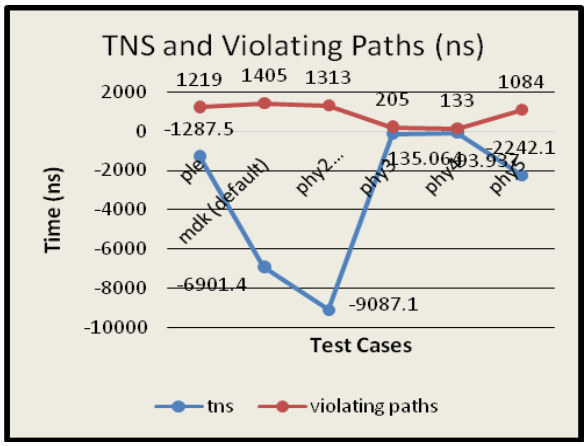

Fig. 8. TNS for All paths and the Number of Violating Paths(PNR signoff)

5.3 Runtime

One important aspect of considering the new Physical flow is its runtime. Adding physical attributes increases the runtime in synthesis as observed in Table 1. For the design under test, synthesis runtime increases by at least 73% in all physical setup. For the physical test cases that showed favourable results, physical 3 and 4 increased by 200%. Increased runtimes in synthesis alone is expected since physical attributes is now incorporated in the process. However, the increase in synthesis runtime should be negated by the decreased in the runtime of the Place and Route.
Recorded runtime for the place and route is also shown in Table 1. Significant reduction in runtime was observed in all physical setups except Physical 1. For the physical test cases that showed favourable results, Physical 3 and 4 has a decreased runtime by 33% and 48%, respectively.

Table 1. Runtime Comparison

RUNTIME	PLE	PHY1	PHY2	PHY3	PHY4	PHY5
Synthesis	19 min 10 sec	33 min 6 sec	32min 14sec	58min 16sec	48min 4sec	42min 56sec
PNR	312 min 44sec	370min 17 sec	268min 58sec	208min 2 sec	166min 16sec	270min 36 sec
Total:	332 min	404 min	301 min	266 min	214 min	312min

Moreover, data shown in table 4.3 suggest that the decrease of runtime in PNR significantly overwhelmed the increase in runtime in synthesis and even reduce the total runtime in all physical setup except for physical 1. Physical 3 and 4 has a decreased total runtime by 20% and 36%, respectively, compared to PLE.

6. Conclusion

In general, Physically-aware synthesis is a technique that utilizes the use of physical information such as physical libraries, placement of macros, blockages, and area information to the synthesis tool. This innovation, as expected, should provide better correlation with the physical data base thus providing more correlated timing results before and after place and route. However, such innovation was not that popular to designers because of its effect to the whole design flow was not studied further. With this study, optimal flow for synthesis run is achieved through several experimental setups. Effects in place and route (PNR), and Static Timing Analysis (STA) is also observed and carefully studied. With setup 3 and setup 4, physically aware synthesis is proven to have improved timing correlation between the synthesis and PNR results. Power after signoff also improved significantly. Total runtime from synthesis to timing closure reduces because of much lesser violations in the first iteration alone.

Acknowledgement

This research was funded by DOST-ERDT research grant and supported by Lattice Semiconductor Corporation. Special thanks to Place and Route (PNR) team managed and headed by Engr. Melvin Joey De Guzman for direct supervision of my research. Also, acknowledging the funding support of USAID-STRIDE and DOST-PCIEERD for the Microelectronics Laboratory IC Design tools.

References

[1] Balasubramanian J., "Making FPGA Synthesis Physically Aware", Retrieved from http://chipdesignmag.com/display.php?articleId=1002, November 2015.
[2] Alpert, C.J., Karandikar, S.K., Zhuo Li ; Gi-Joon Nam, et.al. (2007), Techniques for Fast Physical Synthesis. *Proceedings of the IEEE* 95, 573-599.
[3] Papa, D., Moffitt, M.D., Alpert, C.J., Markov, I.L. (2010), Speeding Up Physical Synthesis with Transactional Timing Analysis. *IEEE Design and Test of Computers* 27, 14-25.
[4] Parimi N. (2015), Leveraging Physically Aware Design-for-Test to Improve Area, Power, and Timing. *Cadence Designs Systems*.
[5] Weste, N., Harris, D., & Banerjee, A. (2005), CMOS VLSI design. *A circuits and systems perspective 11*, 739.
[6] Franzon, P.D., "ASIC Design Flow", Advance VLSI Design. Retrieved from www.ece.ncsu.edu/erl/faculty/paulf.html, 2012
[7] Genus Physical User Guide. Genus Synthesis Solution. Cadence Design Systems, Inc., Seely Avenue, San Jose, CA 95134, USA., 2015
[8] Zeidman, Bob., "Back to the basics: Programmable Systems on a Chip", Zeidman Technologies, Retrieved from http://www.design-reuse.com/articles/10991/back-to-the-basics-programmable-systems-on-a-chip.html , 2005
[9] Moxon T., "Exploring New Design Flows", Retrieved from http://www.eetimes.com/document.asp?doc_id=1216162, 2001
[10] Journal on Demand, "Physically Aware Synthesis Techniques to Lower Power, Improve Timing, Congestion & Correlation", EE Journal, Retrieved from http://www.eejournal.com/archives/on-demand/2014060304-cadence-synthesis/, 2016.
[11] Kudva Gopi, "Using Physically Aware Synthesis Techniques to Speed Design Closure of Advanced-Node SoCs" Retrieved from http://chipdesignmag.com/sld/blog/2015/03/23/using-physically-aware-synthesis-techniques-to-speed-design-closure-of-advanced-node-socs/, November 2015.
[12] Starter Guide SYN + DFT MDK. Lattice Semiconductors Corp., Oregon, United States, 2015.
[13] Tasuoka M, Watanabe R., Otsuka T., and Hasegawa T. (2015), Physically aware high level synthesis design flow. In *52nd Annual Design Automation Conference*, pp. 1-6.
[14] Lin Zhong, N.K. Jha, "Interconnect-aware High-level Synthesis for Low Power". In *IEEE/ACM International Conference on Computer Aided Design*, pp 110-117.
[15] Pandini, D., Pileggi, L., & Strojwas, A. (2002), Congestion-aware Logic Synthesis. In *Conference on Design, automation and test in Europe*, pp. 664.
[16] Clarke, M., Hammerschlag, D., Rardon, M., & Sood, A. (2011). Eliminating routing congestion issues with logic synthesis. *Whitepaper, Cadence Design Systems*., Retrieved from http://www.cadence.com/rl/resources/white_papers/routing_congestion_wp.pdf, December 2016
[17] Wu, J., Ma, C., & Huang, B. (2008). Congestion Aware High Level Synthesis Combined with Floorplanning. In *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, Vol. 2, pp. 935-938.