

# Designing Hierarchically Reusable Digital IPs Using DC-T/ICC Flow

James Deng  
Michael Lai  
Ninh Ngo  
Keith Duwel  
Kevin Huang  
Michael Zheng  
Richard Price

Altera Corporation  
San Jose, California, USA

[www.altera.com](http://www.altera.com)

Liang Xu, Sridhar Panchapakesan, Pallavi Padala  
Synopsys Inc.  
Mountain View, California, USA

## ABSTRACT

Designing hierarchically reusable IPs is challenging due to the multiple levels of hierarchy and MIM. The old flow used for the tape-out of digital IPs in Altera's 45-nm Stratix® IV GX FPGA has issues with correlation, handling multiple hierarchies, and timing closure in MCMM. This paper presents a non-standard DC-T/ICC hierarchical flow for designing digital ASIC IP blocks. *Several issues during the flow development and the approaches to address them are discussed with a focus on the capability of handling MIM and nested ILMs.* The results show that the new flow achieves a better QoR.

## Table of Contents

1	Introduction .....	3
2	Hierarchical-Based Digital Transceiver IP Design .....	4
2.1.	TRANSCEIVER DESIGN.....	4
2.2.	TRANSCEIVER SUB-IPs AND HIERARCHY .....	4
3	DC-T/ICC Implementation Flow .....	5
3.1.	CHALLENGES IN DC/ASTRO FLOW .....	5
3.2.	IMPLEMENTATION OF NON-STANDARD DC-T AND ICC-BASED FLOW .....	5
3.3.	INTERFACE LOGIC MODELS.....	6
3.3.1.	<i>Introduction to ILMs</i> .....	6
3.3.2.	<i>ILM-Based DC-T and ICC Flow Methodology</i> .....	7
4	Issues and Approaches .....	11
4.1.	DC-T SYNTHESIS OF HIGHER LEVEL BLOCKS .....	11
4.1.1	<i>DC-T Synthesis Middle Level Block with MIM</i> .....	11
4.1.2	<i>DC-T Synthesis Highest Level Block Using Nested ILMs</i> .....	12
4.2.	CLOCK TREE BALANCING ISSUE IN ICC .....	12
4.3.	OVER-UTILIZATION ISSUE IN ICC.....	13
4.4.	ISSUE OF USING ILMs IN ICC .....	14
4.5.	BAD PLACEMENT AND SCENIC ROUTE .....	14
4.6.	POWER REDUCTION.....	16
5	Conclusions .....	16
6	References .....	17
7	Appendix .....	17
7.1.	COMPARISON OF DC/DC-T RUN TIME AND MEMORY USAGE .....	17
7.2.	COMPARISON OF ASTRO/ICC RUN TIME, MEMORY USAGE, AND QoR .....	18

## 1 Introduction

This paper describes taking a hierarchical design through the Design Compiler Topographical mode/ IC Compiler (DC-T/ICC) flow. The unique aspect of the paper is that it describes three levels of hierarchy with nested interface logic models (ILMs) and multiple instantiated modules (MIMs). Instead of the standard DC-T/ICC hierarchical flow, the design uses nested ILMs in DC-T and ICC, something that very few users have done. The result shows that DC-T/ICC has better correlation throughout the flow and is able to handle important issues such as MIMs, ILMs, and multi-corner, multi-mode (MCMM) timing closure. Chapter 2 describes how the design used as the test vehicle for the flow comparison. It has MIMs and requires MCMM timing closure. Chapter 3 describes the overall DC-T/ICC flow used, its corresponding sub-flows—including the DC-T flow, the ICC flow, and the ILM flow—and how these flows improve QoR. Chapter 4 describes several issues we ran into during the flow development, and the approaches we applied to address those issues. Chapter 5 concludes with a summary of the paper.

The digital blocks inside FPGAs are designed with many levels of hierarchy to promote re-use and to support multiple-industry standard protocols. Each level of the hierarchy has design-specific requirements, which do not allow DC-T/ICC tools to perform hierarchical partitions freely. This makes the physical implementation quite difficult.

One challenge we had was with nested ILM models. At that time, DC-T (B-2008.09-SP4) did not support nested ILM models. The technique we used was to generate nested ILM models from ICC after the placement stage and use the models back to DC-T to re-run synthesis.

According to the ICC hierarchical design planning (HDP) document, the goal of (standard) HDP is to create a physical hierarchy to manage runtime and memory requirement issues, and to allow parallel processing of major physical sub-blocks. With such goals, no physical hierarchy may be required for small chips or blocks, while two levels of hierarchy are sufficient for the vast majority of customer designs needing large chips, a larger design can always be partitioned into sub-blocks in the second hierarchical level. The tool is thus designed to be flexible and capable of partitioning, power planning, feasibility checking, virtual flat, and pin cutting, etc.

In this project, our goal is different. Our goal is to reuse all sub-blocks at every hierarchical level. To flexibly reuse sub-blocks, our design has different partition requirements: no logic instances at the top level, the shapes of all sub-blocks are pre-determined, the locations of some sub-block pins are constrained or even pre-determined, and there are more than two levels of hierarchy. This design requires nested ILM models and channel-based interface protocols. For chip scalability, some logic is grouped into a basic block, so it can be instantiated multiple times as needed. Hence, MIMs must be part of the design implementation. All the aforementioned unique design requirements make the design synthesis and P&R flow non-standard.

## 2 Hierarchical-Based Digital Transceiver IP Design

This chapter highlights the key features of the design and its levels of hierarchy used in our evaluation.

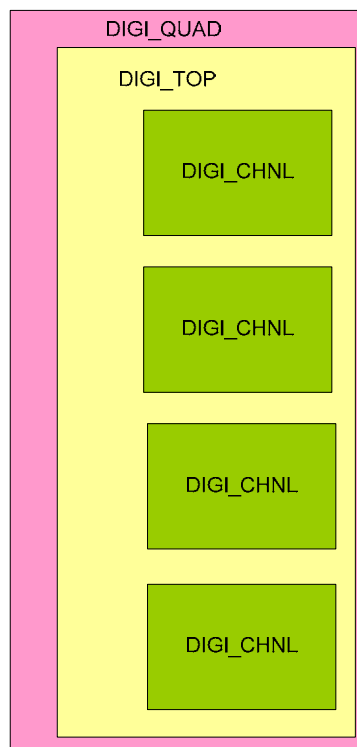
### 2.1. Transceiver Design

The transceiver design introduced in this paper is an IP in Altera's 45-nm Stratix® IV GX family. The IP can operate at speeds of up to 500 MHz. The transceivers perform a multitude of industry standard protocols as well as proprietary protocols. The IP performs the physical coding sublayer (PCS) operation in serial-based protocols.

### 2.2. Transceiver Sub-IPs and Hierarchy

Every hierarchy level contains reusable IPs, hence the necessity to preserve each module. The challenge lies in preserving the information necessary for the back-end flow methodology to connect all levels of the hierarchy. Figure 1 shows the hierarchy of the Digi\_quad design, which follows this methodology.

**Figure 1 - Digi\_quad Hierarchy**



The Stratix IV GX FPGA's PCS and corresponding upper levels are used as test vehicles to compare the results. The design has three hierarchical levels:

- Digi\_quad—multiple-instantiated in the whole transceiver
- Digi\_top—targeted for reuse in other IPs
- Digi\_chnl—multiple-instantiated inside Digi\_top

### **3 DC-T/ICC Implementation Flow**

This section describes the difference between the previous and the new flows, as well as the ILM concept and flow.

#### **3.1. Challenges in old DC/Astro Flow**

Our digital IP was originally taped out as a built-in part of our advanced FPGA. Bottom-up methodology was used. Lower level blocks were synthesized using designer estimated I/O delay constraints. Locations of I/O ports of the lower level blocks were optimized by the tool without knowing how the timing and routing affects upper-level blocks. The three major challenges were correlation between synthesis and P&R, handling multiple hierarchies, and timing closure in MCMM.

Due to the lack of correlation between steps, clocks and I/Os were over-constrained throughout the design flow to meet signoff requirements. Over-constraint was largest at the beginning of design flow (synthesis) and was reduced as the design implementation moved further along in the flow. Thus, there is a lack of correlation between synthesis and P&R.

The digital IP has three levels of hierarchy. The previous flow had difficulty with multiple levels of hierarchy by using hierarchical timing view (HTV) and extracted timing model (ETM) generated from PrimeTime. Neither of these two models can be nested. A workaround using mixed models was invented but it was very complicated and inaccurate.

Our IP blocks, as a built-in part of a FPGA, must support many (>100) modes across multiple-speed bins. Timing closure involved incremental timing fixes across multiple modes and corners. Since previous P&R took only a single set of constraints, workarounds by emulating reduced constraints in a single mode were required. Since this was only possible on a coarse scale, many paths were over-constrained. It ended up that P&R timing (single set of constraints) and PrimeTime timing (multiple sets of constraints) did not correlate well. Over-constraint caused wasted effort on fixing non-real issues. Serious hold violations were found in PrimeTime that required lengthy, inefficient fixes late in the implementation stage. Timing closure was thus very painful due to lack of Multi-Corner, Multi-Mode (MCMM) analysis.

#### **3.2. Implementation of Non-Standard DC-T and ICC-Based Flow**

The goal of the comparison was to verify whether major Stratix IV GX FPGA issues—such as correlation throughout design flow, hierarchical design issues, and MCMM design requirements—could be resolved or addressed by using the DC-T/ICC flow.

For comparison, we took a key sub-block in our digital IP block as the design, and implemented it using the new DC-T and ICC-based flow. The design has three levels of hierarchy as described earlier. The DC-T tool version we used is B-2008.09-SP4, and the ICC tool version is B-2008.09-SP4.

### 3.3. Interface Logic Models

Because the use of nested ILMs plays an important role and was a challenge for our design, we will first discuss the concept of ILMs and nested ILMs.

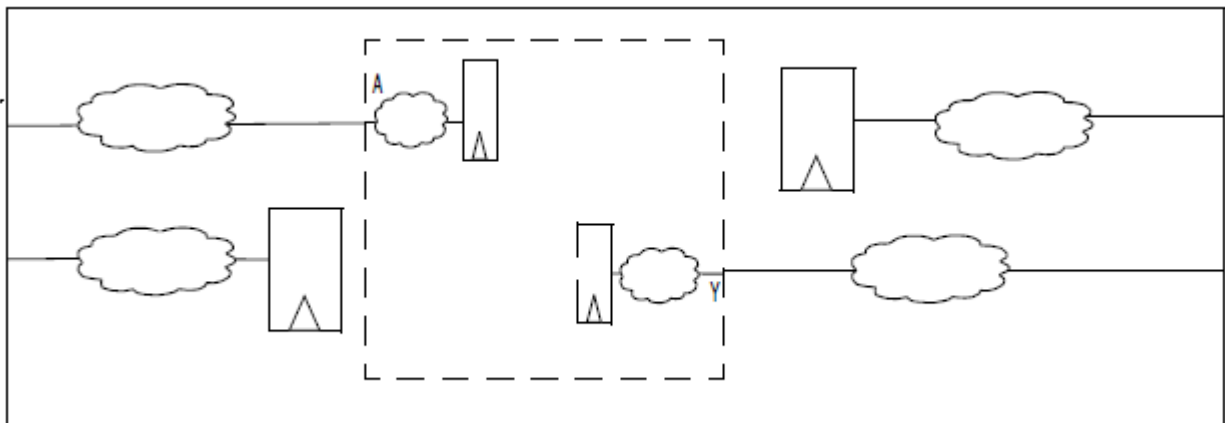
#### 3.3.1. Introduction to ILMs

ILMs are used in Design Compiler to reduce the number of design objects and memory required when performing top-level optimization on large designs. In an ILM, the gate-level netlist for a block is modeled by another gate-level netlist that contains the interface logic of a block and possibly logic associated with the interface logic. Combinational paths from the input ports that do not encounter a sequential element and pass directly to an output port are also retained in the ILM [1]. Logic not associated with the interface is removed [2]. ILMs enhance capacity and reduce runtime for top-level optimization [3].

The intent of an ILM is to produce a model that closely resembles the timing of the interface to that of the block-level netlist. The ILM generator takes as an input the gate-level netlist for a block and generates a flattened Verilog netlist that contains only the interface logic, without the internal register-to-register logic.[1]

A nested ILM is an ILM that is contained in another ILM. When you run the “create\_ilm” command on a block that contains nested ILMs, IC Compiler retains only the interface logic for the current block. Logic from a nested (lower level) ILM is retained only if it is involved in the interface paths for the current block [3] as shown in the following diagram. This minimizes the size of the upper level ILM and also minimizes the size for the required overall memory footprint.

Figure 2 – Nested ILM



The ILM partial netlist is instantiated in place of the block-level netlist within the chip (top level) netlist. The methodology outlined in this chapter provides the steps needed to generate, validate, and use ILMs in DC-T/ICC flow.

### 3.3.2. *ILM-Based DC-T and ICC Flow Methodology*

In our flow, the synthesized netlist of the design was used for initial floor planning and placement in ICC. The physical placement information was then written out in design exchange format (DEF) and fed back to DC-T for re-synthesis. With such placement-aware re-synthesis, timing correlation with the post-place design is expected.

ICC supports ILM, and ILM can be used for multiple levels of hierarchy. Thus, we used a single timing model, ILM, to address all of our hierarchical design needs from synthesis through P&R. Nested ILMs and multiple instantiated ILMs were also fed back to DC-T during re-synthesis to meet our design requirements.

To test the capability of MCMM support in the DC-T/ICC flow, we selected five scenarios, consisting of selected constraint SDC files and timing PVT corners:

- “func”—function mode SDC constraint file with slow-slow and fast-fast corner timing models
- “func1”—function mode SDC constraint file with slow-slow and fast-fast corner timing models, but at a different temperature from “func”
- “scan”—scan mode SDC file with slow-slow and fast-fast corner timing models
- “leak”—function mode SDC constraint file with the typical-typical corner timing model only
- “CTS-only” —mode defined for clock tree synthesis (CTS) stage only

The “func”, “func1”, and “scan” scenarios were used for timing closure with respect to two sets of constraints concurrently. The “leak” scenario was used for leakage power optimization concurrently with timing closure, a capability that was not available in the previous flow. In all scenarios, there was mixed-Vt cell usage (consisting of a mix of SVT cells and HVT cells in non-critical timing paths).

Figure 3 shows the DC-T hierarchical flow used for the top-level synthesis. Besides taking as an input the physical floorplan information DEF file generated from ICC, ILM generated from ICC was also used as input. The reason we used ICC generated ILM instead of DC-T generated ILM was that DC-T did not have the support for nested ILM at the time of the flow development. Starting with B-2009.06, DC-T supports nested ILM generation.

Figure 3 – DC-T Hierarchical Flow

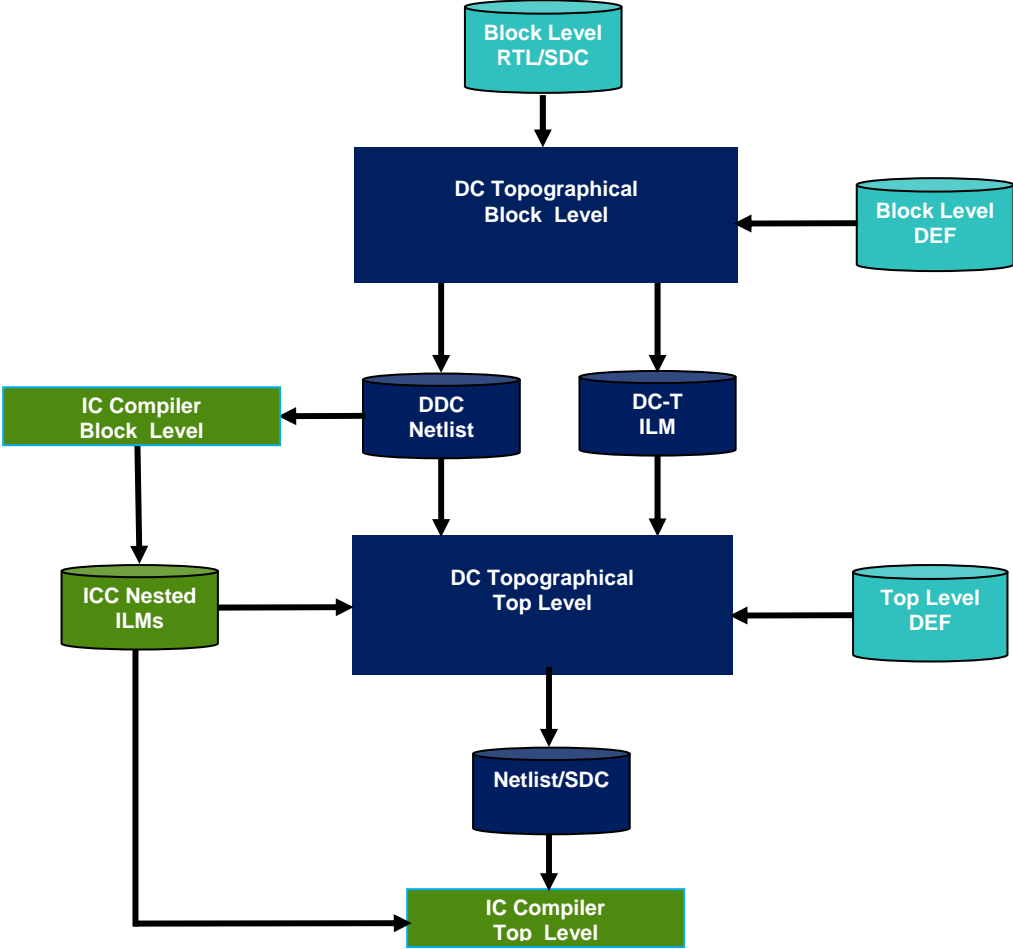


Figure 4 shows the ICC flow used. The flow is based on the standard Synopsys reference methodology (RM), with ILM inputs as additions.

Figure 4 – ICC RM Flow

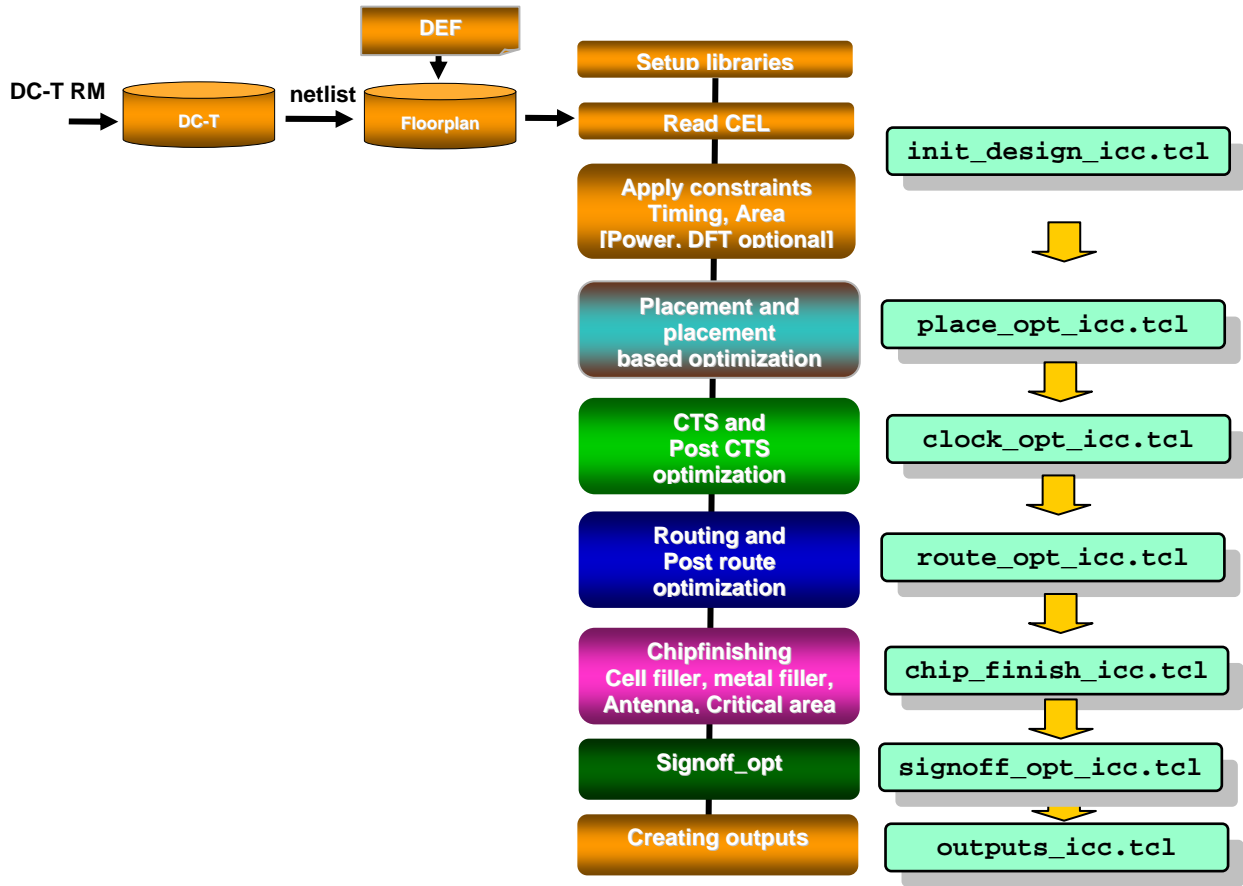
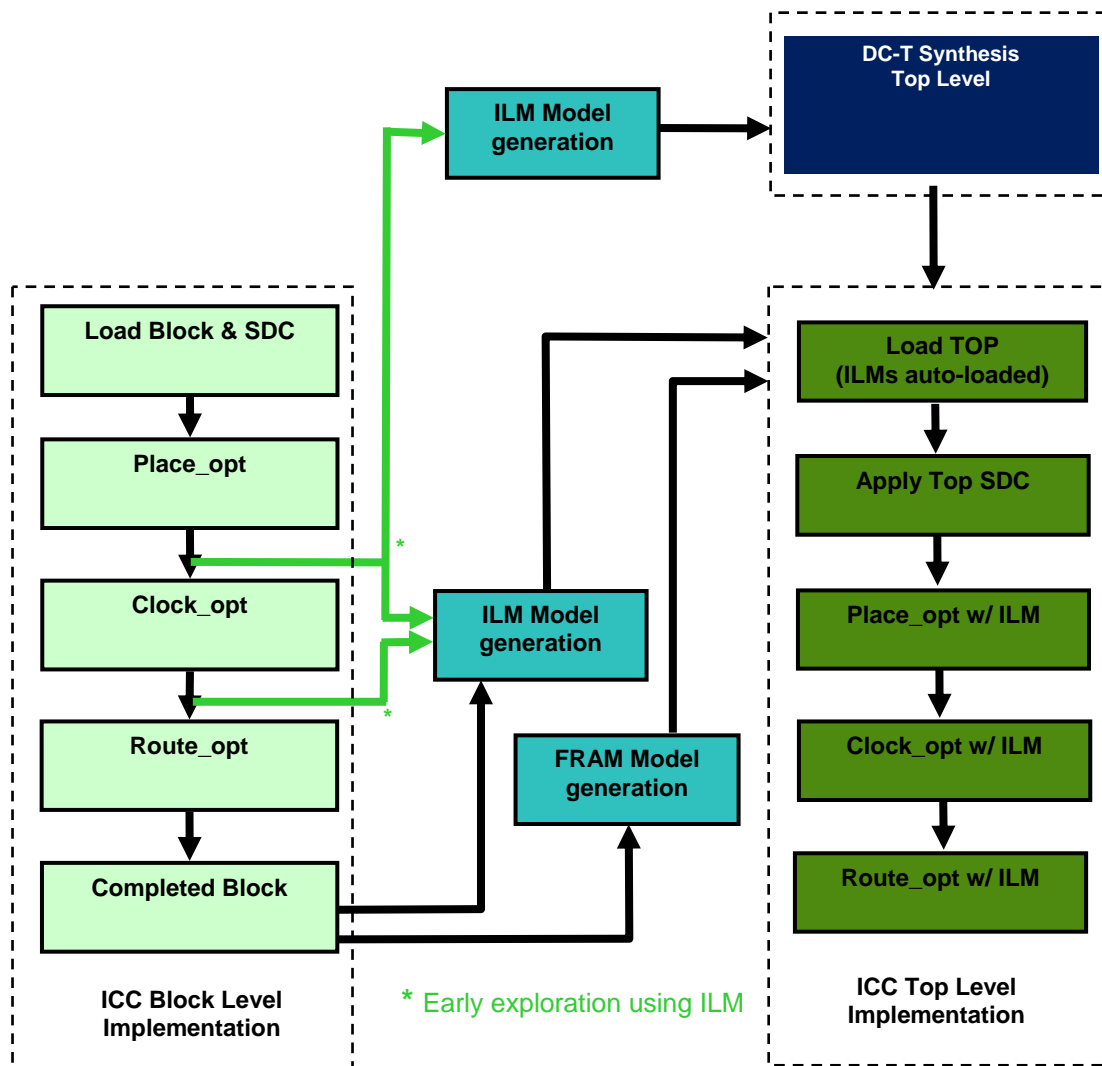


Figure 5 shows the ILM flow. The sub-block ILMs were generated from ICC after place\_opt was used in DC-T for re-synthesis. Two levels of ILMs were generated from ICC and both fed back to DC-T for top-level synthesis. Note that, for top-level design (digi\_quad) synthesis, ILM1 (digi\_top) is sufficient, as it contains all four instances of nested ILM2 (digi\_chnl). However, if only ILM1 is fed back into DC\_T for synthesis, the timing analysis can only reach to the second hierarchical level (digi\_top). When both ILM1 and ILM2 were read into DC-T, timing analysis could see into the third hierarchical level (digi\_chnl), which is required by our design to analyze the interface delay of digi\_chnl, because digi\_chnl is also a reusable IP block. In addition, the same block-level ILM was used by ICC for early exploration of top-level P&R.

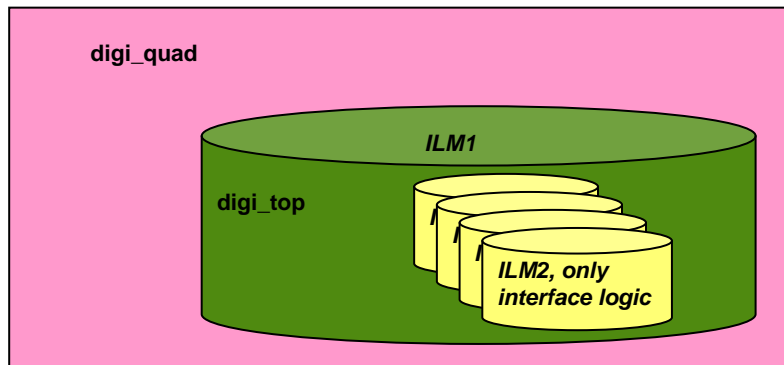
Figure 5 – ILM Flow



In the implementation of the third hierarchical level block digi\_chnl, the ILM (ILM2 as illustrated in Figure 6) is generated after the placement stage. In the implementation of the second hierarchical level block digi\_top, ILM2 is instantiated four times to replace the four original di-

gi\_chnl blocks. After implementation, another ILM (ILM1 as illustrated in Figure 6) is again generated to represent block digi\_top. Thus, ILM2 is nested inside ILM1. During top level block (digi\_quad) DC-T synthesis and ICC implementation, only ILM1 is required.

Figure 6 – Nested ILMs in the Design Hierarchy



During our implementation, we found that the ILM generated by DC-T (version B-2008.09-SP4) could not handle nested ILMs, contrary to what the manufacturer claims for the tool. An enhancement request was sent to Synopsys. This implied that we are one of the earliest users to try to use this method of nested ILMs. The work-around we tried was designed to generate the ILM from ICC, then let DC-T use ICC-generated ILMs.

## 4 Issues and Approaches

This section describes several issues we ran into during the flow development, and the approaches we applied to address those issues.

### 4.1. DC-T Synthesis of Higher Level Blocks

There were two issues with DC-T synthesis of higher level blocks. The first issue was handling MIM at the middle level block. The second issue was in the the highest level block taking nested ILMs of the middle and lowest level.

#### 4.1.1 DC-T Synthesis Middle Level Block with MIM

There are two options to pass lowest level block (digi\_chnl) DC-T results to middle level (digi\_top) DC-T: using DDC format or using ILM model.

However, at digi\_top level, there was a MIM issue by using DC-T to generate a DDC file. There are four digi\_chnl instances at digi\_top level. When loading the digi\_chnl DDC file for digi\_top, DC-T always uniquifies those four digi\_chnl instances. This is because when the sub-block DDC is used, the tool gives the user the flexibility to use it as a physical block or just as a logical block. In order to provide this flexibility when multiple instances of the sub-block are used, the tool uniquifies them. We could not use this approach as we wanted to re-use the P&R work of the sub-block.

We then tried the ILM format and the MIM problem was avoided. The ILM of `digi_chnl` block was generated from both DC-T and ICC. At the `digi_top` level, we were able to use both DC-T and ICC generated ILMs to resolve the unification issue.

#### 4.1.2 DC-T Synthesis Highest Level Block Using Nested ILMs

Our design has complicated clock structures where many clock muxes are used in the bottom level block `digi_chnl`. In `digi_quad` level synthesis, nested ILMs from the lower two levels need to be used to enable checking of timing paths through the middle level of hierarchy down to the bottom level. However, there was a problem with DC-T generated ILMs. A nested ILM generated in DC-T did not have appropriate placement attributes of leaf ILM objects, which resulted in wrong timing information of `digi_chnl` to the `digi_quad`. The issue was reported to Synopsys R&D, and it was fixed later in DC-T version 2009.06.

We then moved to ICC to generate `digi_chnl` ILM and `digi_top` ILM for DC-T to synthesize `digi_quad`. A nested ILM generated in ICC works for DC-T. The methodology we used for the `digi_quad`:

- Read in the verilog netlist of `digi_quad`
- Use ICC ILMs for `digi_top` and `digi_chnl`
  - Specify the ILMs as part of the Milkyway Reference libraries
  - Add ICC ILMs to the `link_library`

## 4.2. Clock Tree Balancing Issue in ICC

Our design has generated clocks which have multiple sources. Our original clock definition for a three-sources generated clock is like this:

```
create_generated_clock [get_pins d1/rclk_by2_reg/Q] -name rclk_by2 \  
    -source [get_pins d1/rclk_by2_reg/CP] -divide_by 2
```

Such clock definition is straightforward and has no problem in DC-T. ICC has no problem in completing CTS either. However, ICC does not balance the clock tree across all sources and bad skew is seen. DC-T has no such issue. DC-T treats clocks nets as ideal nets and does not need to do clock tree synthesis. ICC CTS automatically picks one clock source for clock tree synthesis. Our design requirement was that all three source clocks and generated clocks should be balanced with each other.

To address this issue, instead of creating one generated clock on the pin, we created three generated clocks, each with its own clock source.

```
create_generated_clock [get_pins d1/rclk_by2_reg/Q] -name rclk_by2_1 -source txpclk \  
    -divide_by 2 -add  
create_generated_clock [get_pins d1/rclk_by2_reg/Q] -name rclk_by2_2 -source rcvclk \  
    -divide_by 2 -add  
create_generated_clock [get_pins d1/rclk_by2_reg/Q] -name rclk_by2_3 -source refclk \  
    -divide_by 2 -add  
set_clock_groups -physically_exclusive -group {rclk_by2_1} -group {rclk_by2_2} \  
    -group {rclk_by2_3}
```

Since ICC automatically balances a generated clock to its source, and since ICC can be specified to balance all sources using the following command:

```
set_inter_clock_delay_options -balance_group "txpclk rcvclk refclk"
```

all generated clocks on the pin and their sources are then balanced.

### 4.3. Over-utilization Issue in ICC

The bottom level block `digi_chnl` has a high utilization. Since DC-T does not fix hold, it does not provide accurate information about congestion and routability. In ICC, when we tried to fix hold with aggressive margins, the block became un-routable. The major technique we used to address the over-utilization issue was adding in two different sets of delay cells for hold fixing. First, we use only small delay cells to fix hold timing violations in function mode only, then we use both large and small delay cells to fix hold in scan mode.

In our 45nm library, there are over a dozen delay buffers, with nominal delay ranging from 25ps to 500ps. While larger delay cells are more area efficient, they are long-channel delay cells and tend to be less accurate in timing due to large manufacturing variation, so they are not used in function paths to reduce manufacturing risk.

The two sets of buffers are shown below:

```
set ICC_FIX_HOLD_PREFER_CELLS " \
t45hvt/DEL025 \
t45hvt/DEL050 \
t45hvt/DEL075 \
t45/DEL025 \
t45/DEL050 \
t45/DEL075 \
t45hvt/BUFFD2"
set ICC_FIX_HOLD_PREFER_CELLS_SCAN " \
t45hvt/DEL025 \
t45hvt/DEL050 \
t45hvt/DEL075 \
t45hvt/DEL100 t45hvt/DEL125 t45hvt/DEL150 t45hvt/DEL175 \
t45hvt/DEL200 t45hvt/DEL220 t45hvt/DEL250 \
t45/DEL025 \
... \
t45hvt/BUFFD2"
```

In ICC, first, we set "func" scenario active and set "scan" scenario inactive, then

```
icc_shell> set_prefer -min $ICC_FIX_HOLD_PREFER_CELLS
```

and fix function mode holds. Second, we also activate "scan" mode and

```
icc_shell> set_prefer -min $ICC_FIX_HOLD_PREFER_CELLS_SCAN
```

to fix scan mode hold violations. Note in the "func" scenario, scan paths must be disabled. This was done through the `set_case_analysis` command in the two SDC files.

Our experiments showed that over 5% area reduction was achieved by using the above approach, and the block became routable without physical DRC violations.

#### 4.4. Issue of Using ILMs in ICC

After digi\_chnl block was implemented and timing-clean, its ILM was generated:

```
icc_shell> create_ilm -include_xtalk -keep_full_clock_tree -keep_boundary_cells \  
-no_auto_ignore -compact none -include_side_load all
```

The ILM was passed to next level digi\_top for implementation. However, at the digi\_top level, ICC reported erroneous timing violations inside digi\_chnl. This issue was reported to Synopsys and it was fixed.

At digi\_quad, CTS had one issue. For hierarchy reuse, in hierarchical planning, we let some input clocks to digi\_quad go down directly to digi\_top, so no CTS is needed for those clocks at digi\_quad level. The right side boundary (where clocks come in) of digi\_top overlaps with that of digi\_quad. When using the Synopsys reference methodology to run CTS, CTS guide buffer was not able to be removed after CTS, and was placed far away due to placement blockage. We set option to not use guide buffer to resolve this issue.

```
icc_shell> set cts_remove_ilm_guide_buffer true
```

The methodology we used for digi\_quad :

- Read in the verilog netlist of digi\_quad
- Use ICC ILMs for digi\_top and digi\_chnl
  - Specify the ILMs as part of the Milkyway Reference libraries

#### 4.5. Bad Placement and Scenic Route

Another uncommon feature in our design is its thin but very tall shape. The digi\_top has an aspect ratio of more than 1:10. Both DC-T and ICC are challenged by such a shape.

When tracing timing violation paths, a few scenic route paths were found. It was due to bad placement of cells. Cells were placed up, down, and up again. See figure 7 below.

DC-T showed no timing violations, but post P&R showed large timing violations. This placement quality issue was reported to Synopsys R&D, but there is no solution for enhancement. It could be the limitation of the tool when doing cell placement on shapes with extreme aspect ratios.

Figure 7 – Cell Placement and Scenic Route



The approach we used to address this issue was to create placement bounds on groups of cells in ICC place\_opt\_icc.tcl. The sample command is:

```
create_bounds -name bd7 -effort medium {d1/../../pip[7]*}
```

With the creation of such placement bounds, the cells which used to be placed too far-away were now confined in a region, and scenic placement issue was mitigated.

In our case, the above placement problem was seen on both DC-T and ICC. Just the placement issue in DC-T was not bad enough to cause timing violation in DC-T. Since DC-T also supports the `and create_bounds`, it may also be used in DC-T to help DC-T timing.

#### **4.6. Power Reduction**

Our leakage power signoff corner is different from timing signoff corners. We used typical process, typical voltage but high temperature for signoff, which differs from setup timing corners where we use lower voltage and two different temperatures (to cover the temperature-inversion effect), and also differs from hold timing corner where we use higher voltage and low temperature. With the MCMM capability in ICC, we are able perform leakage power optimization and measure leakage power only in leakage-only scenario.

Our experiment showed that the power curve was non-monotonic, and performing ICC power optimization in timing corners did not result in minimum power when measured in leakage corner.

We did several experiments: using only ICC to do power optimization, using both DC-T (an additional Power Compiler license is required) and ICC to do power optimization. For ICC power optimization, we used the best-effort. For DC-T power optimization, we tried combinations of five major options in DC-T power compiler to synthesize different netlists. While doing ICC power optimization on the not-power-optimized DC-T netlist showed larger percentage power improvement, the minimum post P&R power was achieved by using DC-T power-optimized netlist.

## **5 Conclusions**

DC-T/ICC flow addresses the major issues we faced during the Stratix IV GX FPGA's digital IP implementation. Using the new flow, we are able to achieve better QoR and tighter correlation between DC-T and the back end. The flow is able to handle multiple levels of hierarchy by using nested ILMs with a workaround of generating nested ILMs from ICC and using them for DC-T. The design also instantiates four instances of the lower-level ILM, which proves the flow works with MIM. Regarding MCMM, the flow is able to optimize both timing (different modes and different corners) and leakage power (leakage-only corner) concurrently with a marginal increase in runtime and thus reducing the number of iterations. The methodology also achieved better skew and insertion delay in CTS. When using the ICC Z-Route technology, the routing time is significantly reduced and the results are DRC clean.

The nested ILM model approach can significantly reduce iterations for multiple-level hierarchical design implementation. Altera is using this approach for our next-generation product implementation, which has even more levels of hierarchy. Future work also includes exploration of I/O constraint budgeting with a top-down flow.

## 6 References

- [1] PrimeTime Modeling User Guide (Version D-2009.12, December 2009):
- [2] Design Compiler User Guide (Version C-2009.06, June 2009):
- [3] IC Compiler Implementation User Guide (Version C-2009.06-SP4, December 2009):
- [4] IC Compiler 2: HDP Workshop:

## 7 Appendix

The following tables summarize the results of our implementation. In the implementation, the original floorplan DEF files were used for all three level blocks to maintain the same area, ports placement, power, blockage etc. physical information. We tried to maintain the same SDC files as much as possible, except some necessary modifications as described in Chapter 4.

### 7.1. Comparison of DC/DC-T Run Time and Memory Usage

Table 1 compares the run-time and memory usage of DC vs. DC-T.

Table 1 – DC-T vs. DC

DESIGNS	Run Time (Minutes)		Memory Usage (MB)	
	DC	DC-T	DC	DC-T
<b>digi_chnl</b>	<b>82</b>	<b>37</b>	<b>737</b>	<b>921</b>
<b>digi_top</b>	<b>50</b>	<b>50</b>	<b>1719</b>	<b>2055</b>
<b>digi_quad</b>	<b>10</b>	<b>20</b>	<b>1397</b>	<b>1630</b>

Notable results are:

- DC and DC-T runtimes are comparable
- DC-T consumes ~ 20% more memory due to placement aware synthesis

- Tighter correlation between DC-T to backend

## 7.2. Comparison of Astro/ICC Run Time, Memory Usage, and QoR

Table 2 compares the run time, memory usage, utilization, and number of iterations between Astro (Version 2007.03-SP6) and ICC.

Table 2 – ICC vs. Astro (1/2)

Tools	DESIGNS	Run Time (Hours)		Memory Usage (Mb)		Utilization (%)		# of Iterations*	
		Single	MCMM	Single	MCMM	Single	MCMM	Single	MCMM
Astro (DC Netlist)	digi_chnl	24.9	n/a	2000	n/a	81	n/a	more	n/a
	digi_top	16.5	n/a	2073	n/a	27	n/a	more	n/a
ICC (DC Netlist)	digi_chnl**	4.92	5.86	847	913	70**	78**	less	less
ICC (DC-T Netlist)	digi_chnl**	3.86	4.71	785	878	79**	86**	less	less
	digi_top	15.16	n/a	2163	n/a	27	n/a	less	n/a

\* # of iterations is the total of (1) the number of synthesis netlist handed-off to PnR and (2) the number PnR runs per given netlist

\*\* ICC/DC and ICC/DC-T netlists use different constraints (100ps setup relaxed for ICC/DC)

Notable results include:

- ICC is faster than Astro in single scenario runs
- ICC MCMM optimizes across different modes/corners concurrently with marginal increase in runtime
- Number of iterations is smaller for DC-T/ICC flow than DC/Astro flow
- The utilization of the ICC/DC netlist is less than the ICC/DC-T netlist because the constraints in each experiment were slightly different (ICC/DC had 100ps relaxed setup constraint)

Table 3 compares the worst negative slack (WNS) setup and hold, and clock network delay and skew between Astro and ICC.

Table 3 – ICC vs. Astro (2/2)

Tools	DESIGN	WNS Setup Time (ps)		WNS Hold Time (ps)		CTS Longest Insertion Delay (ps) / Skew (ps)					
		Single	MCMM	Single	MCMM	txpclk	refclk	rcvclk	rcvclk0	rxclk	txclk
		Astro*** (DC Netlist)	digi_chnl	0	n/a	0	n/a	759 / 106	767 / 103	698 / 95	650 / 38
digi_top	0		n/a	0	n/a	815 / 601	1113 / 667	994 / 520	994 / 520	691 / 141	549 / 77
						814 / 601		724 / 516		488 / 60	431 / 41
						814 / 601		723 / 516		502 / 60	445 / 41
						813 / 601	723 / 516	484 / 60	435 / 41		
ICC (DC Netlist)	digi_chnl	0	0	-20	-30	551 / 49**	568 / 48**	506 / 40	448 / 29	238 / 7	261 / 10
ICC (DC-T Netlist)	digi_chnl	0	0	-26	-40	555 / 47**	589 / 54**	514 / 48	401 / 29	259 / 5	257 / 13
	digi_top (ch0,1,2,3)	-310	n/a	-40	n/a	597 / 421	1095 / 476	795 / 414	795 / 414	330 / 33	393 / 22
						592 / 421		560 / 399		307 / 13	319 / 8
						596 / 425		565 / 403		307 / 13	320 / 8
						577 / 409	546 / 387	305 / 13	320 / 8		

\*\* ICC/DC and ICC/DC-T netlists use different constraints (100ps setup relaxed for ICC/DC)

\*\*\* Astro result is sign-off quality. No ECO timing fixes done on the ICC netlists.

Notable results include:

- DC-T/ICC was able to achieve good QoR (setup WNS clean, hold – violations) in a much more automated fashion. There are some timing violations left at digi\_top level as we did not do any ECO to close the timing.
- CTS: DC-T/ICC achieved better skew and insertion delay
- Routing: Significant reduction in routing time with clean DRC using ICC zroute
- Digi\_top insertion delay improved by the new flow
- Digi\_top skew amongst the three clock sources txpclk, refclk, and rcvclk is improved partially by the new flow and partially due to the constraint modification as described in Section 4.2



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

Copyright © 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.