

Formal verification of ‘full chip’ containing ‘shell’ partitions with and without feed-thrus

Rashid Iqbal

Intel Shannon Ireland

emali:rashid.iqbal@intel.com

Abstract— the ‘Full Chip’ module of an ASIC is made up of various partitions and similar to individual partitions, it also goes through various stages of the physical design. The initial design-planning works on the existing pins of a partition, adds feed-thru pins and performs custom placement and routing on signal and clock nets. Near tape-in, ECO (Engineering Change Order) forces manual changes to design as opposed to taking it through full implementation cycle. At the final stages, when layout database of each partition meets the requirements, a bottom up integration is carried out to create full chip layout. All these stages can create logical differences between layout and RTL of the top level interface. In order to verify that no unintentional logical change has happened to full chip, a robust formal verification strategy with numerous practical considerations is necessary. Designers also make use of 100% or partial shell models at the floor-planning phase which does bring advantages but also creates challenges for the formal verification flow. This article documents these challenges by explaining the formal verification approach taken on Intel’s next generation network processing chip.

Keywords – ASIC, physical design, equivalence check, full chip, feed-thrus

I INTRODUCTION

Formal equivalence verification (FEV) is a mandatory part of every ASIC physical design flow. The top level design or full chip (FC) module is divided into various partitions. Due to the large size of the full chip, the equivalence checking is also done at two stages. The first stage performs FEV on individual partitions and second stage verifies the FC interface. The FEV of a partition is carried out at least for the three stages: 1) after synthesis 2) after scan insertion or stitching 3) after fully routed design. The purpose of full chip FEV is to make sure that the partition interface at full chip has not gone through any unintentional change by the physical design. The number and type of stages, a full chip physical design goes through, depends upon project specific methodology.

The physical design (PD) cycle of the full chip starts at the floorplanning or exploration stage which determines the dimension and location of each partition. It also determines the physical location of each pin of a partition. The early models of a partition in floorplanning phase consist of a mixture of shell and detail netlists. For abutted designs, feed-thrus are created which change ports of a feed-thru

partition and depending upon the methodology the additional feed-through wrapper/logic is also added to the partition. Late ECOs can occur on the designs which cause manual logical changes to the design instead of taking it through the full synthesis to routing cycle. Besides logical changes, a custom routing or buffer addition can also occur at full chip PD. After all partitions have gone through their PD cycle and meet the requirements, final integration is carried to form full chip and do various full chip validations.

In summary, a full chip module goes through various processing stages in PD and therefore it is very important to perform regular FEV checks between the physical databases (revised model) and the RTL (golden model). Various practical considerations have to be taken into account to perform a robust full chip FEV at these stages. Due to the specific methods or constraints used by the industry FEV tools [1], a thorough review of all full chip nets under various scenarios have to be carried out.

This paper is organized into three main sections. Section II explains different stages of the physical design that can change a full chip interface. A detail explanation of the shell models and feed-

thru methodology is provided with examples from the real design. Section III goes into the details of mandatory FEV scenarios and important considerations for the setup of each scenario. The summary of the work is provided under ‘conclusion’ section.

II STAGES WHERE FEV IS REQUIRED

On physical design side, a full chip design goes through following major stages:

- 1) Early floorplan explorations
- 2) Final netlist & layout integration
- 3) Late ECOs on full chip interface

The floorplan iterations at the early stage of the project are performed by ‘reading in’ netlist of each partition into the floorplan tool. These netlists are ‘verilog’ format files that are created by performing synthesis on each partition. Ideally we need to ‘read in’ complete netlist, but due to the large size of some designs, a ‘reduced netlist’ or ‘shell’ model is sufficient. Depending upon whether full internal detail of a partition is required or not, the logical model of a partition can be swapped between the detail netlist and the shell. The ‘shell model’ of a partition only contains the interface definition and no sub-block instances [2].

In an abutted full chip floorplan, adjacent partitions do not contain any channels or logic between their boundaries. In this scenario a partition talking to another partition not adjacent to itself has to pass its signals/wires through another partition (feed-thru partition) before it reaches its destination partition. Physical design engineers have to create additional ports in feed-thru partition and this is where RTL full chip and physical full chip become different. Typically there are large number of feed-thru ports so whether this editing process is done manually (less likely) or using a ‘script’, a mistake is likely to happen. This editing can also affect non feed-thru pins of the partition.

The shell models of the feed-thru partition are further modified to contain new feed-thru ports and feed-through wrappers. Figure 1 shows the full chip floorplan with various partitions. An example of feed-thru partition is GP. As shown in Figure 2, the shell model of GE and PE is completely empty (100% shell) while feed-thru partition has feed-thru wrapper inside its shell. This ‘feed-thru wrapper’ contains some ‘hard coded’ buffers to connect feed-thru inputs to the feed-thru outputs. Inside shell models of these partitions, none feed-thru ports are still not connected to any logic.

After multiple floorplan iterations, each partition owner is provided with the partition dimensions, location of ports, feed-thru wrappers (if any) and feed-thru ports (if any). FEV of this database has to be done before delivering this data to

the partition owner. This ensures that FC floorplan owner has not created any unintentional logical change to the design.

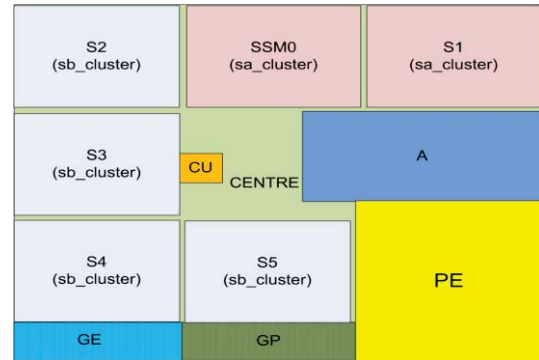


Figure 1 Full chip floorplan

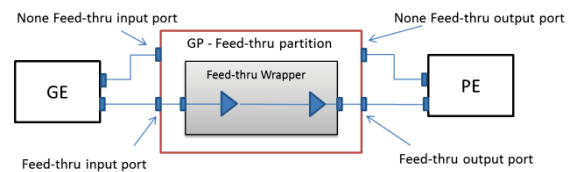


Figure 2 shell model of a feed-thru partition

After taking this floorplan data, each partition owner carries out multiple synthesis and PnR (Place and Route) iterations before achieving a ‘converged’ design that meets all physical design requirements. Each partition owner provides final routed netlist and layout to the full chip owner for timing, noise, reliability and layout verifications. Prior to giving this data to the FC owner, each partition owner also carries out block level FEV to verify that netlist or layout is logically equivalent to the partition RTL. After building FC design with these netlists, an FEV of the FC interface has to be done. The purpose of this full chip FEV is to make sure that full chip design with netlists, matches with the full chip RTL. This will also verify that partition PnR work has not changed any logic on feed-thru ports and feed-thru wrappers.

Late ECOs happen near Tape-In (TI) when a logical connection has to change and instead of taking new RTL through synthesis and other physical design flow stages, a manual fix is done on the routed databases. These ECOs can also change the full chip interface, requiring changes to the FEV methodology. To give an example of this: the power domain change for GP partition at very late stage required the addition of isolation gates at all ports including feed-thrus. The initial feed-thru wrappers which were given to the partition owner didn’t have isolation gates. The partition owner had already manually changed the design inside PnR tools to add isolation cells at those ports. He or she had not gone through the full cycle of changing those wrappers first and then synthesizing and performing PnR. The

FC integration owner tries to run FEV between the netlist (which has isolation cells) and the original wrappers which didn't contain isolation cells. This required additional checks and scenarios in the FEV methodology.

III FEV SCENARIOS & IMPORTANT CONSIDERATIONS

To perform FEV at different physical design stages that were discussed in previous section, one has to pick scenarios from Figure 3.

	GOLDEN		REVISED	
	Feed-thru	None Feed-thru	Feed-thru	None Feed-thru
1	100% SHELL RTL	100% SHELL RTL	partial SHELL RTL	100% SHELL RTL
2	partial SHELL RTL	100% SHELL RTL	partial SHELL NETLIST	100% SHELL NETLIST
3	100% SHELL RTL	100% SHELL RTL	partial SHELL NETLIST	100% SHELL NETLIST
4	partial SHELL RTL	100% SHELL RTL	partial SHELL RTL ECO	100% SHELL RTL ECO
5	partial SHELL RTL ECO	100% SHELL RTL ECO	partial SHELL NETLIST ECO	100% SHELL NETLIST ECO
6	100% SHELL RTL	100% SHELL RTL	partial SHELL RTL ECO	100% SHELL RTL ECO

Figure 3 FEV scenarios

For example at early stage, when no routed netlists and no ECOs are done, scenario '1' is required. Once routed netlists are available from partitions then additional scenarios such as '2' and '3' are required. To the end of a project, if an ECO is performed then additional scenarios of '4' and '5' are required.

Some of these scenarios will not show completely clean reports. For example scenario '4' and '6' will show some mismatches. These mismatches need to be looked at one by one (if the total number is small) or should be compared using a script or should be modeled with correct behavior in golden or revised.

Next we take a look at the general considerations for all the scenarios in Figure 3. The use of 'shell' models requires us to look at the following aspects:

- 1) A partition cannot be verified until it has same state (black box or not block box) in both golden and revised models.
- 2) A shell model without any input to output connection (a 100% shell) is considered a 'black box' module by the tool [1].
- 3) The shell model of a feed-thru partition contains feed-thru wrappers which have feed-thru inputs connected to the feed-thru outputs. By default, these partitions are not considered 'black box' unless these are explicitly defined this way in the tool.
- 4) An FEV setup where a feed-thru partition is 100% shell model in golden and partial

shell (for example scenario 1 in Figure 3) in revised, a dummy input-to-output connection has to be created in the golden. The purpose of this edit is to stop the tool from treating the golden as 'black box'.

- 5) The dummy input-to-output connections do not create interface issue for FEV as two sub-checks are performed within each scenario.

Due to the use of 'shell' models and the creation of dummy input-to-output connection, a single FEV run cannot ensure 100% coverage. We need to perform two separate runs for each scenario:

- 1) Checking feed-thru ports of the partition
- 2) Check none feed-thru ports of the partition

As shown in Figure 4, the two categories depend on whether feed-thru partitions are treated as 'black box' or not.

Inside Each Scenario

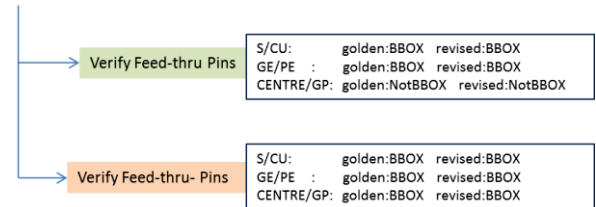


Figure 4 Two separate checks required within each scenario

Checking feed-thru pins only

An FEV setup where all none feed-thru partitions are 'black box', can only ensure a full verification of the interface whose source and destination partitions are also 'black box'. The feed-thru pins obviously fall into this category. However the interface whose source or sink pins is part of a feed-thru partition do not fall into this category.

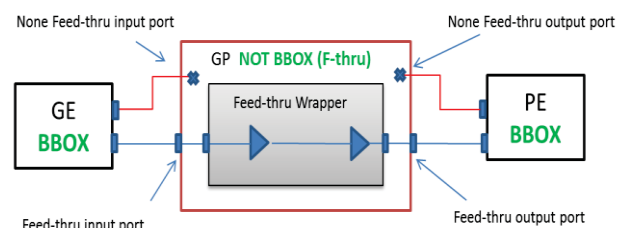


Figure 5 FEV sub-scenario to check feed-thru pins

From FEV point of view, all partition pins can be categorized into two categories: 1) valid key points 2) floating or disconnected pins. A key point is a sequential pin, primary input, primary output and 'black box' input [1][3]. A floating pin (such as none feed-thru pin of a partition) is given a default

value of logic '0' or '1'. A key point is validated by the tool for both logic '0' and logic '1' by traversing its fan-in or fan-out logic cones until primary input or primary output is reached.

All pins of a 'black box' partition are treated as key points. As mentioned earlier, the feed-thru cluster is not treated as 'black box' by the tool, because there is a connection from feed-thru input to feed-thru output with buffers between them. This translates none feed-thru pins of the feed-thru cluster into disconnected or floating pins. This can result into an issue if there is a disconnection caused on these pins by mistake. In this case the FEV tool will not catch that problem.

In order to illustrate this with an example, consider a net that connects GE to CENTRE in golden (RTL) model. The two partitions are adjacent to each other so no feed-thru is involved. Since CENTRE is a feed-thru partition, to the FEV tool this connection appears as disconnected at CENTRE boundary. Assume a mistake is made during the feed-thru addition which left this net disconnected from CENTRE. Now the interface is disconnected in both golden (due to the FEV/tool setting) and revised (due to the mistake). The FEV tool will not report any problem on this due to the two cases to be logically identical.

Checking all non-feed-thru pins

During this check, all partitions (even the feed-thru ones) are made 'black box'. This makes all pins of the feed-thru partitions as key point, which is good for none feed-thru pins.

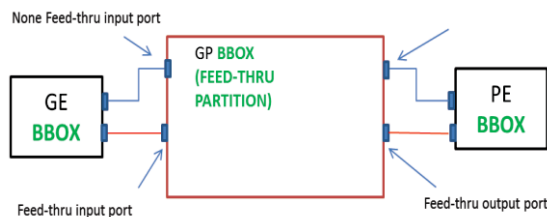


Figure 6 FEV sub-scenario to check none feed-thru pins

However this creates a problem for feed-thru pins, as tool does not see connection through feed-thru wrappers. This causes all feed-thru pins reported as 'not equivalent'. In order to overcome this false reporting, we used tool commands to model all feed-thru wrapper behaviour. If feed-thru pins are not modelled in this way there is a possibility that an actual none feed-thru pin is hidden in the list. By using modelling commands we received fully clean reports, thus there was no need to maintain separate waivers.

CONCLUSION

There are various FEV scenarios to be performed for different stages of the physical design of full chip interface. These physical design stages are: early floorplan exploration, full chip integration and ECO near tape-in. There were six different scenarios that were performed on a real taped-out design with 15 million instances. Earlier on, the floor planning was carried out using 'shell' models with major benefit of reduced design size. The shell models of the feed-thru partitions had 'feed-thru' wrappers created in physical domain to connect their feed-thru inputs to the feed-thru outputs. Due to these shell models and feed-thru wrappers, two sub-scenarios were performed within each six main FEV scenarios (thus a total of twelve scenarios). The feed-thru partitions are not treated as black box by the tool. For none feed-thru ports, black boxing all partitions caused lot of false errors on feed-thru ports which. A proper modeling of feed-thrus was performed to make sure that an actual error is not bypassed. An ECO near tape-in caused additional FEV scenarios.

REFERENCES

- [1] Conformal www.cadence.com
- [2] Rashid Iqbal, "Hierarchical Design-Planning of a Multi-million Instance Design" SNUG France 2012.
- [3] "A Formal Verification Methodology for a Fully Abutted Hierarchical Design" International Cadence user group, Santa Clara 2004.