

ECO flow using Physical Compiler

Niv Margalit

niv@cisco.com

Abstract:

An ASIC's logic is developed in HDL (Hardware Description language) to allow higher level of abstraction other than simple gate representation. The HDL is synthesized and a netlist representation of the very same logic is achieved. The process of synthesis is time consuming.

ECO (Engineering Change Order) is the process of inserting a logic change directly into the netlist. This approach is beneficial because thus, there is no need for full ASIC synthesis (saving time) and a reduced price is charged by the ASIC vendor.

Traditionally it is regarded as very hard task to perform ECO that involves more than few hundreds of gate. Hence, when the logic change requires a change of many gates, we may be drawn back to the time and money consuming effort of changing the HDL then synthesizing the full ASIC again.

Developing better and safer ECO process can be very rewarding. There are few approaches for actually doing an ECO, among which:

- Manual fixing
- Regular synthesis tools with different placement tools (for example Synopsys Design Compiler with IBM PDS)
- Formal equivalence tools to do an ECO
- Synopsys Physical Compiler (PC)

In CSI, (Cisco Systems Israel) we have developed and ECO flow based on PC.

In this presentation will show how beneficial this flow via PC can be. We will shortly define few ECO methods. We will introduce, in more details the PC flow and demonstrated its use on the Sharq ASIC (HFR) on which 100,000 new logic gates (!) were added.

1.0 Introduction

An ASIC's logic is developed in HDL (Hardware Description language) to allow higher level of abstraction other than simple gate representation. The HDL is synthesized and a netlist representation of the very same logic is achieved. The process of synthesis is time consuming.

ECO (Engineering Change Order) is the process of inserting a logic change directly into the netlist. This approach is beneficial because thus, there is no need for full ASIC synthesis (saving time) and a reduced price is charged by the ASIC vendor.

Traditionally it is regarded as very hard task to perform ECO that involves more than few hundreds of gates. Hence, when the logic change requires a change of many gates, we may be drawn back to the time and money consuming effort of changing the HDL then synthesizing the full ASIC again.

Developing better and safer ECO process can be very rewarding.

2.0 What is an ECO?

There are 2 main scenarios for doing an ECO.

- 1) Doing a spin to an existing design for fixing a bug or adding new features.
- 2) Adding or changing a netlist at a very late stage of the project.

In both cases we want to change a netlist with minimum effort and cost.

An ECO is a local change and therefore there is no need to touch the full netlist.

In many cases doing an ECO does not require new clocks, this reduces the time and effort required if we compare it to a full netlist spin.

Some of the ECOs can be done as a metal fix only, this means that new gates will be implemented with gate array cells (spare cells) and only the wiring mask will be updated. This reduces the cost of the spin. This paper will not go into details of preparing a metal fix but in general the way to achieve this is by limiting the synthesis tool to using only Gate Array cells.

When doing an ECO there is no need to rewire the full design, it is possible to only run an incremental wiring job to do the local area of the ECO.

In many cases it will be much faster to update the netlist with an ECO flow then with a full synthesis of the design.

3.0 Traditional ECO flows

The traditional ECO flow is build out of 5 stages.

- 1) RTL coding of the ECO –

The needed change in the design as it would be written in a high level language (for example Verilog). Limitations on how the new code needs to be written depends on the way that the ECO will be later synthesized later on.

2) Synthesize the new ECO RTL code –

The new code is synthesized and the first feed back loop is created to the writing stage, the loop between the synthesis stage and the writing stage will be closed only after all of the synthesis constrains are met. The output of this stage is an ECO netlist.

3) Integration of the ECO netlist into the design –

After synthesizing the new code we now have 2 netlists, which have to be merged into a single netlist. This stage is usually done by a script that connects all of the ECO's inputs to the proper drivers and all of the ECO's outputs to the cells that they have to drive. The cells that are driven by the ECO's are cell from the old design so before they are connected to the new logic they first have to be disconnected from the old logic. The old logic which is now floating after it was replaced by the new logic can be either terminated or recursively cleaned by a script.

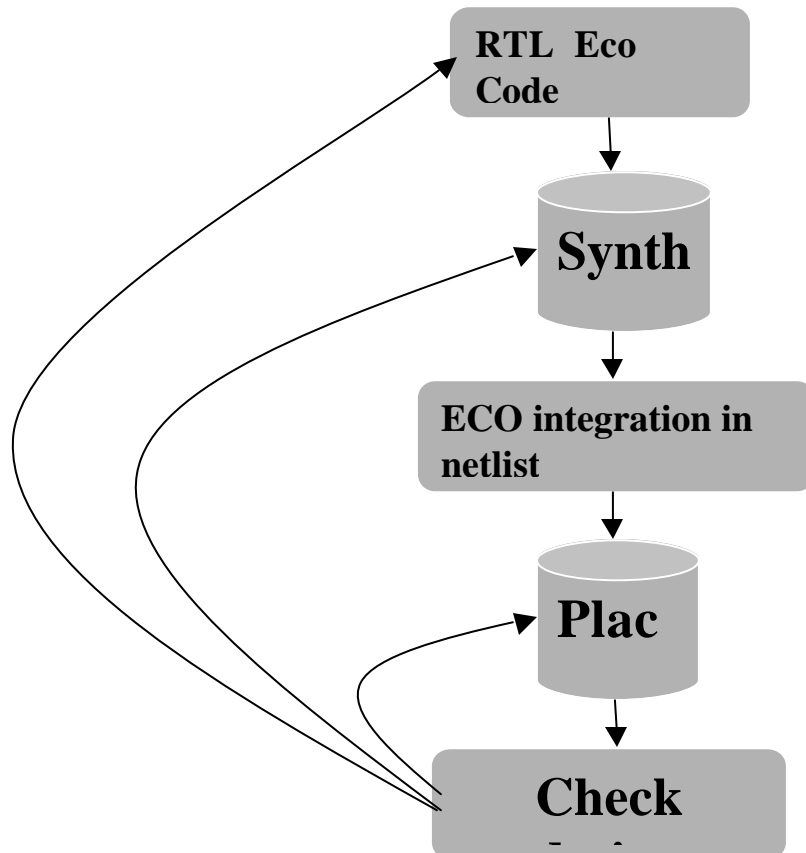
4) Placement of the ECO modified design –

Now that we have the ECO netlist integrated into the design, the new ECO cells need to be placed in legal placements between the existing cells of the old design.

5) Design Verification –

The last stage is to check that the design is fully legalized and that there are no timing violations, Because of the connection stage and the placement stage these checks usually fail on the first few iterations and there is a need to go back to one of the previous stages and fix the failures

Figure 1: Traditional ECO flow



4.0 ECO flow using Physical Compiler

The ECO flow that uses Physical Compiler is boiled out of 3 stages

1) RTL coding of the ECO –

The needed change in the design as it would be written in a high level language (for example Verilog). Limitations on how the new code needs to be written depends on the way that the ECO will be later synthesized later on. (This stage is the same as it was in the traditional ECO flow).

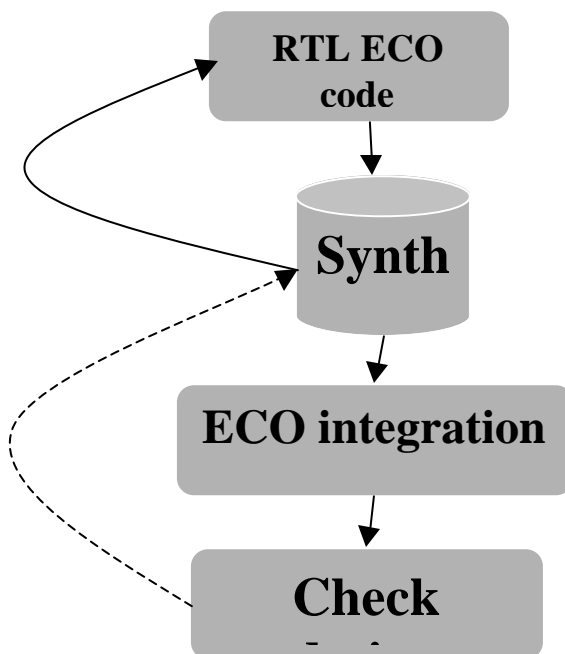
2) Synthesize the new ECO code using Physical Compiler –

This is not a regular Physical Compiler run, before starting the run we show the tool all of the existing cell of the old design that are in the area of the ECO, all the rest of the chip area is blocked. Under these physical constrains it is now guaranteed that all of the new cells will be legally placed without overlapping any of the existing cells. The results of this stage give fast feedback to the design and placement constrains and this stage should keep doing iterations until the design is timing clean.

3) Connect and check the design.

These 2 stages are now together because the single synthesis run provides us a netlist, which is timing clean and is fully legalized. The connect script is the same as the one used in the traditional flow but all of the check should now be clean.

Figure 2: ECO flow using PC



5.0 Writing constrains for PC

1) Blockages

In order to create the needed blockages the area of the ECO has to be defined, this is done by identifying the location of all of the cells that will be driving the ECO's inputs and the location of all of the cells that are driven by the ECO's outputs. After marking the area of the ECO the script will automatically create blockages to block all the area outside of the ECO area. Inside the ECO area the script will collect all of the old design cells information, cell type, cell location, cell size, mirror, and flip the combination of all this information will create a blockage for each of the old cells inside the ECO area.

2) Ports

The ECO's inputs and outputs must be preplaced before the run begins, each input port will be placed next to its driving cell and each output port will be placed next to the cell that it needs to drive.

All of the ECO's ports have to be isolated, to make sure that the ECO's logic will not over load the driving cells. Also it is recommended to define a weak driving strength to the ECO's inputs and this way the isolation buffer will be placed very close to the driving cell. It is also recommended to give a good load to the ECO's outputs in order to guarantee that the ECO's outputs will be strong enough to drive the old cell that they will be connected to at the end of the ECO.

3) Timing

All of the ECO's paths must be timing limited according to the clock cycle if it is a flop to flop path or according to time budget for the new logic in cases of, logic to flop, flop to logic or logic to logic paths.

It is recommended to write the ECO in a way that the paths are flop to flop, this way the timing constraint section of the PC run is very simple and can be defined by a single line that limits the max delay from all inputs to all outputs to the clock cycle minus needed margins.

6.0 Conclusions and Recommendations

The ramp up for using Physical Compiler is long but once it is done then it can be relatively easy to do the needed changes in order to use it for doing ECOs. The fact that it is possible to show the tool all of the existing logic and then in close timing and find a legal place to all of the new logic makes it very easy to close any size of an ECO with in a very short schedule.

The tool is currently limited to 300k blockages for a single run so in cases of very large ECO's, which are spread across a large area, there is a need to split the ECO into a few smaller runs.