Clock Path ECO with PrimeTime DMSA fix_eco_timing

Anne Yue
Rajeev Srivastava

Synapse Design
San Jose CA, USA

www.synapse-da.com

ABSTRACT

PrimeTime fix_eco_timing feature has been improved drastically over the last two years. A lot of published paper testified for how good are the results from PT_ECO. But not much paper discuss on how to fix the left over violations after the PT_ECO. PT_ECO can fix ~85% setup violations by "data path cell sizing". But PT_ECO cannot touch the clock path cells. This paper addresses the issues and solutions for "working with PT for the clock path ECO". Three solutions can be used for different timing closure scenarios and design requests. This paper also describes the wish list for Synopsys new features with the clock path ECO.

Table of Contents

Table of Figures

Table of Examples

1. Introduction

PrimeTime has been the golden timing signoff tool for ASIC signoff. No matter what implementation tool has been used in different projects, PrimeTime can always work seamlessly in different design flow.

With ASIC design size increasing exponentially every year, the SOC timing signoff & timing ECO phase has increased their weight in terms of overall project time. We as ASIC engineers found ourselves working on 20 million instance designs last year, and this year we are dealing with 65 million instance designs. With 32nm & less technology, the temperature inversion and power related analysis bring in more PVT corners for timing STA to cover.

With such a big SOC project, one PrimeTime_ECO run normally takes 3~5 days. The turn-around-time for timing ECO and timing signoff has become a bottle neck in the later stage of the project.

PrimeTime with DMSA

PrimeTime did a very good job utilizing computer power to help designers speed up the timing signoff. Especially, the introduction of DMSA makes it possible to handle the multi-corner, multi-mode in parallel runs.

Compared to other tools in the whole design flow, Primetime is the one which can see the complete pictures of all the timing violations. The DMSA concept makes it possible for EDA tools to see and fix the design issues with different mode/corner in one run, and it makes it possible for Primetime to take a more important role on the ECO stage.

PrimeTime in ECO

In the past 2~3 years, the PrimeTime DMSA ECO feature was improved drastically for every PT release. The *fix_eco_timing* and *fix_eco_drc* have changed the landscape of SOC timing signoff task assignment territory.

PrimeTime has started to take on functions more traditionally associated with ASIC implementation. For the design signoff stage, PrimeTime is changing its function from “observation only” to “part of design implementation”.

Currently PrimeTime ECO is good at doing the MCMM cell upsizing & buffer inserting. The tool pretty much covers all the options for upsizing & inserting. But what PrimeTime ECO cannot do is the clock tree ECO. PrimeTime ECO cannot touch the cells along the launch/capture clock path.

By understanding what PrimeTime can/cannot do, we can work with the post_PT_ECO netlist, and know how to fix the stubborn violation paths PT_ECO cannot fix.

There are certain proven timing ECO strategies which can fix the setup/hold violation, but the algorithms take a lot of manual work and create back-end iterations, which again increase the project time.

All those post_PT_ECO manual ECO algorithms in this paper are proven in silicon, and could be good candidates for Synopsys tool PT_ECO automation improvement. This paper suggests several PT_ECO features for the clock tree ECO. If those features can be integrated into the PrimeTime tool, it will help the ASIC designer achieve timing signoff faster, and have better Turn-around-time.

2. Challenges for Clock path ECO with PrimeTime DMSA

What PrimeTime can do in ECO

Backend design teams hand off the back annotated ready netlist to the STA team. We make certain assumptions before we decided to start a timing ECO phase. We are not doing the QA for the netlist, but we do want make sure the netlist has the following aspects before we feed it into the PrimeTime tool:
. A good CTS structure, with workable clock skew.
. Backend tool did go through optimization and timing closure. There are no big transition or fanout violations.
. Setup/hold WNS and TNS are workable. At least the timing report within back-end tool is reasonable.
. Parasitics extracted and back annotated without errors.
. Signoff corner list & mode list are defined and agreed across the whole project team.
. Constraints are verified and approved.

With well-defined DMSA Scenarios, PrimeTime tool will start the timing ECO based on the violation files from different Scenarios. PrimeTime will work on the violation path if it is showing up in the DMSA reports. If for constraint reasons, this path cannot be reported by PrimeTime, then there is no chance PrimeTime can work on it.

Good constraint files are important for a fast, convergent signoff process. We are not intending to cover the SDC constraint topic in this paper. But SDC constraints are important to the design, just like RTL quality is important to the design.

In PrimeTime, there are two types of fix_eco command available: *fix_eco_timing* & *fix_eco_drc*

fix_eco_timing has 2 options:
- setup
- hold.

fix_eco_drc has 3 options:
- max_tran
- max_cap
- max_fanout

fix_eco_drc helps the netlist get better transition/fanout performance, and actually prepares the netlist for *fix_eco_timing* run.

The DMSA ECO log files show us very complicated iteration and process for a fix_eco run. The algorithm is complicated, but exactly what tool can do is very simple. If we use the following violation path as example, what *fix_eco_timing* and *fix_eco_drc* can do is:

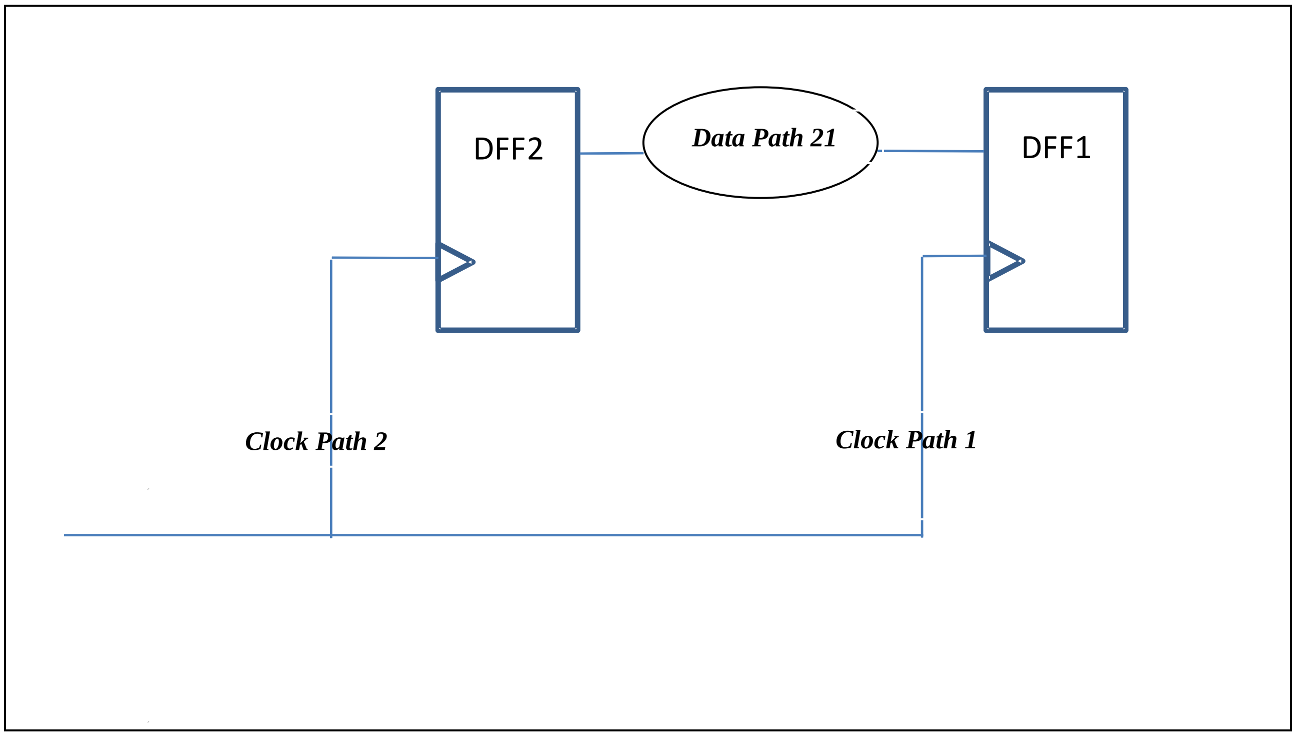

Figure 1 – DFF with Clock Path & Data Path

. Sizing the cells in Data Path 21(DFF2->DFF1)
. Add buffers in Data Path 21.

There is nothing wrong with these two approaches, because sizing and buffering is a very efficient way to resolve setup/hold violations. In general, after one *fix_eco_timing* run, 85~90% of setup violations, and 99% of hold violations are gone. This proves that PrimeTime *fix_eco_timing* did choose a very good approach.

But, by studying Figure 1, we will ask the question :
Does everyone forget the Clock Path 1 & Clock Path 2 ?

What PrimeTime cannot do in ECO

Both *fix_eco_timing* and *fix_eco_drc* cannot do:

. Change the Clock Path 1 & Clock Path 2.
. Remove cells. PrimeTime ECO cannot remove any cell along Clock Path 1, Clock Path 2 & Data Path 12.

For good reasons, PrimeTime tool doesn’t want to touch the CTS cells, which is understandable. A stabilized clock tree is necessary for a reliable and convergent signoff process. The huge CTS clock tree for the whole chip is very fragile. Once the loading and routing resource lose balance, the setup/hold timing reports will start change the topology, just like a broken spider web.

WLM in timing report

PrimeTime Pre_ECO timing analysis uses Wire Load Model(WLM) for timing reports once disconnect/connect is used in ECO tcl.

That means the Pre_ECO timing reports are not reliable anymore. This kind of PrimeTime behaviour is definitely not good news when we want to do the clock path re-stitching. It creates difficulty for clock path ECO, and requires at least one more iterations for us to confirm that the new re-stitched clock path can handle the latency we planned.

Multiple iterations for accurate target insertion delay

In certain ECO circumstances, we want to create a small scope clock tree structure aside of main stream CTS clock tree. Our approch is to utilize backend tools as a jump start to create a basic frame-work. We can setup the target latency as a input to BE tools, but pretty much, the first round of annotation result will not match our latency goal.

In general, it needs two rounds of ECO to achive accurate target clock tree insertion delay.

3. Solutions for Clock path ECO with PrimeTime DMSA ECO

As we mentioned before, 85~90% of setup violations can be fixed by current PrimeTime *fix_eco_timing*. With DMSA MCMM timing reports, the PrimeTime tool will analyze all the corner, mode, setup/hold and SI information. It can do better job than human on data path cell sizing and data path buffer inserting.

So the reality is, after PrimeTime *fix_eco_timing* run, we are running out of options on the Data Path fix. We believe all the good options on Data Path fix have been present inside the ECO.tcl from PrimeTime DMSA ECO.

To fix the left over 15~10% setup violations, we have no choice; we need to touch the Clock Path.

Solution I : Upsize the DFF

PrimeTime ECO cannot change the Clock Path 1 & Clock Path 2 in Figure 1. In general, Clock Path components include the clock buffer/inverter cells. The bad news is, beside the general clock tree cells, PrimeTime considers the clock path endpoint cell (DFF) as part of the clock tree system, hence the PrimeTime ECO will not touch the DFF cells.

Example 1 is a sample Post_PrimeTime_ECO path report:

```
Startpoint: spicnt_b/raddr_reg_0_
     (rising edge-triggered flip-flop clocked by gclk_bypass_neg)
Endpoint: rf_b/pe_counter2/pe_cnt_cl_g/gclk_en_reg
    (falling edge-triggered flip-flop clocked by gclk_bypass_neg)
Path Group: gclk_bypass_neg
Path Type: max
Scenario: func_ss

Point                                     Fanout       Incr         Path
 ---------------------------------------------------------------------
-------------------
clock gclk_bypass_neg (rise edge)                    0.0000       0.0000
clock network delay (propagated)                     3.7267       3.7267
spicnt_b/raddr_reg_0_/CK (DFFRX2M)                   0.0000       3.7267 r
spicnt_b/raddr_reg_0_/QN (DFFRX2M)                   0.4649 &     4.1916 r
spicnt_b/n_77 (net)                        1
spicnt_b/FE_RC_637_0/AN (NOR2BX8M)                   0.0001 &     4.1917 r
```

```
spicnt_b/FE_RC_637_0/Y (NOR2BX8M)              0.1533 &   4.3450 r
spicnt_b/FE_RN_283_0 (net)          1
spicnt_b/FE_RC_652_0/A (NAND4X8M)              0.0000 &   4.3451 r
spicnt_b/FE_RC_652_0/Y (NAND4X8M)              0.1127 &   4.4578 f
.....
.....
.....
```

Example 1 – Post PrimeTime_ECO path report

As shown in this path report, all the Data path cells are proparely upsized to *8M, except the DFF cell. The DFF cell is with the size *2M, which is the default size from synthesis tool.

This is a low hanging fruit we don't want to miss for a still-violating path. Upsizing the DFFs will benefit the setup fix in the following ways:

- The DFF cell delay # will improve.
- New upsized DFF improves the transition delay for downstream cells, which give possibility for another round of PT_ECO run to optimize the data path cells.

So what we plan to do next is:

- upsizing DFF
- additional PT_ECO run

The ECOed netlist show that we gain back ~100 ps for this path.

- ***Point of interest for Synopsys tool development team***: This should be an easy implementation on top of current DMSA ECO. By releasing the "not touch" restriction for all the DFFs, the ECO iteration could cover the sizing for the necessary DFFs.
- ***Good news to PrimeTime User***: The 2013.06 version of PrimeTime will include an enhancement to *fix_eco_timing* to allow register sizing.

Solution II : Clock tree re-stitching

If *Solution I* cannot solve the problem and there are still violations, then we need to touch the clock tree. Additional timing reports need to be generated to study the feasibility of clock tree re-stitching approach. The check is to find out if the previous stage of DFF paths has extra setup margin.

PrimeTime setup/hold violation path report is an end-point base report format. So, we choose to go with the end point → start point fashion for the clock path ECO.

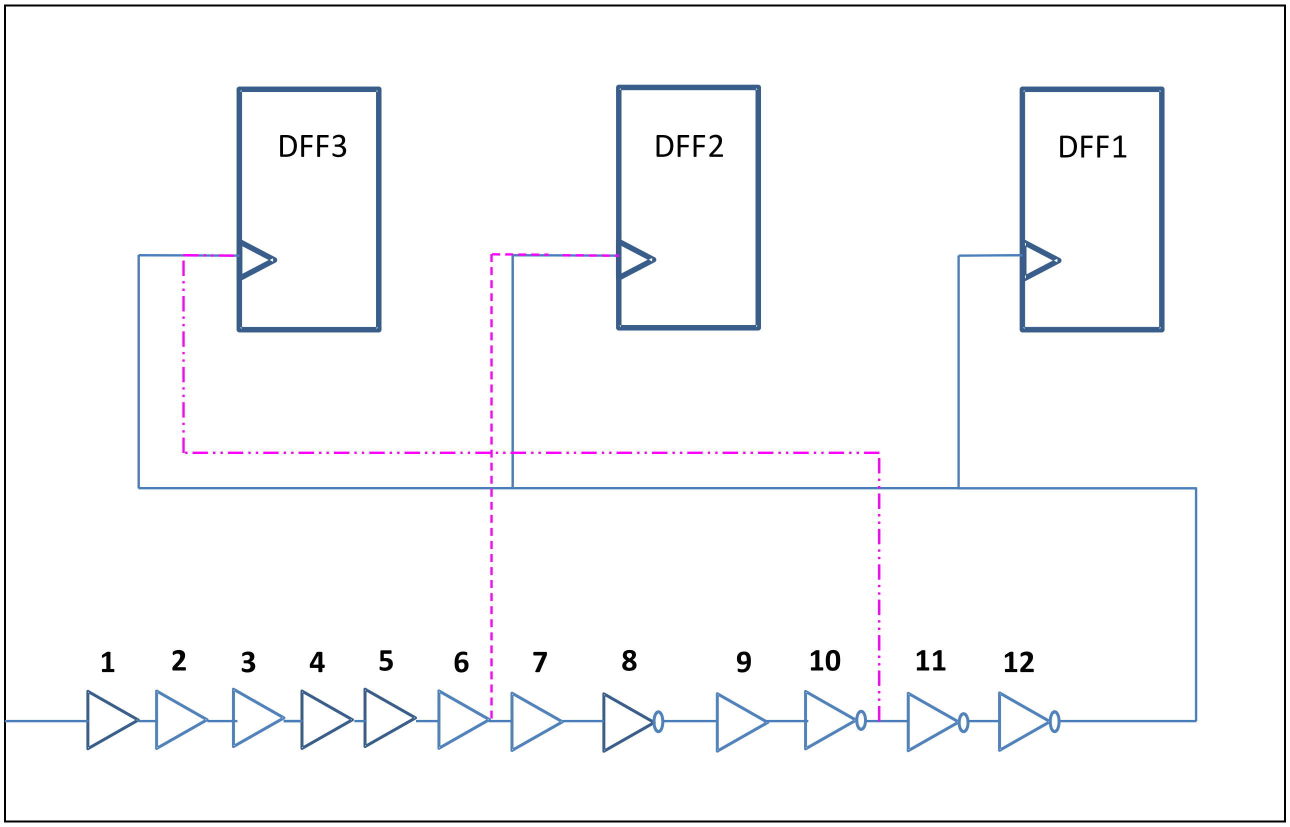

Figure 2 – Clock tree re-stitching with 1st level & 2nd level borrowing

With Figure 2, what we start with is the violation of DFF2→DFF1 path. For some reason the violation is not fixed even after the PT DMSA *fix_eco_timing*. The sizing of DFF2->DFF1 Data Path is not sufficient to fix the setup negative slack. We will perform the following steps to do the Clock Path ECO:

- DFF1 is the endpoint of the violation path. There may be multiple violation paths with one single end point. Report all the violating start points (DFF2s) with this endpoint. The violation slack from different start points should have different numbers.
- Make a list of the slack values from all of the violating paths. Use reasonable interleaves to sort the list into several buckets. Make a decision on: How many re-stitching points we want to implement? For example, we want to make 2 re-stitching point for: 400ps & 800ps.
- Set up the borrowing target# as: violation negative slack# + eco-fix margin (~200ps). We need this extra eco-fix margin to compensate the SPEF distraction from the ECO place & route impact.
- Use the 1st level borrow point (DFF2) as timing report endpoint to check how much setup margin this path (DFF3→DFF2) has. We want to see a good amount of setup margin for this path for us to borrow.

- Choose a good re-stitching point on the clock tree structure. Insertion delay is a concern, but we also want to create minimum slew impact to the current clock tree. Transition report is also a help on top of PD GUI analysis. With the clock path example in Example 2, *gclk__L6_I1/Y* & *gclk__L10_I2/Y* are two good spots for re-connection.
- Implement the disconnect/connect for the planned CK points in ECO.tcl file.
- Add two buffers before the re-stitched CK pins if the new connection is far apart.
- Always run Formality with the disconnect/connect eco activities.

```
Point                       Cap      Trans       Incr       Path
-------------------------------------------------------------------
----------------------------------------------------
CLK (inout)                        2.5484   0.0001   0.0000 &   0.0000 r
io_ring_i/clk_b/DOUT (BD33LV_8_A) 0.6274   0.2401   0.8123 &   0.8123 r
oclk__L1_I0/Y (CLKINVX40M)         0.0457   0.0841   0.0868 &   0.9756 f
oclk__L2_I0/Y (CLKINVX40M)         0.1912   0.1188   0.0950 &   1.0710 r
clkgen_b/gfmux_i/oclk__Fence_I0/Y (CLKBUFX40M)
                                   0.0631   0.0685   0.1430 &   1.2196 r
clkgen_b/gfmux_i/gclk_0/g16/Y (AND2X12M)
                                   0.0458   0.1115   0.1574 &   1.3783 r
clkgen_b/gfmux_i/g23/Y (OR2X8M)    0.0165   0.0702   0.1168 &   1.4960 r
clkgen_b/gfmux_i/gclk__Fence_I1/Y (CLKBUFX40M)
                                   0.1430   0.1120   0.1564 &   1.6524 r
clkgen_b/gfmux_i/oclk (gfmux) (gclock source)
                                            0.1120   0.0000 &   1.6524 r
gclk__L1_I1/Y (CLKBUFX16M)         0.0940   0.1489   0.1964 &   1.8497 r
gclk__L2_I1/Y (CLKBUFX20M)         0.0937   0.1270   0.1954 &   2.0475 r
gclk__L3_I1/Y (CLKBUFX20M)         0.0942   0.1270   0.1887 &   2.2383 r
gclk__L4_I1/Y (CLKBUFX20M)         0.1195   0.1517   0.2034 &   2.4440 r
gclk__L5_I1/Y (CLKBUFX32M)         0.0973   0.0976   0.1729 &   2.6200 r
gclk__L6_I1/Y (CLKBUFX40M          0.1057   0.0923   0.1528 &   2.7753 r
gclk__L7_I1/Y (CLKBUFX40M)         0.1156   0.0981   0.1542 &   2.9326 r
gclk__L8_I1/Y (CLKINVX40M)         0.2517   0.1502   0.1243 &   3.0605 f
gclk__L9_I6/Y (CLKBUFX40M)         0.0538   0.0595   0.1526 &   3.2152 f
gclk__L10_I2/Y (CLKINVX40M)        0.0582   0.0489   0.0500 &   3.2659 r
gclk__L11_I0/Y (CLKINVX32M         0.1633   0.1208   0.0958 &   3.3625 f
gclk__L12_I1/Y (CLKINVX40M)        0.2425   0.1509   0.1215 &   3.4863 r
....
....
```

Example 2 – Clock path report with Cap & Trans

In some cases (~ 10% chance), 2nd level of borrowing is needed. This means, if the 1st level margin is less than the borrowing target, DFF3->DFF2 path will become a setup violation path, then we also need to put DFF3 into the re-stitching list.

We want to do as little as possible clock tree stitching. To avoid 2nd level borrowing, rerun DMSA ECO after 1st level clock re-stitching. Utilize the PT_ECO data path sizing features for potential 2nd level setup fix. In Figure 2, since the path DFF3->DFF2 is exposed to PrimeTime ECO as first time during the (1st level borrowing + PT_ECO), the PT_ECO can definitely do something on the DFF3->DFF2 data path. The potential 2nd level setup violation has good chance to be fixed, or at least be reduced after the (1st level borrowing + PT_ECO).

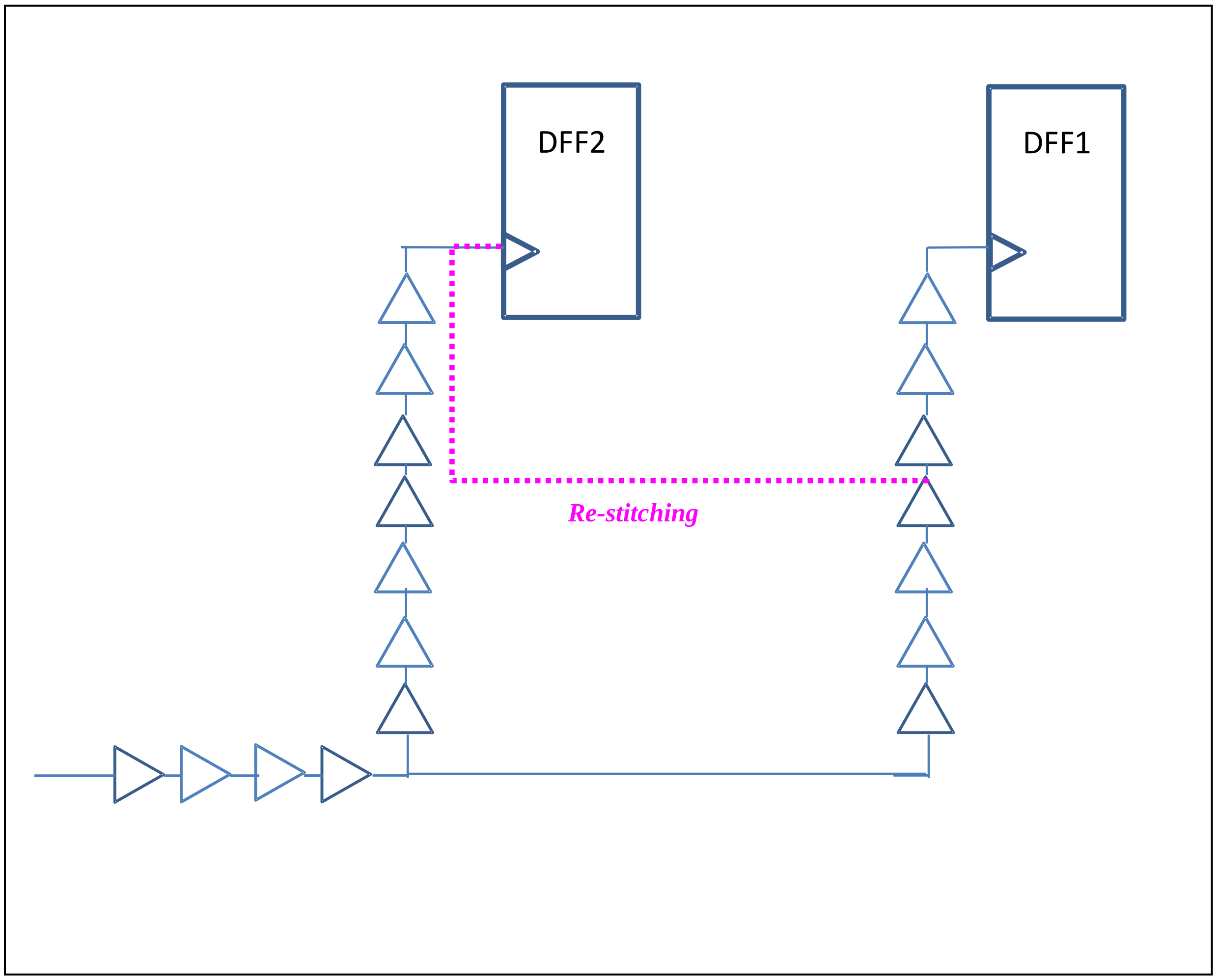

Figure 3 – Extra CRPR timing bonus with re-stitching

- ***Tips:*** We can get the extra CRPR timing bonus for re-stitching. In Figure 3, by re-arranging the launch & capture clock (DFF2/CK & DFF1/CK) to the same CTS branch (DFF1/CK branch), we can easily bypass the CRPR penalty, and get less tool calculation pessimism. In some cases, we can get ~120ps extra bonus on this in-depth study re-stitching.

- ***Point of interest for Synopsys tool development team***:
  . Is it a bug (or feature) for "WLM used for disconnected nets"? Could that be fixed?
  . Current 1st level setup borrowing and 2nd level setup borrowing analysis utilize some scripts, but they have room for more automation. Synopsys should be able to implement this "check margin + check stitch point" in a more efficient way.
  . "CRPR extra bonus" could be an option for high weight critical path.

Solution III: Clock tree in-law unit

It is not a good practice to touch the clock tree too much. There is tipping point where the CTS clock tree will become unbalanced. Routing of data path and clock path also has risk of being disturbed too much. So, "clock tree in-law unit" could be an option when there are big amount of DFFs need to reconnect to the clock tree.

We borrow the term "in-law unit" because it perfectly describes what we intend to have.
With a house, when we talk about an "in-law unit", we mean:

. An "in-law unit" is built in the backyard, if the main house does have spare space.
. It off-loads some of the functions of the main building.
. We want the "in-law unit" to follow the same fashion of main building. It should not impact the style of the main building. In best case, people even may not notice that this structure is an add-on.

So, with this "clock tree in-law unit", we mean:

. A "clock tree in-law unit" is built aside of main CTS structure, if the current design has spare space.
. It off-loads some of the functions of the main CTS.
. We want the "clock tree in-law unit" to follow the same fashion of CTS. It should not impact the balance of the current chip level CTS. In best case, the setup/hold timing report of other non-ECOed path should not see any change.

It should not be the first choice for the clock tree timing fix. We use it because in our design there is a group of stubborn logic paths. Those logic paths cannot been fixed by *Solution I* & *Solution II*.

Since it is an ECO with bigger scope than regular timing ECO, we need to communicate with front-end designer for understanding the path logic and ask the question: Is there any other option to fix it rather than a “clock tree in-law unit”? In this type of ECO, agreement between frond end/STA/PD is very important.

To bring in minimum impact to the CTS clock tree, in Example 2, `gclk__Fence_I1/Y` is picked as the root of “in-law unit”. ICC tool can be used to build the in-law ECO clock structure. Targeting insertion delay is calculated in the same way as descript in *Solution II*.

The information passed to back-end tool includes:
- List of CK endpoints for ECO.
- Target insertion delay number for the “clock tree in-law unit”.

Because this is a brand new clock structure, we expect to have two rounds of SPEF->ECO to get the right insertion #.

For those 3 Solutions in this paper, *Solution III* is the one which needs more effort for decision process and preparation. Since we utilize the ICC help, the exact ECO is less human interactive, hence the *Solution III* ECO results are actually more reliable.

- ***Point of interest for Synopsys tool development team***:
  . There is manual work and iterations between PrimeTime and ICC tool for the ECO clock structure building. It will improve the turn-around-time if it could be automated by Synopsys.
  . This new feature is not easy to implement. ICC should be a better place to host this feature.

4. Beyond ECO

When we get some issues in ECO stage, we need to understand the scope of the right solutions. That means the STA engineer needs to have certain understanding of both back-end tool and RTL field. One example is the constraint issues we mentioned in Chapter 2. STA engineer needs to have the ability to identify the constraint related PrimeTime report problems (for example, a clock group issue), and be able to communicate with the front-end team and system level team to find a good fix on constraint to make the project moving forward.

Thinking out of the PrimeTime tool sometime can find the real cause of a hard-to-fix path, and hence find a right solution for the problem.

Risk Factors with clock path ECO

The severe risk factor for ECO implementation is the design utilization rate. For a congested design, high utilization or tight routing resource will make the design have no tuning space for ECO.tcl legalization.

In some cases we run into a dead end for the timing ECO fix. There are possibilities of Prime-Time ECO.tcl not converging in the PD tool. More expensive iteration loops will be suggested for those cases. One example is to change the floorplan and chip size. That situation can be avoided by putting more design margin in the early back-end planning stage.

5. Acknowledgements

We would like to thank Ajay Bavle (SNUG), Khem Pokhrel (SNUG) and Bernard Miller (Synopsys) for helping us reviewing the paper. We appreciate the management support and inspiration from Anam Haque and Charlie Kahle on the initial of this paper. We want to acknowledge the help and support from Vijay Peruri, Lesley Goodwin and Bill Johnston during this exciting project.

6. References

[1] PrimeTime Fundamentals User Guide
[2] Design Compiler User Guide
[3] IC Compiler Implementation User Guide