

TOGA NETWORKS

STA in multi-GHz custom circuits made easy with NanoTime

Udi Nir

Toga Networks

Hod Hasharon, Israel

www.toganetworks.com

ABSTRACT

High speed digital blocks in mixed signal IPs are designed using full custom flow, to maximize the performance while meeting the required area and power constraints. On the other hand, industry trends such as IP quality, predictability and time to market suggest employing EDA tools wherever possible in the design. This paper will address the challenges in the STA domain, by examining NanoTime transistor level static timing flow, executed on a CMOS levels 30GSPS serializer. Although the employed latch topology is not automatically recognized by NanoTime, using manual latch marking and custom timing checks, 100% of the IN2REG and REG2REG timing arcs can be covered. NanoTime capabilities such as parasitic back annotation and POCV are discussed. Tool accuracy is compared against automatically generated spice deck and original schematic dynamic simulation. The work is summarized with a list of conclusions and follow-up work.

Table of Contents

1. Introduction 4
2. STA flow for custom high speed digital circuits 4
3. Case study – 30Gbps serializer 6
3.1 Serializer block description 6
3.2 High speed dynamic latch 8
3.3 NanoTime customized configuration steps 9
3.3.1 Create differential clocks 9
3.3.2 Mark custom latches and flops 9
3.3.3 Generated clock 10
3.3.4 POCV settings 11
3.3.5 Custom timing checks 11
3.3.6 Increasing NanoTime accuracy 12
4. Flow debug and Results Analisys 12
4.1 Reports review 12
4.1.1 report_topology 12
4.1.2 report_transistor_direction 12
4.1.3 report_clock 13
4.1.4 report_clock_arraivals 13
4.1.5 report_clock_network 13
4.2 Parasitics annotation 13
4.3 Debugging incorrect path timing 13
4.4 Correlating path timing with spice simulation 14
Table 3 summarizes the full path delay differences between NanoTime report and write_spice deck measurements for 4 other selected paths. The accuracy here is also within the 5% limit. 16
5. Conclusions 16
6. References 17

Table of Figures

Figure 1: Typical analog IP sub blocks and chip integration levels 5
Figure 2: Analog IP timing flow with NanoTime/PrimeTime 5
Figure 3: NanoTime detailed flow 6
Figure 4: 30Gbps serializer 7
Figure 5: 2TO1 MUX cell and timing diagram[1] 7

Figure 6: DTG dynamic latch 8
Figure 7: Differential clocking scheme 9
Figure 8: Custom latch marking 10
Figure 9: Custom flip-flop marking 10
Figure 10: Generated clock with differential source 10
Figure 11: SS/125C/800mV N type delay variation 11
Figure 12: ck_div_out transition difference marked 14
Figure 13: NT timing report of compared path 15
Figure 14: Path delay in block level dynamic simulation 15

Table of Tables

Table 1: Serializer custom logic statistics 8
Table 2: Path timing correlation to dynamic simulation 16
Table 3: Selected full paths timing correlation to write_spice results 16

1. Introduction

Designing mixed signal IPs in advanced node technologies is extremely challenging and expensive. Accuracy and efficiency are required at all stages of the design flow. Looking at the high speed digital components of the IP, STA flow traditionally utilize measurements extracted from SPICE/ FastSpice dynamic simulations. Such flow relies heavily on circuit team expertise, since dynamic simulation is vector-dependent and usually executed on limited set of critical paths modeled into a cross section. Result validation is a tedious work, and usually requires ongoing maintenance of a measurement extract script. Furthermore, the verification quality can be insufficient and should be backed up with slow statistical runs to cover OCV. Post layout parasitic runs with coupling capacitors require extended run time and more attention to internal nodes behavior.

For these reasons, a transistor-level static timing tool like NanoTime can be used to perform the timing analysis and model generation of custom digital circuit blocks. Using such a tool can increase statistical/ multi-corner coverage, while maintaining near-SPICE accuracy and reducing the required run time and review work. Dynamic simulations measurements are used as golden results which NanoTime timing reports are correlated with.

In this paper, a feasibility study of NanoTime STA flow for advanced node custom CMOS serializer will be described. The paper is organized as follows: Chapter 2 will describe the STA flow; Chapter 3 will describe the serializer structure and building blocks, as well as the NanoTime customized configuration steps; Chapter 4 will discuss the results analysis; And chapter 5 will conclude the work.

2. STA flow for custom high speed digital circuits

Typical mixed signal IP is presented in Figure 1 . The analog IP is composed of semi custom digital block, custom digital block and analog blocks at the far end. The analog IP is integrated inside the IP digital top level, which is integrated in the chip top level. As mentioned above, in traditional flow the verification of the timing arcs in the analog IP are done as part of the dynamic simulation of analog side. Automated STA is run at the IP and chip level, using a timing model of the analog IP, covering one or more stages in the semi-custom digital block.

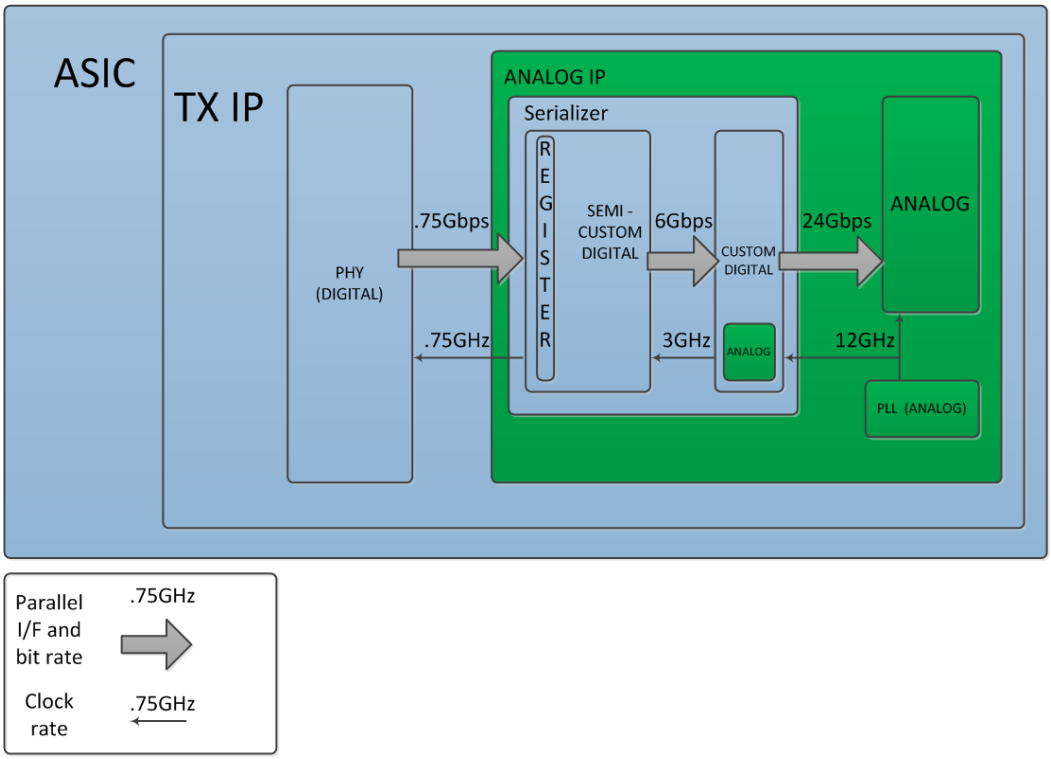


Figure 1: Typical analog IP sub blocks and chip integration levels

With NanoTime, the analog IP STA coverage can be extended to cover all digital components of the analog IP, as illustrated in Figure 2. Because of the sensitivity of the high speed digital circuit, an overlap is maintained between the NanoTime run and the dynamic simulation. NanoTime results are correlated to the higher accuracy dynamic simulation results, when OCV and SI analysis are turned off.

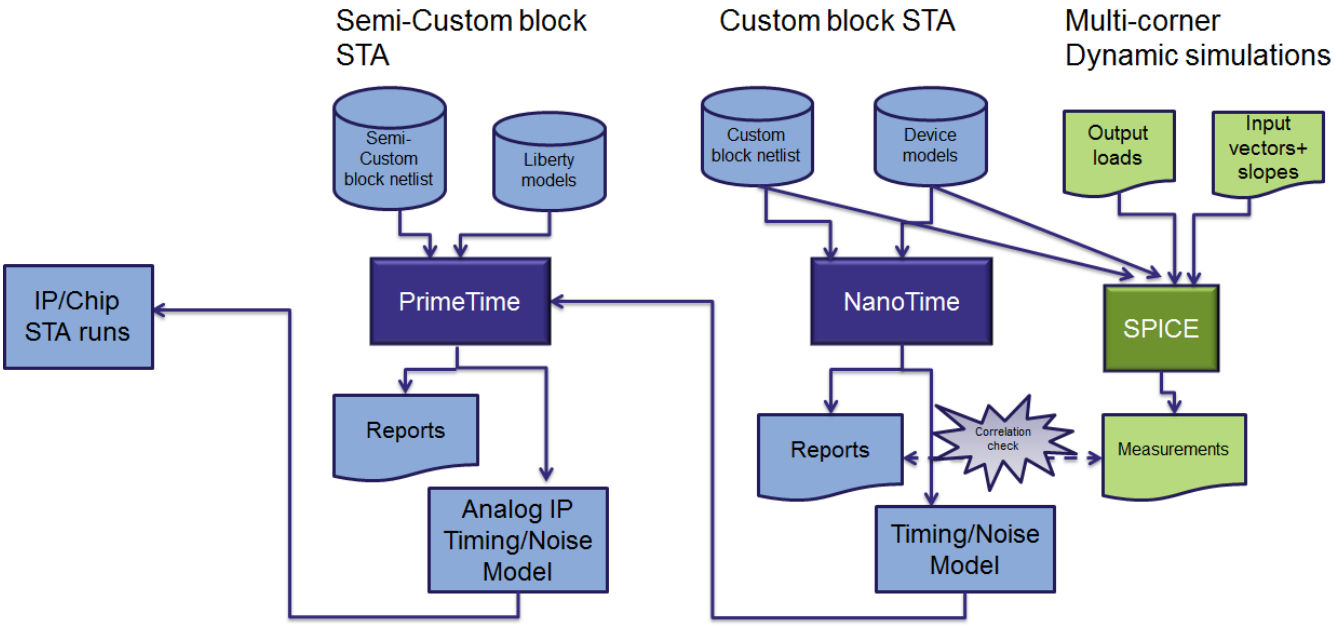


Figure 2: Analog IP timing flow with NanoTime/PrimeTime

Figure 3 illustrates NanoTime detailed STA flow on the custom digital block. The flow is similar to PrimeTime flow, with few custom steps, which require close attention from the designer:

- Setting single-ended/ differential/ generated clocks.
- Marking custom latches , not automatically recognized
- Setting POCV values taken from statistical device simulations.
- Adding custom timing checks for custom structures
- Iterating the match_topology stage until clock network, transistor direction and all endpoints are defined correctly

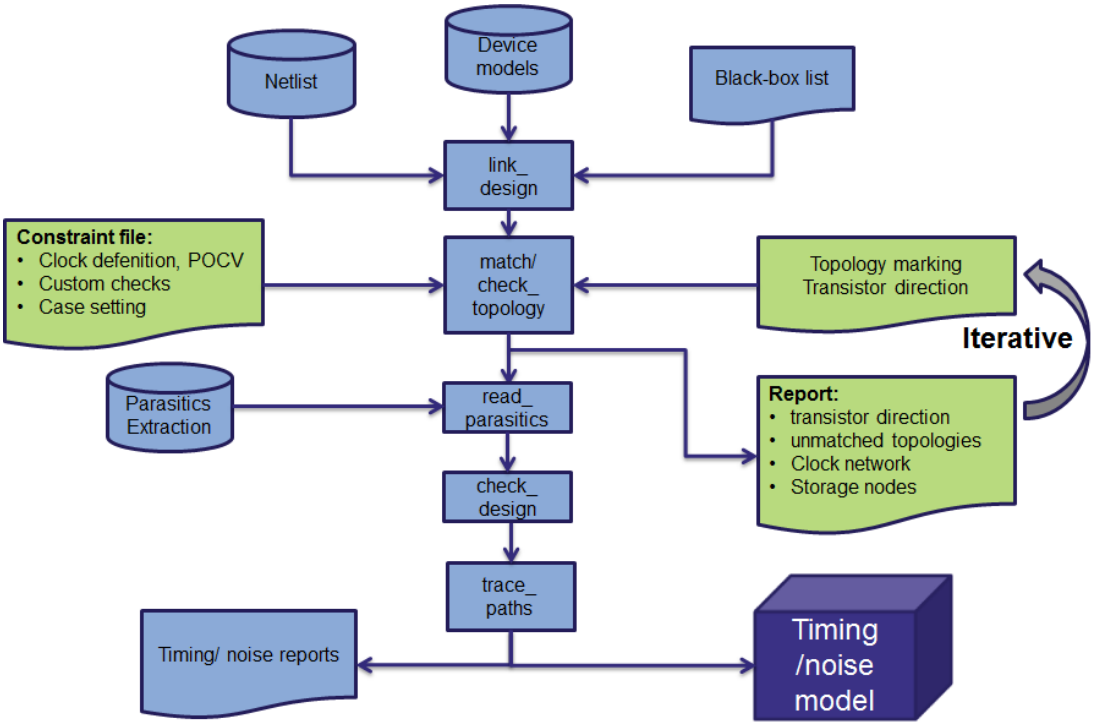


Figure 3: NanoTime detailed flow

3. Case study – 30Gbps serializer

3.1 Serializer block description

The suggested flow was executed on a FFE enabled PAM4 30Gbps serializer used in 60GBps SERDES. The integrated custom logic is described in Figure 4. The core of the custom side serializer is a tree-type mux composed of 4 stages of 2:1 flop-mux cells. Each such tree serializes 16 inputs into a two bit stream (PAR2<1:0>). These bits are delayed by 1 unit interval (half cycle of the PLL clock), to generate the PRE/MAIN and POST cursors of the FFE. A final mux creates a single bit stream out of each PAR2<1:0> bits, running @ 30Gbps. Two of such tree-muxes together generate a total of 6 outputs of the serializer to the driver.

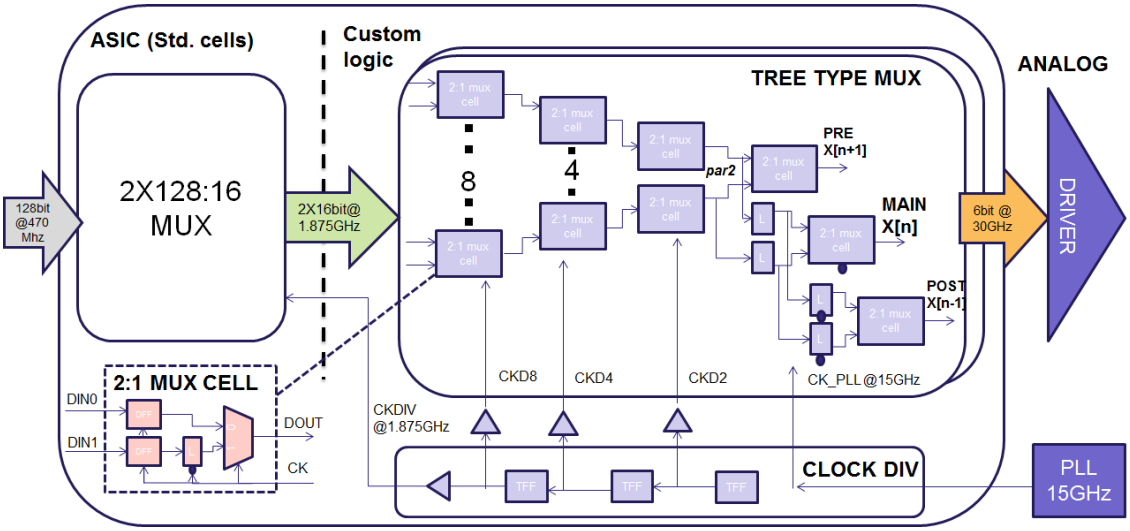


Figure 4: 30Gbps serializer

The basic 2to1 Mux cell (Figure 5) is composed of two flops to sample the data, a latch on ***Do*** and a mux. The latch is added to prevent noise on mux output and for increased hold margin. For low deterministic jitter on driver input , setup and hold of ***De/Do'*** should be verified against ***S*** (C2clk) , to make sure serializer output bits are clock driven and not data driven. Similar checks are performed on previous stages to improve design robustness and reduce noise.

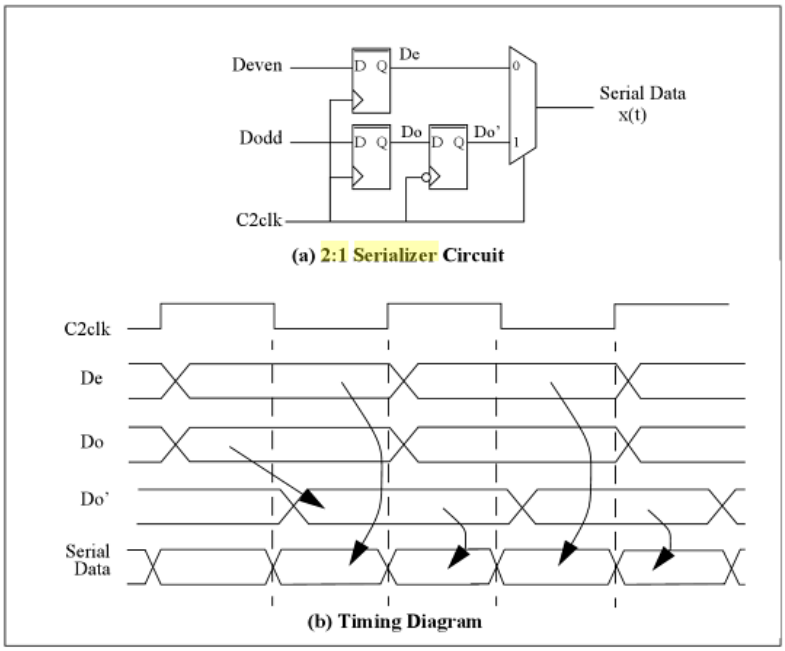


Figure 5: 2TO1 MUX cell and timing diagram[1]

Table 1: Serializer custom logic statistics summarize the serializer custom logic statistics. Although the input clock rate is 15G/ 66ps cycle time, because of the half rate topology, the final 2to1 stage has timing paths from rising clock edge to falling, which are 33ps long.

Table 1: Serializer custom logic statistics

Parameter	Value
Sequential cells (FF/ LTC)	**200**
Logic cells	**1000**
Timing endpoints (FF/ LTC/ MUX)	**250**
Input Clock rate*	**15GHz/ 66p**
Uncertainty	**5p**

3.2 High speed dynamic latch

CMOS Dynamic latches are used in high-speed digital blocks to save power over CML approach. The serializer stages above 1.875GHz are utilizing Dynamic Transmission Gate (DTG) latch topology[2]. The removal of the feedback devices and the use of differential clocks are minimizing the clock-to-Q delay of the latch to the range of the technology node inverter delay. The time limited storage is utilizing the latch net parasitic capacitance. The latch circuit is illustrated in Figure 6, compared with static latch cell.

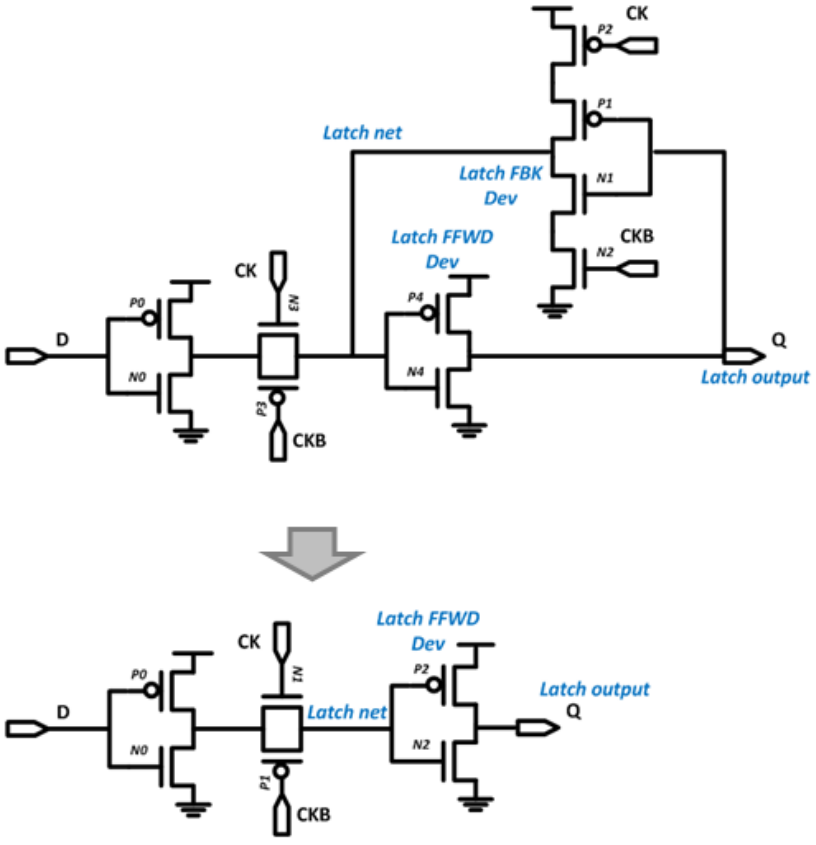


Figure 6: DTG dynamic latch

3.3 NanoTime customized configuration steps

NanoTime requires some manual work to setup STA on custom circuits.

3.3.1 Create differential clocks

The DTG latch is using differential clock – two full swing clock signals having complimentary polarity with femtosecond skew between them. NanoTime supports definition of differential clock, by defining one of the clocks as a single ended clock and then marking the other one as the differential pair:

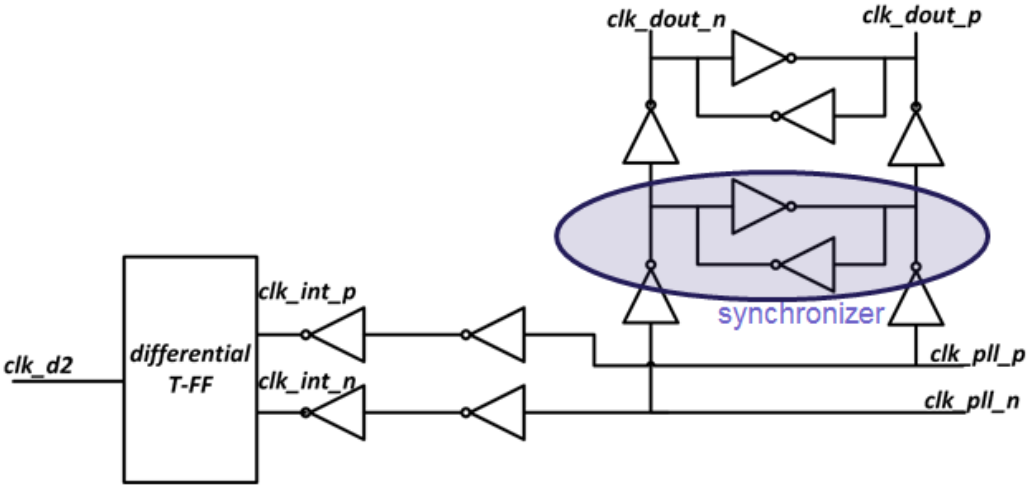


Figure 7: Differential clocking scheme

```
create_clock -name clk -waveform {0 0.033} -period 0.066 [get_ports
clk_pll_p]
set_differential [get_ports {clk_pll_p clk_pll_n }]
```

Differential clocking usually utilizes skew-reducing synchronizers – usually back-to back inverters (latch structure), connected between the output of each buffer stage of the differential clocks. Their purpose is to reduce differential noise and skew. Standard synchronizers are automatically recognized by NanoTime. Custom synchronizers can be marked manually:

```
mark_differential_synchronizer -name synchronizer1 -transistors {…}
```

NanoTime determines differential skew automatically. Single ended clock features as CRPR(PBSA), ETM, SI, MIS, Active Miller and more are also available for differential clocks.

3.3.2 Mark custom latches and flops

The DTG latch topology is not automatically recognized by NanoTime , but can be marked manually:

```
mark_latch -latch_net qn -inputs d -output q -clock {xp1/g xn1/g} -
feedforward {n2 p2} -name DynLatch -setup_to latch_net
```

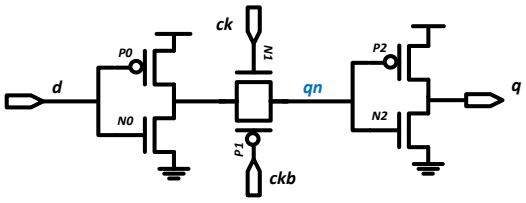


Figure 8: Custom latch marking

The marking is incomplete since feedback devices, and sometime feedforward devices are omitted in DTG latch. This creates a warning but doesn't disturb the timing analysis. `foreach_in_collection` can be used to mark all latches iteratively, if specific naming convention is used for latch cell or instance name.

Two cascaded latches can be marked as flip-flop:

`mark_flip_flop -master_latch` **`m_net`** `-slave_latch` **`s_net`** ←*Master latch FF devices omitted*

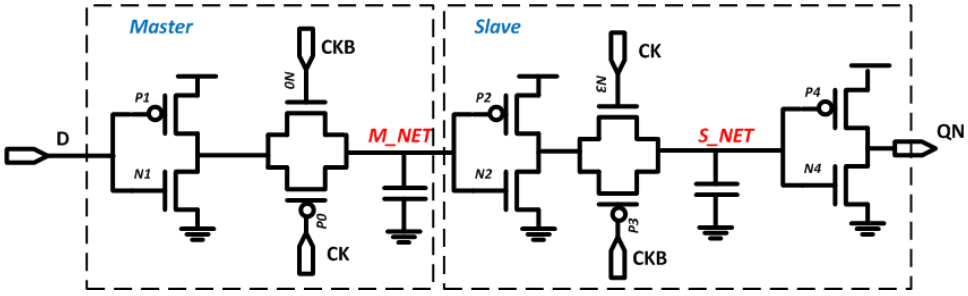


Figure 9: Custom flip-flop marking

3.3.3 Generated clock

A clock divider utilizing custom differential flip-flop is illustrated in Figure 10 . In NanoTime the clock is defined by:

```
create_generated_clock -name clk_d2 -source [get_ports clk_pll_p]
-edges {1 3 5} [get_pins x0/xp5/g x0/xn5/g]
```

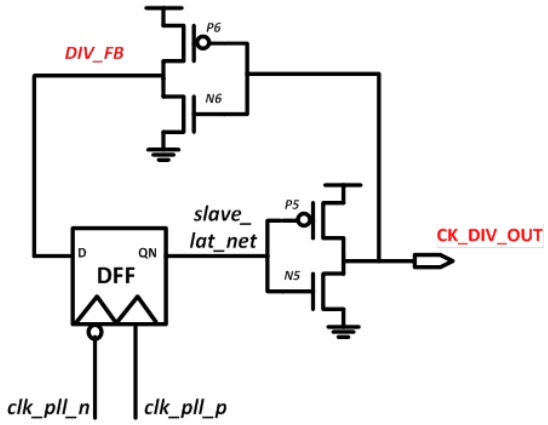


Figure 10: Generated clock with differential source

The clock source is one of the ports used for the definition of the complimentary clock. The generated clock should be marked on the slave latch net, by selecting both P and N type device gates driven by this net – otherwise only one edge will be propagated to the generated clock network. After the creation of the clock, div_fb net becomes part of the clock network. The following command should be used to prevent clock to data propagation warning:

```
mark_clock_network -stop div_fb
```

3.3.4 POCV settings

One of the great benefits of STA over dynamic simulations is the ability to take into account on-chip variation in every path of the design, without generating huge amount of statistical data. Parametric OCV can be defined per device type/ width / length/ voltage level and check (min/max). Variation value is the percentage of one sigma out of the device delay. This information is taken from the device reference manual or by running very fast statistical dynamic simulation. In the example below, N type delay @ SS corner/ 125C/ 800mV supply is checked to have 10.4ns average delay and 0.28ns 1 sigma variation.

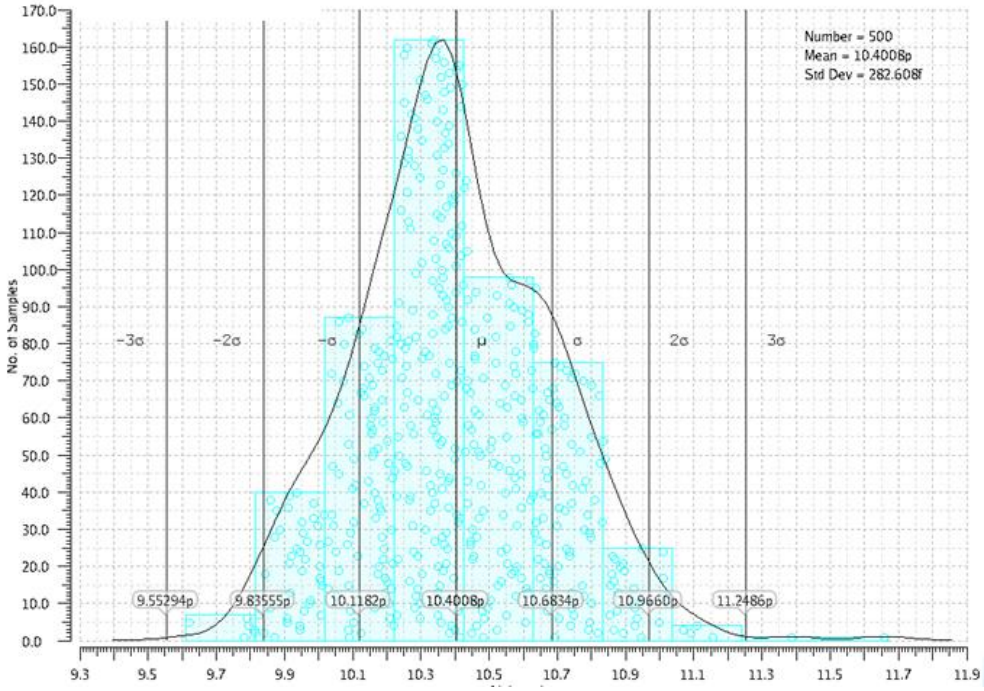


Figure 11: SS/125C/800mV N type delay variation

The 1 sigma variation is ~3% of the nominal delay in this corner. It is applied by using the command

```
set_variation_parameters -type default  -variation 0.03
```

3.3.5 Custom timing checks

As elaborated in section Serializer block description3.1 , setup and hold should be checked on MUX inputs De, Do' in Figure 5. The mux is not recognized automatically by NanoTime as legitimate endpoint. It is constrained manually with custom timing checks of setup and hold. Setup and hold values are calculated in dynamic simulation by checking the margin required for the input to be stable with respect to the pulse width of the mux select. For mux phase 0, the checks are defined as follows:

```
create_timing_check -from de  -fall_to c2clk -setup 0.010
create_timing_check -from de  -rise_to c2clk -hold 0.010
```

3.3.6 Increasing NanoTime accuracy

Increased accuracy is required for correct timing of differential circuits. Synopsys recommends to use the script `set_accuracy_effort -level` **`high/ultra`**, which is defined in an annex Tcl file. Among others, this script enables:

- Nonlinear waveform analysis
- Miller effect capacitance
- Reduce SI differential pessimism
- Dynamic Clock Simulation(DCS) – entire differential simulation unit is contained within DCS region

Differential PBSA analysis should be enabled to remove differential common point pessimism:

```
set pbsa_allow_reconvergent_common_net true
trace_paths –pocv –pbsa
```

4. Flow debug and Results Analisys

4.1 Reports review

Good path analysis requires complete clocking and sequential topology recognition. Reviewing the following reports helps to clean up the design and verify the flow setup.

4.1.1 report_topology

Check sequential, clock gate and MUX/ tgate topology counts:

```
Report : topology
Design : P2S
                                        User        Global
Structure       Auto        Manually    Library     Library
Type            Marked      Marked      Marked      Marked      Total
-----------------------------------------------------------------------------------------------
--------------------
clock_gate      1           0           0           0           1
feedback        2           0           0           0           2
latch           0           96          0           0           96
mux             32          0           0           0           32
flipflop        0           32          0           0           32
```

4.1.2 report_transistor_direction

Verify no transistor direction is left unresolved.

4.1.3 report_clock

Verify correct period, waveform and attributes:

```
Clock      Period    Waveform         Attrs            Sources
-------------------------------------------------------------------------------------------
clk        0.066     {0.000 0.033}     pD             {clk_pll_p clk_pll_n}
clk_d2     0.132     {0.000 0.066}     pG             {x0/p5/g x0/xn5/g}
clk_d4     0.264     {0.000 0.132}     pG             {x1/p5/g x1/xn5/g}
```

4.1.4 report_clock_arraivals

Verify No missing clock nets, clock domains, clock arrival times:

```
Net          Min Rise   Min Fall  Max Rise Max Fall  Period  Attrs Clock
-------------------------------------------------------------------------
clk_dout_n   0.006      0.038     0.009    0.042     0.066   D     clk
clk_dout_p   0.038      0.006     0.042    0.009     0.066   D     clk
clk_int_n    0.006      0.038     0.009    0.042     0.066   D     clk
clk_int_p    0.038      0.006     0.042    0.009     0.066   D     clk
clk_d2       0.070      0.003     0.072    0.011     0.132         clk_d2
clk_d4       0.137      0.005     0.140    0.010     0.264         clk_d4
```

4.1.5 report_clock_network

Verify no invalid sequential points.

4.2 Parasitics annotation

By the time this work was done, top level extraction was not available. The read_parasitic command was used to read in some of the lower hierarchies, which had RC extraction ready. This command is issued after the check_topology step.

```
read_parasitics –increment -path Xhier1 -format DSPF hier.dspf
```

If layout only devices exist in the extraction, following environment variable is used to define the prefix of such elements so NanoTime can recognize them. If not used, an error is issues and it can lead to removal of annotated parasitics from schematics nets having RC networks connected to these elements.

```
set parasitics_xref_layout_instance_prefix layout_only
```

4.3 Debugging incorrect path timing

During the timing closure there were few paths with big difference between the clock path and data path transition time in one of the nets belonging to the CRPR common path. One of these paths is shown in Figure 12. In the data path section, the ck_div_out/g pin (GC net) transition is 7.4p and in the clock path section it is 187ps.

```
Startpoint:         xck_d2_div_out/xmm1_1/g
Endpoint:           xmux_4to2<7>/xhmuxb2/xm2q/d
Path Type:          min
Constraint:         user specified hold check
PBSA Common Net:    clk_p
PBSA Complement:    clk_n

                                                Coeff    Xtalk
    Path      Incr    Adjust       Var          adj     Delta     Trans      Cap  NT    Point
-------- --------- --------- --------- ---------- --------- --------- -------- ---- -- ------------------------------
  0.0000             0.0000                                     0.0400   0.0120  C    f clk_p (inout)
  0.0000    0.0000                0.0000     0.0000    0.0000   0.0400   0.0120  C    f xck_p_int0<0>/xmm0_1/g (INVD2)
  0.0077    0.0077                0.0000     0.0000    0.0000   0.0148   0.0038  C    r xck_p_int2<0>/xmm1_1/g (INVD8)
  0.0117    0.0040                0.0000     0.0000    0.0000   0.0053   0.0077  C    f xdiv_14to7_dff/xdsltc/xm1/g (p2s_cpl_ltcn_uu_4)
  0.0155    0.0038                0.0000     0.0000    0.0000   0.0074   0.0038  GC   r xck_d2_div_out/xmm1_1/g (INVD6)
  0.0188    0.0033                0.0004     0.0000    0.0000   0.0039   0.0071  C    f xinv_ck_d2_b/xmm0_17g (INVD12)
  0.0234    0.0046                0.0005     0.0000    0.0000   0.0066   0.0339  C    r xmux_4to2<7>/xltcn1_1/xm3/g (p2s_cpl_ltcn_uu_1)
  0.0266    0.0032                0.0004     0.0000    0.0000   0.0044   0.0007  L    f xmux_4to2<7>/xbit1_buff/xmm0/g (INVD1)
  0.0296    0.0030                0.0004     0.0000    0.0000   0.0060   0.0017       r xmux_4to2<7>/xhmuxb2/xm2q/g (p2s_cpl_ltcn_uu_2)
  0.0334    0.0038                0.0005     0.0000    0.0000   0.0058   0.0012  U    f xmux_4to2<7>/xhmuxb2/xm2q/d (p2s_cpl_ltcn_uu_2)
  0.0319             -0.0014                                                            POCV adjustment
  0.0319                                                                                data arrival time
            0.0334    0.0141    0.0010               0.0000                             Total

  0.0000             0.0000                                     0.0400   0.0120  C    r clk_n (inout)
  0.0000    0.0000                0.0000     0.0000    0.0000   0.0400   0.0120  C    r xck_p_int0<1>/xmm1_1/g (INVD2)
  0.0085    0.0085                0.0000     0.0000    0.0000   0.0148   0.0038  C    f xck_p_int2<1>/xmm0_1/g (INVD8)
  0.0118    0.0033                0.0000     0.0000    0.0000   0.0055   0.0076  C    r xdiv_14to7_dff/xdsltc/xm3/g (p2s_cpl_ltcn_uu_4)
  0.0160    0.0041                0.0000     0.0000    0.0000   0.1871   0.0038  GC   r xck_d2_div_out/xmm1_1/g (INVD6)
  0.0285    0.0126                0.0008     0.0000    0.0000   0.0374   0.0071  C    f xinv_ck_d2_b/xmm0_17g (INVD12)
  0.0381    0.0095                0.0006     0.0000    0.0000   0.0165   0.0339  C    r xinv_ck_d2/xmm1_17g (INVD12)
  0.0447    0.0067                0.0004     0.0000    0.0000   0.0106   0.0287  C    f xmux_4to2<7>/xhmuxb2/xm3/g (p2s_cpl_ltcn_uu_2)
            0.0447    0.0160      0.0010               0.0000                             Total
  0.0497    0.0050                                                                      hold time
  0.0497    0.0000                                                                      clock uncertainty
  0.0497                                                                                data required time
-------------------------------------------------------------------------------------------------------------------------
  0.0319                                                                                data arrival time
 -0.0497                                                                                data required time
-------------------------------------------------------------------------------------------------------------------------
 -0.0178                                                                                slack (VIOLATED)
```

Figure 12: ck_div_out transition difference marked

The debug of this issue was done using write_spice command and XA (Fast Spice) simulation of the generated spice decks. To generate these decks reference_timing_path and prefix_path_id attributes of timing paths were used.

```
set data_path [get_timing_path …]

set clock_path [get_attribute $data_path reference_timing_path] ← use this attribute to get the clock path related to the data timing path

set data_prefix_path [get_attribute $data_path prefix_path_id] ← use this attribute to get the generated clock source path

set clock_prefix_path [get_attribute $clock_path prefix_path_id]

write_spice $data_prefix_path –output datapath

write_spice $clock_prefix_path –output clkpath
```

Observation of the spice simulation waveforms with WaveView showed that in clkpath the GC net is not reaching the supply rail in the clkpath deck. Looking at the deck netlist, the reason was clear – it had the GC P type device of the slave latch (Figure 9) in off state (The tool assumed this setting will cause WC timing). But since this latch is driven by differential clock, which are on/off simultaneously, this scenario should be waived. To waive it, the following flag should be set to true when running differential latch STA:

```
set topo_tgate_mark_all_pairs true
```

4.4 Correlating path timing with spice simulation

NanoTime's full path delays are reported to be within ±5% of HSPICE simulation of write_spice deck. To verify the validity of the path timing analysis done in this flow, a comparison was made between the report_path timing (Figure 13Figure 13: NT timing report of compared path

), HSPICE simulation measurement and measurements done on the same path in block level dynamic simulation run using a different spice simulator (Figure 14). The path selected for this comparison is the custom check created on final mux inputs. (3.3.5). It runs through circuit section which had parasitics annotated to it.

```
Startpoint:          clk_p (inout port)
Endpoint:            xp2s_mux_out<0>/x0/xhmux_p0/xm0/d
Path Type:           max
Constraint:          user specified setup check
PBSA Common Net:     ck_p_buf<6>

    Path      Incr    Adjust      Var    Xtalk Delta    Trans      Cap  NT    Point                                Net
-------- --------- --------- -------- -------- ------- ------- ---- --  ---------------------------------- ----------------------------------
  0.0000                0.0000                          0.0400  0.0201  C&   f clk_p (inout)                  clk_p
  0.0009    0.0009                      0.0000  0.0000  0.0400  0.0201  C&   f xck_tree/xck_inv2<0>/xmm0_1/g (invd12)
                                                                                                             clk_p
  0.0115    0.0106                      0.0000  0.0000  0.0173  0.0324  C&   r xck_tree/xckinv3<6>/xmm1_1/g (invd6)
                                                                                                             xck_tree/ck_p_i2<0>
  0.0220    0.0106                      0.0000  0.0000  0.0141  0.0170  C&   f xp2s_mux_out<0>/x0/xltcn0/xm1/g (p2s_cpl_ltcn_uu_4)
                                                                                                             ck_p_buf<6>
  0.0267    0.0046                      0.0002  0.0002  0.0123  0.0045  L&   r xp2s_mux_out<0>/x0/xl3inv/xmm1_1/g (invd2bwp16p90cpdulvt)
                                                                                                             xp2s_mux_out<0>/x0/dyn_bit0_n
  0.0319    0.0053                      0.0002  0.0004  0.0084  0.0025  &    f xp2s_mux_out<0>/x0/xhmux_p0/xm0/g (p2s_cpl_ltcn_uu_2)
                                                                                                             xp2s_mux_out<0>/x0/bit0_p
  0.0360    0.0041                      0.0002  0.0000  0.0061  0.0015  U&   r xp2s_mux_out<0>/x0/xhmux_p0/xm0/d (p2s_cpl_ltcn_uu_2)
                                                                                                             xp2s_mux_out<0>/x0/xhmux_p0/i_b
  0.0364                0.0004                                                  POCV adjustment
  0.0364                                                                        data arrival time
              0.0360    0.0004    0.0004  0.0007                                Total

  0.0355                0.0355                          0.0400  0.0201  C&   r clk_p (inout)                  clk_p
  0.0359    0.0004                      0.0000  0.0000  0.0400  0.0201  C&   r xck_tree/xck_inv2<0>/xmm1_1/g (invd12)
                                                                                                             clk_p
  0.0436    0.0077                      0.0000  0.0000  0.0173  0.0324  C&   f xck_tree/xckinv3<6>/xmm0_1/g (invd6)
                                                                                                             xck_tree/ck_p_i2<0>
  0.0531    0.0095                      0.0000  0.0000  0.0136  0.0170  C&   r xp2s_mux_out<0>/x0/xhmux_p0/xm3/g (p2s_cpl_ltcn_uu_2)
                                                                                                             ck_p_buf<6>
  0.0569                0.0003                                                  PBSA adjustment
              0.0176    0.0358    0.0000  0.0000                                Total
  0.0469   -0.0100                                                             setup time
  0.0419   -0.0050                                                             clock uncertainty
  0.0419                                                                        data required time
-------------------------------------------------------------------------------------------------------------------------------------------------
  0.0419                                                                        data required time
 -0.0364                                                                        data arrival time
-------------------------------------------------------------------------------------------------------------------------------------------------
  0.0055                                                                        slack (MET)
```

Figure 13: NT timing report of compared path

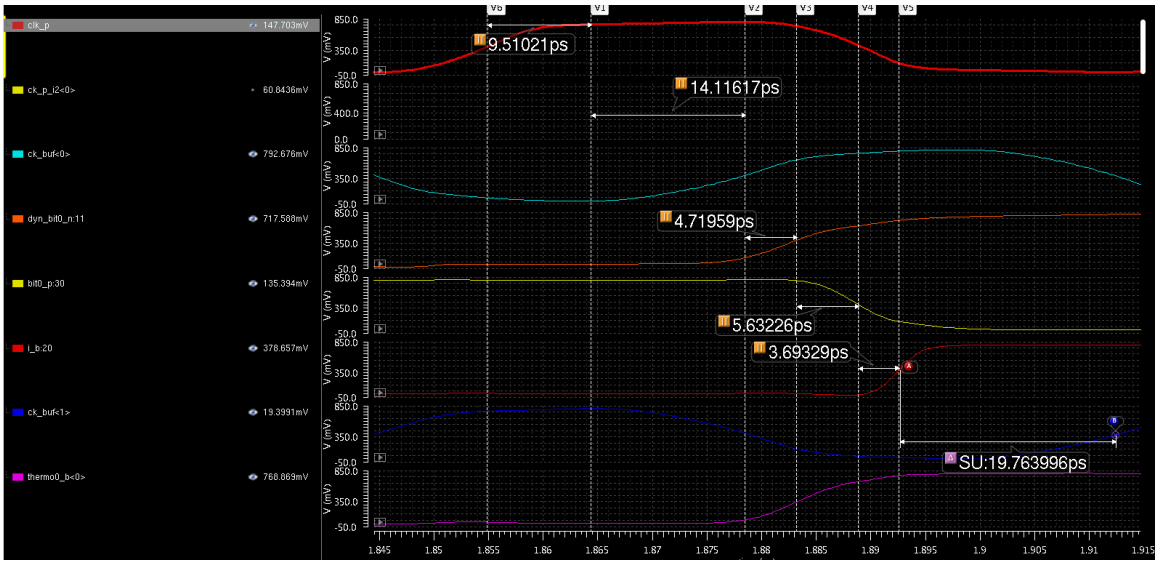


Figure 14: Path delay in block level dynamic simulation

Table 2 summarizes the differences between the stage and path delay calculated in each one of the methods. It can be seen that the entire path falls inside the advertised accuracy of +/- 5%. Intermediate nodes have higher differences, especially in the simulation done using other vendor tool. Stage delays, being small in nature, can manifest higher difference percentage though the numeric delta is quite small. Beside solver difference, it might be explained by a selection of a different RC network node for the measurement then the one used by NanoTime.

Another reason can be the linear ramp used as the input to the stage by the tool, while the dynamic spice simulation used prev. stage output.

Table 2: Path timing correlation to dynamic simulation

Stage	report_path	write_spice	Other vendor
clk_p→ ck_p_i2	11.5p	11.2p (-2.5%)	9.5p (-17.4%)
ck_p_i2→ck_buf	10.6p	10.1p (-4.7%)	14.1p (+33%)
ck_buf→ dyn_bit0_n	4.6p	5.0p (+8.7%)	4.7p (+2.2%)
dyn_bit0_n→ bit0_p	5.3p	4.5p (-15%)	5.6p (+5.6%)
bit0_p→ i_b	4.1p	4.2p (+2.5%)	3.7p (-10%)
Whole path: clk_p-->-i_b	36p	35p (-2.8%)	37.6p (+4.4 %)

Table 3 summarizes the full path delay differences between NanoTime report and write_spice deck measurements for 4 other selected paths. The accuracy here is also within the 5% limit.

Table 3: Selected full paths timing correlation to write_spice results

Path ID	report_path	write_spice
3138	36.1p	35.9p(-0.5%)
3114	36.4p	36.4p(-0%)
4432	28.4p	27.8p(-2.2%)
4411	28.4p	27.8p(-2.2%)

5. Conclusions

The proposed NanoTime flow was executed on 30GSPS serializer. All custom latches, differential clocking scheme and clocked mux topologies were recognized and timed correctly (pre-layout). Parasitic annotation was applied only partially, since full extraction was not available at the time this experiment took place. Timing report of a path with parasitics annotation was verified against spice simulations - both NanoTime generated spice deck and original schematic simulation. Overall path delay was found to be within (+/-5%). Some of the capabilities of the tools couldn't be evaluated at this stage, and will be deferred to the time when full extraction is available: These are SI delay and Noise, extensive timing path correlation and Extracted Timing Model (ETM) generation. However, It is evident that the original goal of increased timing coverage of custom IP can be achieved using NanoTime, with spice level measurements accuracy, and a run time which is a fraction of a single dynamic simulation (1 minute for NanoTime vs. 40 minutes for block level dynamic simulation) .

The effort required for the setup of NanoTime should not be under estimated. The tool has many options, flags, settings and accelerators, which should be adapted to reach complete and accurate timing coverage. However, once it is setup, the environment can be reused for similar designs without extra work.

Below are some recommendations, which can ease the debug work:

- Generate Clock failure – there is no warning if the clock is not created correctly, and after trace_paths clock waveform or clock_arrivals times are missing for some nodes. It can be understood from these report that there is a problem, but it's not clear what is the root cause.
- Net not completing full swing – as explained before, when the *topo_tgate_mark_all_pairs* was set to false, the clock path couldn't reach vdd because the pfet was off. Maybe NanoTime can check this and report, to save the time of dumping and running the spice deck.

6. References

[1] **High Speed Serdes Devices and Applications** David Robert Stauffer, Jeanne Trinko-Mechler, Michael A. Sorna, Kent Dramstad, Clarence Rosser Ogilvie, Amanullah Mohammad, James Donald Rockrohr

[2] **Flip-Flops for Accurate Multiphase Clocking: Transmission Gate Versus Current Mode Logic** Ramen Dutta et al. - IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, 2013

[3] **Transistor-Level Timing and Noise Analysis of Peripheral Logic of High Speed Memory Design** Johnie Au, Sunil Koduru, Jun Li, Sahil Bargal, Cypress/Synopsys - 2013

[4] **Static Timing Of Full Swing Differential Circuits Differential Clock and Topology Handling in NanoTime** Norb Heindl , 2014

[5] **Analysis and Characterization of Memories and Custom Digital Blocks Using NanoTime** Chad Lawrence , 2013

[6] **Accurate Transistor-Level STA Methodology for 20nm Custom SRAM Macro Using NanoTime** Shahnaz Nagle, Mandeep Singh, Charles Jiang , Samsung/ Synopsys

[7] **Transistor-level Static Timing Analysis and Characterization of Embedded SRAM Using NanoTime for Memories** Clint Parker , Charles Jiang , HP/ Synopsys