Where have all the phases gone? Using multiclock propagation in PrimeTime

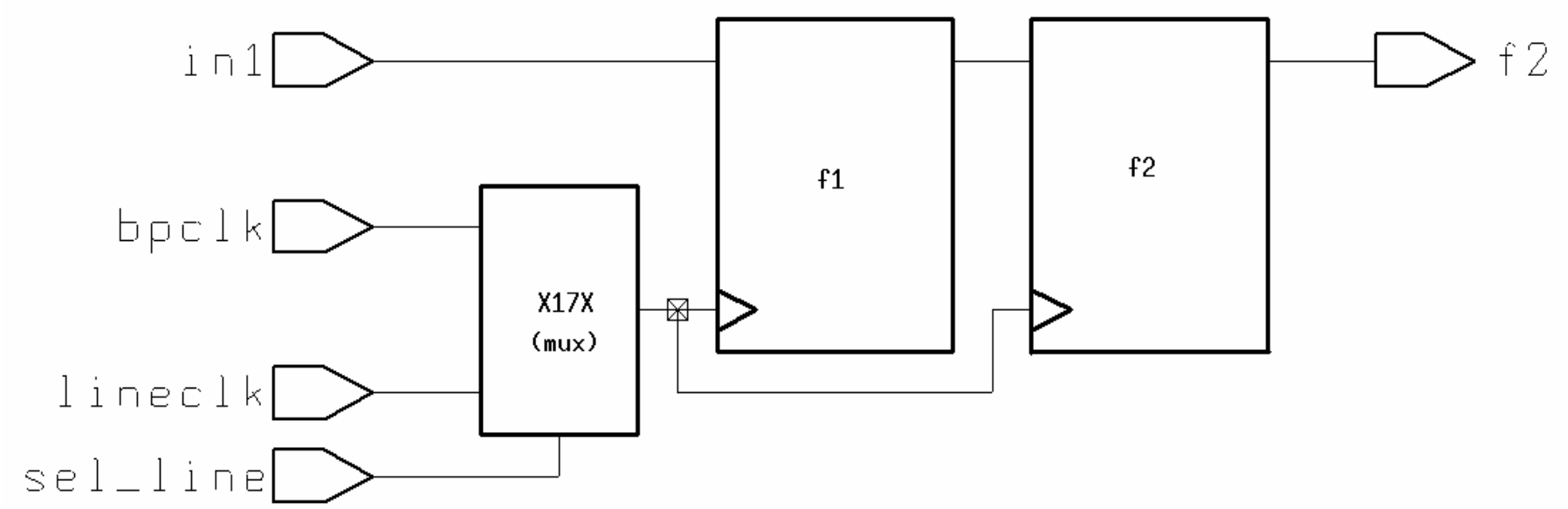
Paul Zimmer

Zimmer Design Services

(paulzimmer@zimmerdesignservices.com)

Introduction

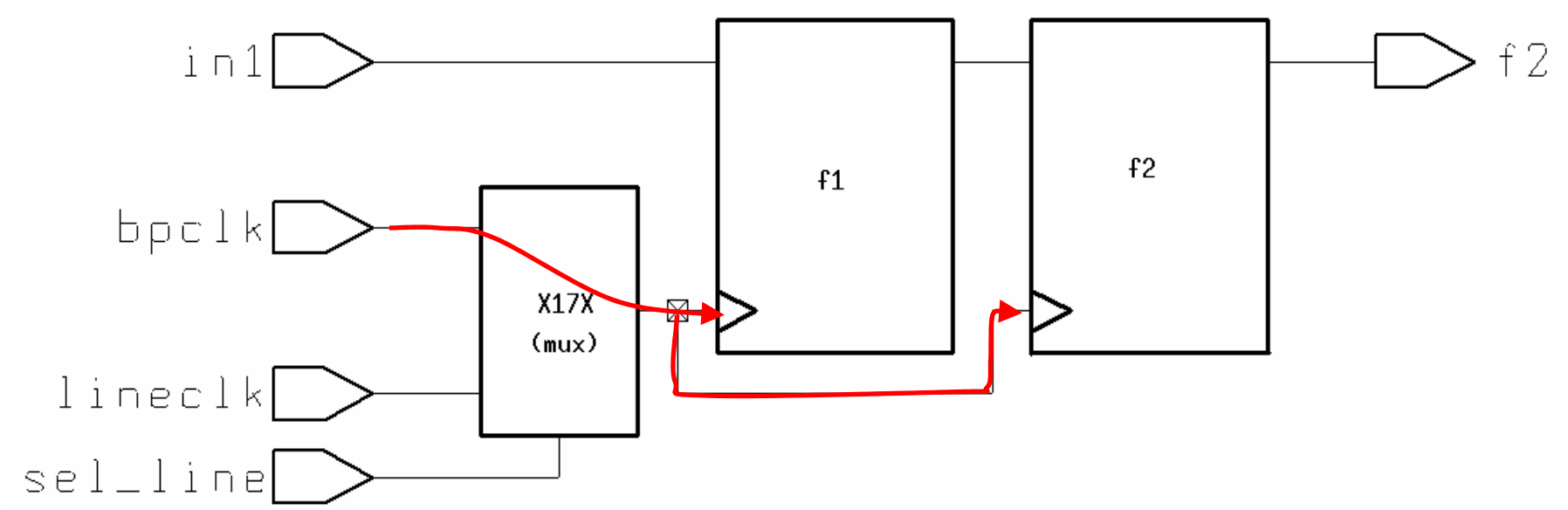
From my 2001 paper:



Introduction

From my 2001 paper:

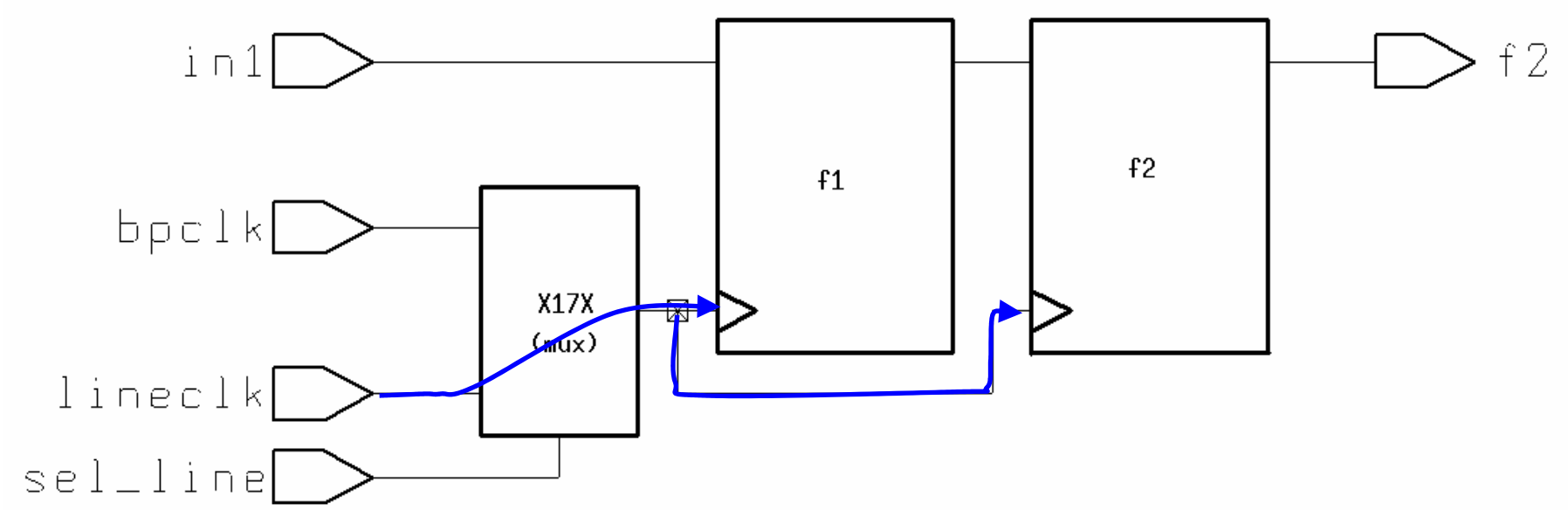
This:



Introduction

From my 2001 paper:

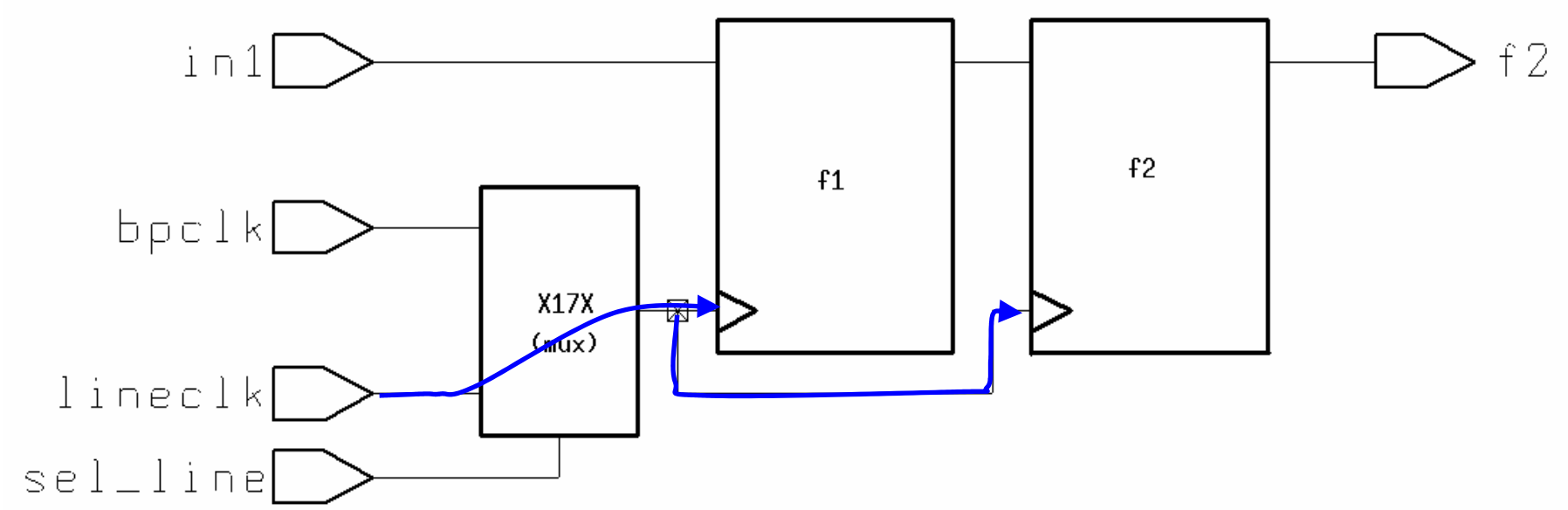
Or this:



Introduction

From my 2001 paper:

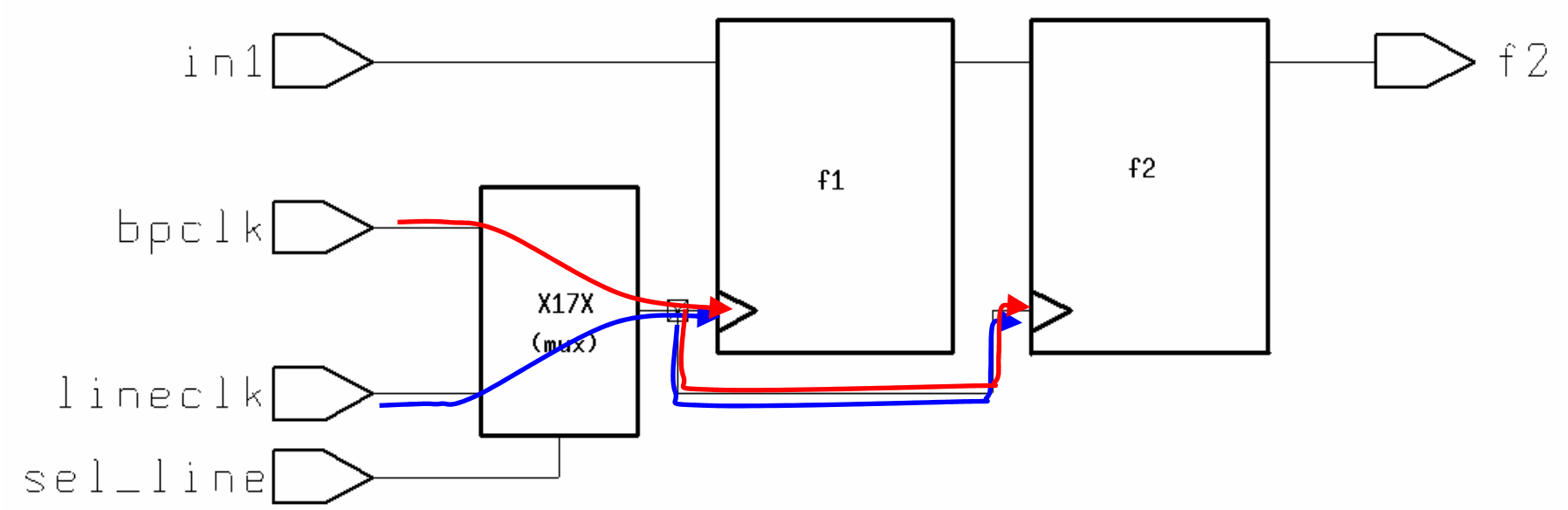
Or this:



BUT NOT BOTH!

Introduction

With multiclock propagation:



BOTH!

Introduction

Why use multiclock propagation?

- Reduces number of modes in “modes x corners”
- Accurate noise analysis. All clocks present.
- Ultimately, will be needed by physical tools.
- And - set_case_analysis doesn’t always propagate.

Introduction

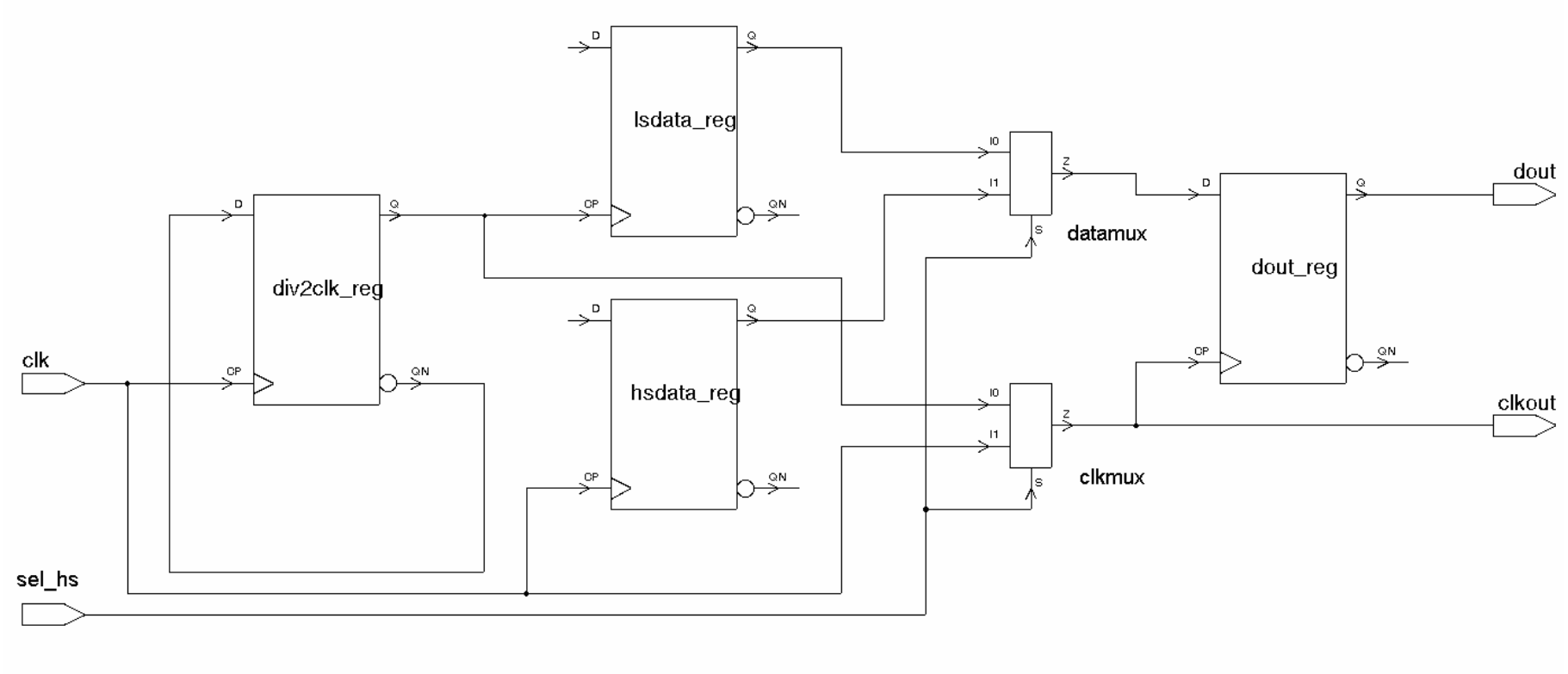
Going to introduce some new(er) features in PT:

1. Clock killer (set_clock_sense -stop_propagation)
2. clocks attribute
3. create_generated_clock –combinational
4. The final nail in the coffin of promiscuous clocks.

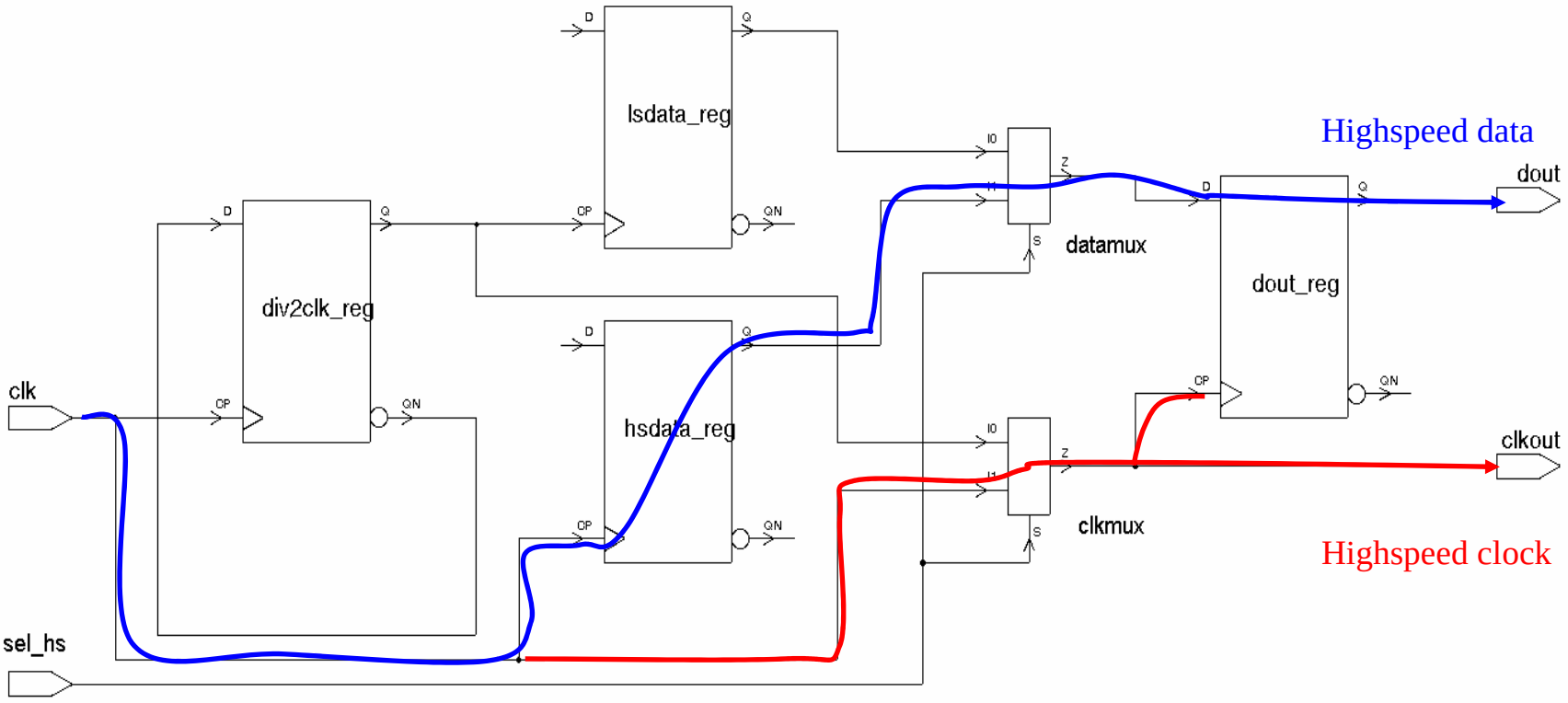
Examples

1. Shared output circuit
2. Shared input circuit
3. Programmable output circuit
4. Conclusion

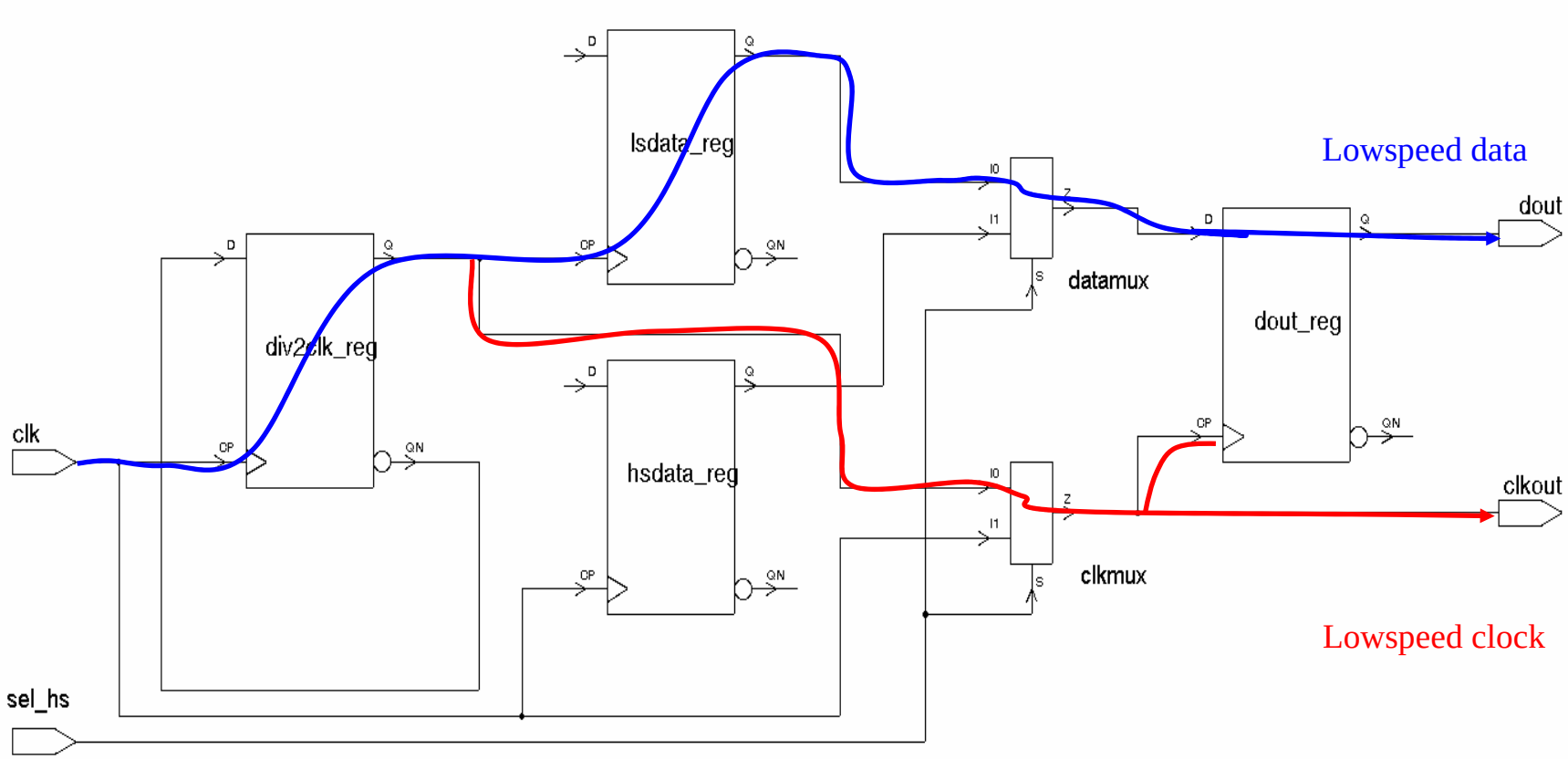
Shared outputs



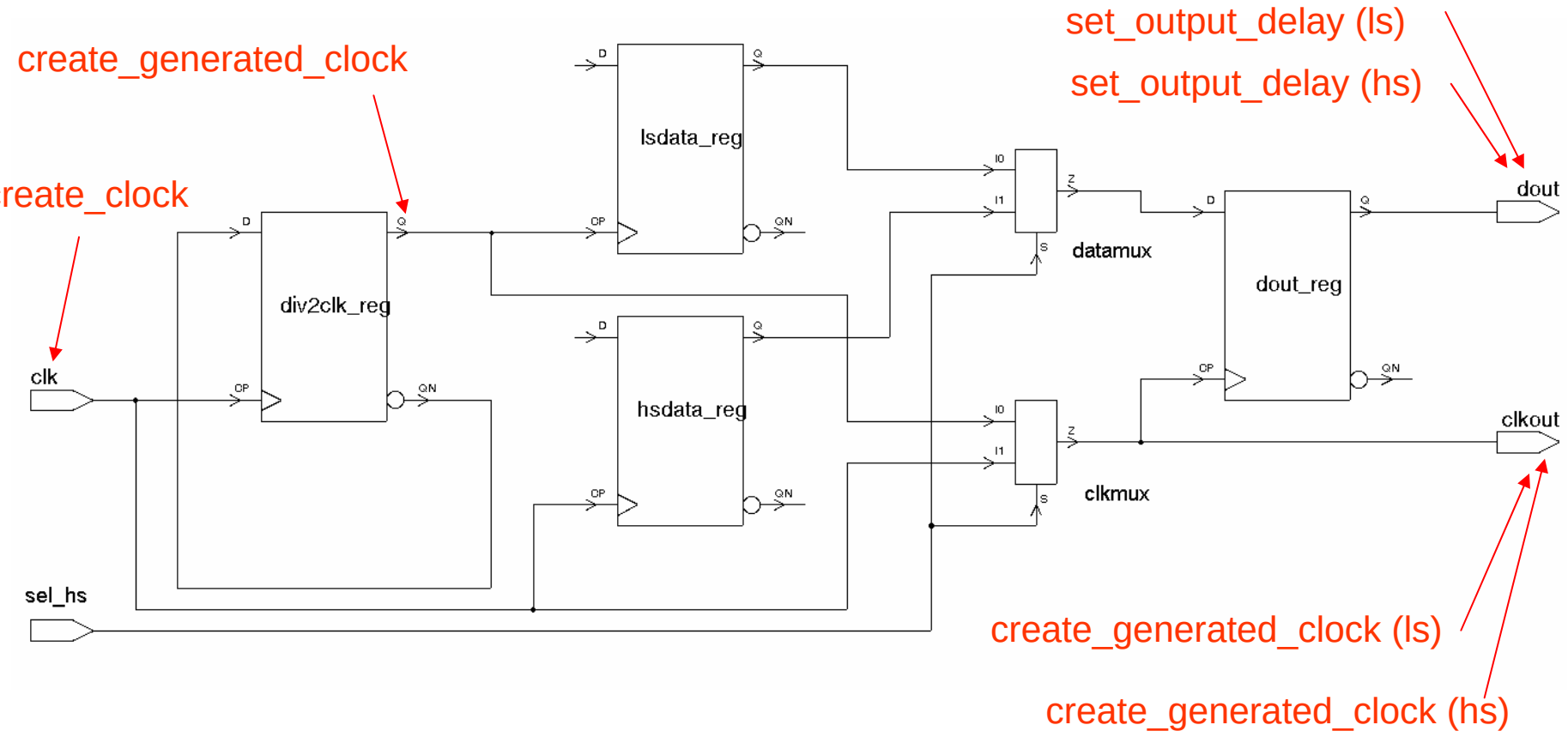
Shared outputs



Shared outputs



Shared outputs



Shared outputs

Create the input and divide-by 2 clocks:

```
set hsperiod 1.0

create_clock -period $hsperiod [get_ports clk] -name hsclk

create_generated_clock \
  -name lsclk \
  -source [get_attribute [get_clocks hsclk] sources] \
  -divide_by 2 \
  -master_clock hsclk \
  -add \
  [get_pins div2clk_reg/Q]
```

Shared outputs

Create the output clocks (src-sync i/f):

```
create_generated_clock  \
  -name hsclkout \
  -source [get_attribute [get_clocks hsclk] sources] \
  -divide_by 1 \
  -master_clock hsclk \
  -add \
  [get_ports clkout]

create_generated_clock  \
  -name lsclkout \
  -source [get_attribute [get_clocks lsclk] sources] \
  -divide_by 1 \
  -master_clock lsclk \
  -add \
  [get_ports clkout]
```

Shared outputs

Set output delays:

```
set_output_delay [expr -1 * 0.1] -clock hsclkout -min [get_ports dout]
set_output_delay 0.5 -clock hsclkout -max [get_ports dout] -add_delay
set_output_delay [expr -1 * 0.2] -clock lsclkout -min [get_ports dout]
-add_delay
set_output_delay 1.5 -clock lsclkout -max [get_ports dout] -add_delay
```

lsclk times against hsclk

```
report_timing -to dout

  Startpoint: dout_reg (rising edge-triggered flip-flop clocked by lsclk)
  Endpoint: dout (output port clocked by hsclkout)
  Path Group: hsclkout
  Path Type: max

  Point                                     Incr       Path
  ------------------------------------------------------------
  clock lsclk (rise edge)                   0.00       0.00
  clock network delay (propagated)          0.50       0.50
  dout_reg/CP (dfnrb1)                      0.00       0.50 r
  dout_reg/Q (dfnrb1)                       0.32       0.82 f
  dout (out)                                0.00       0.82 f
  data arrival time                                    0.82

  clock hsclkout (rise edge)                1.00       1.00
  clock network delay (propagated)          0.16       1.16
  output external delay                    -0.50       0.66
  data required time                                   0.66
  ------------------------------------------------------------
  data required time                                   0.66
  data arrival time                                   -0.82
  ------------------------------------------------------------
  slack (VIOLATED)                                    -0.16
```

Creating the clock groups

Create clock groups:

```
set_clock_groups -name muxed_out -logically_exclusive \
  -group [get_clocks "hs*"] \
  -group [get_clocks "ls*"]
```

Problem solved

```
report_timing -to dout

  Startpoint: dout_reg (rising edge-triggered flip-flop clocked by hsclk)
  Endpoint: dout (output port clocked by hsclkout)
  Path Group: hsclkout
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock hsclk (rise edge)                  0.00       0.00
  clock network delay (propagated)         0.16       0.16
  dout_reg/CP (dfnrb1)                     0.00       0.16 r
  dout_reg/Q (dfnrb1)                      0.32       0.48 f
  dout (out)                               0.00       0.48 f
  data arrival time                                   0.48

  clock hsclkout (rise edge)               1.00       1.00
  clock network delay (propagated)         0.16       1.16
  output external delay                   -0.50       0.66
  data required time                                  0.66
  -----------------------------------------------------------
  data required time                                  0.66
  data arrival time                                  -0.48
  -----------------------------------------------------------
  slack (MET)                                         0.18
```

Why is the insertion delay so different?

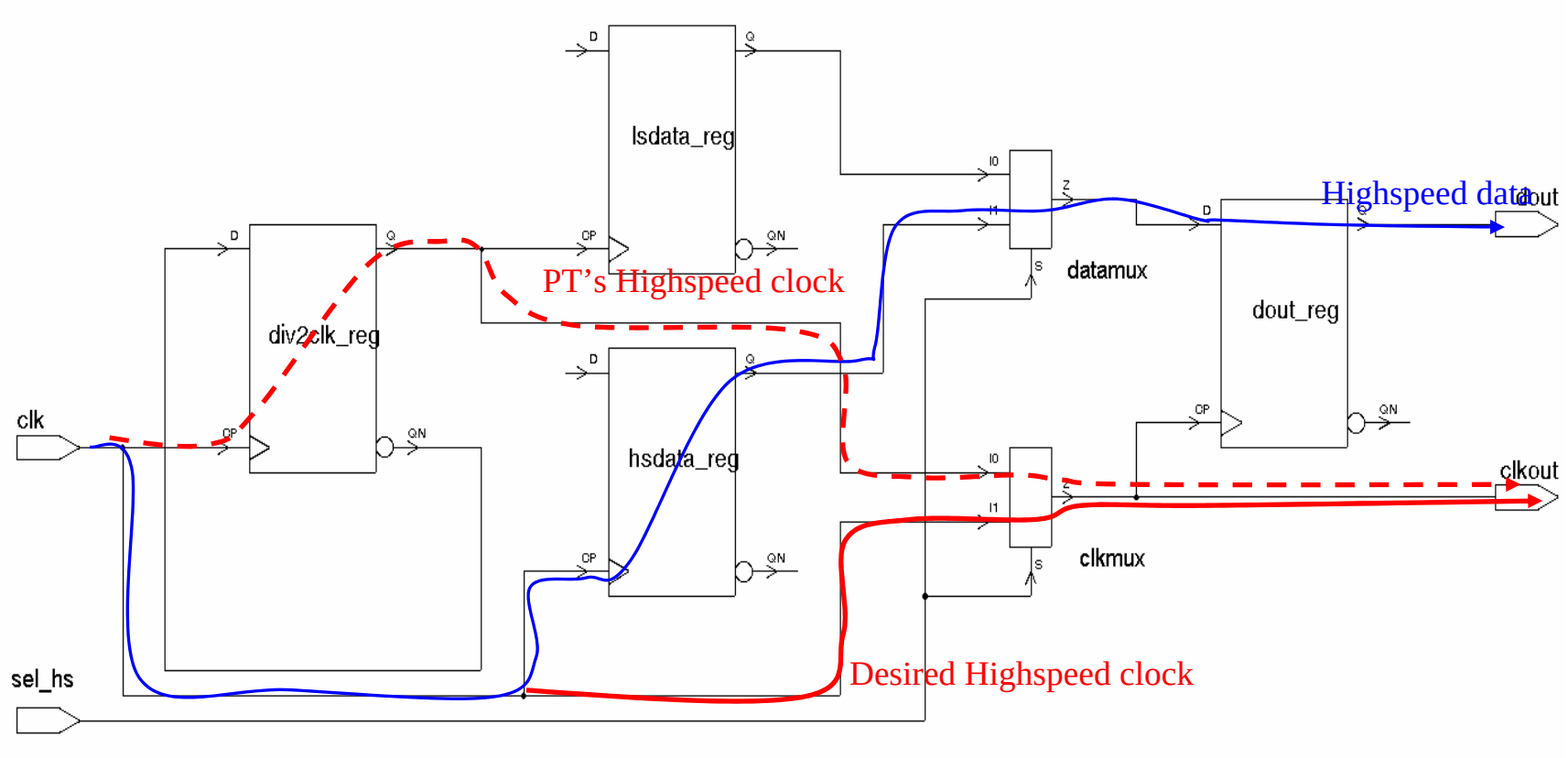
```
report_timing -to dout -delay min -group hsclkout

  Startpoint: dout_reg (rising edge-triggered flip-flop clocked by hsclk)
  Endpoint: dout (output port clocked by hsclkout)
  Path Group: hsclkout
  Path Type: min

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock hsclk (rise edge)                  0.00       0.00
  clock network delay (propagated)         0.16       0.16
  dout_reg/CP (dfnrb1)                     0.00       0.16 r
  dout_reg/Q (dfnrb1)                      0.32       0.48 r
  dout (out)                               0.00       0.48 r
  data arrival time                                   0.48

  clock hsclkout (rise edge)               0.00       0.00
  clock network delay (propagated)         0.50       0.50
  output external delay                    0.10       0.60
  data required time                                  0.60
  -----------------------------------------------------------
  data required time                                  0.60
  data arrival time                                  -0.48
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.12
```

Why does hsclkout go through the divider??

```
report_timing -to dout -delay min -path full_clock_expanded -input -group hsclkout
  Point                                              Incr       Path
  -----------------------------------------------------------------
  clock hsclk (rise edge)                          0.00       0.00
  clock source latency                             0.00       0.00
  clk (in)                                         0.00       0.00 r
  clkmux/I1 (mx02d0)                               0.00       0.00 r
  clkmux/Z (mx02d0)                                0.16       0.16 r
  dout_reg/CP (dfnrb1)                             0.00       0.16 r
  dout_reg/Q (dfnrb1)                              0.32       0.48 r
  dout (out)                                       0.00       0.48 r
  data arrival time                                           0.48

  clock hsclkout (rise edge)                       0.00       0.00
  clock hsclk (source latency)                     0.00       0.00
  clk (in)                                         0.00       0.00 r
  div2clk_reg/CP (dfnrb1)                          0.00       0.00 r
  div2clk_reg/Q (dfnrb1)                           0.32       0.32 r
  clkmux/I0 (mx02d0)                               0.00       0.32 r
  clkmux/Z (mx02d0)                                0.18       0.50 r
  clkout (out)                                     0.00       0.50 r
  output external delay                            0.10       0.60
  data required time                                          0.60
  -----------------------------------------------------------------
  data required time                                          0.60
  data arrival time                                          -0.48
  -----------------------------------------------------------------
  slack (VIOLATED)                                           -0.12
```

div2clk_reg should not be in ***hs*** path!

Promiscuous clock problem revisited



Shared outputs

This is the “promiscuous clock problem” I discussed in last year’s paper.

But there’s a new solution!

New fix for promiscuous clocks

Use new –combinational switch:

```
create_generated_clock  \
  -name hsclkout \
  -source [get_attribute [get_clocks hsclk] sources] \
  -comb \
  -master_clock hsclk \
  -add \
  [get_ports clkout]

create_generated_clock  \
  -name lsclkout \
  -source [get_attribute [get_clocks lsclk] sources] \
  -comb \
  -master_clock lsclk \
  -add \
  [get_ports clkout]
```

New fix for promiscuous clocks

Use new –combinational switch:

```
create_generated_clock  \
   -name hsclkout \
   -source [get_attribute [get_clocks hsclk] sources] \
   -comb \
   -master_clock hsclk \
   -add \
   [get_ports clkout]

create_generated_clock  \
   -name lsclkout \
   -source [get_attribute [get_clocks lsclk] sources] \
   -comb \
   -master_clock lsclk \
   -add \
   [get_ports clkout]
```

Shared outputs

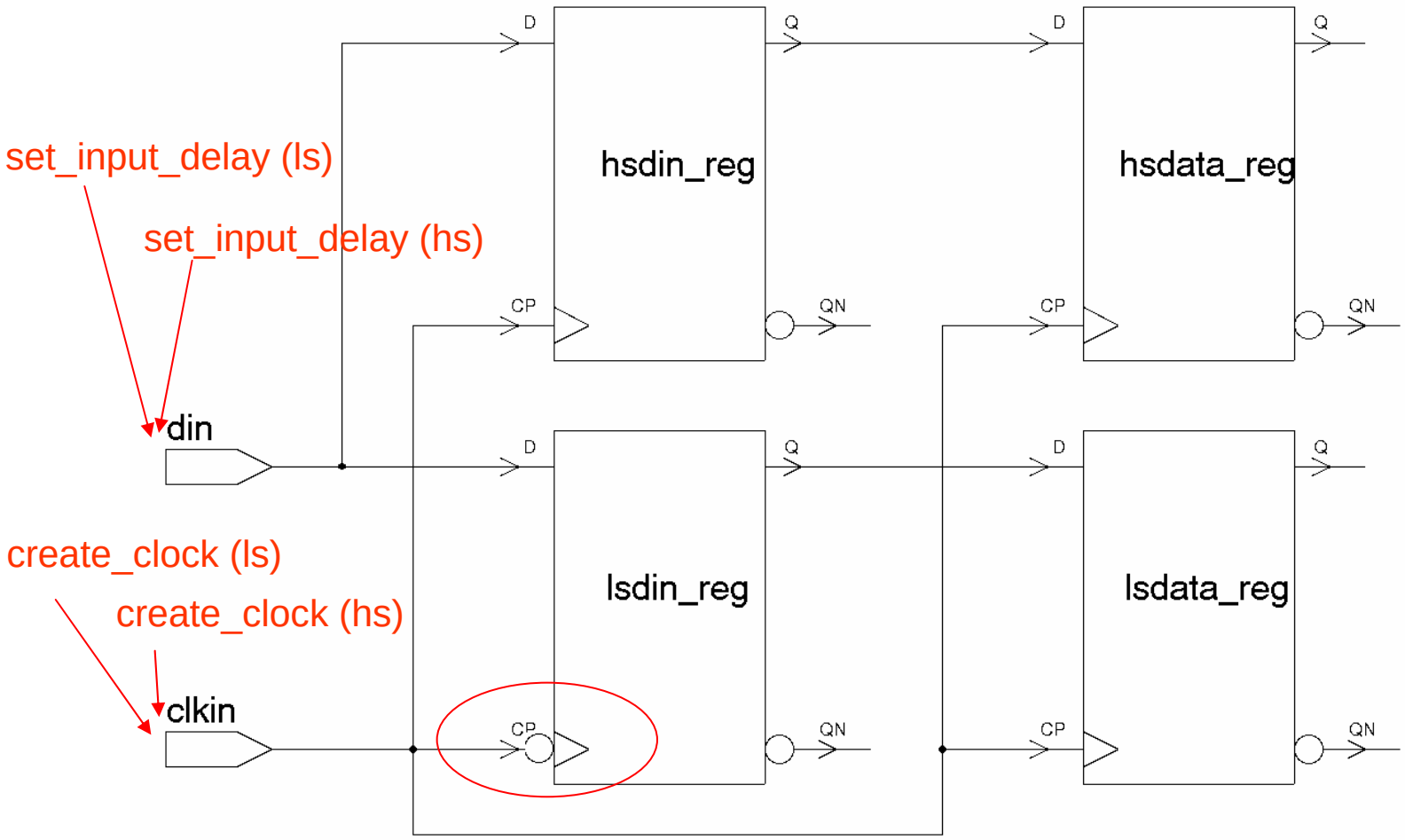
```
report_timing -to dout -delay min -path full_clock_expanded -input -group hsclkout

  Point                                          Incr       Path
  ---------------------------------------------------------------
  clock hsclk (rise edge)                        0.00       0.00
  clock source latency                           0.00       0.00
  clk (in)                                       0.00       0.00 r
  clkmux/I1 (mx02d0)                             0.00       0.00 r
  clkmux/Z (mx02d0)                              0.16       0.16 r
  dout_reg/CP (dfnrb1)                           0.00       0.16 r
  dout_reg/Q (dfnrb1)                            0.32       0.48 r
  dout (out)                                     0.00       0.48 r
  data arrival time                                         0.48

  clock hsclkout (rise edge)                     0.00       0.00
  clock hsclk (source latency)                   0.00       0.00
  clk (in)                                       0.00       0.00 r
  clkmux/I1 (mx02d0)                             0.00       0.00 r
  clkmux/Z (mx02d0)                              0.16       0.16 r
  clkout (out)                                   0.00       0.16 r
  output external delay                          0.10       0.26
  data required time                                        0.26
  ---------------------------------------------------------------
  data required time                                        0.26
  data arrival time                                        -0.48
  ---------------------------------------------------------------
  slack (MET)                                               0.22
```

div2clk_reg bypassed

Examples

1. Shared output circuit
2. Shared input circuit
3. Programmable output circuit
4. Conclusion

Shared inputs



Shared inputs

Create clocks, input delays, clock groups:

```
set hsperiod 1.0
set lsperiod 10.0
create_clock -period $hsperiod [get_ports clkin] -name hsclkin
create_clock -period $lsperiod [get_ports clkin] -name lsclkin -add

set_input_delay 0.1 -clock hsclkin -min [get_ports din]
set_input_delay 0.5 -clock hsclkin -max [get_ports din] -add_delay
set_input_delay 0.7 -clock lsclkin -min [get_ports din] -add_delay
set_input_delay 2.5 -clock lsclkin -max [get_ports din] -add_delay

set_clock_groups -name muxed_in -physically_exclusive \
  -group [get_clocks "hs*"] \
  -group [get_clocks "ls*"]
```

Shared inputs

Create clocks, input delays, clock groups:

```
set hsperiod 1.0
set lsperiod 10.0
create_clock -period $hsperiod [get_ports clkin] -name hsclkin
create_clock -period $lsperiod [get_ports clkin] -name lsclkin -add

set_input_delay 0.1 -clock hsclkin -min [get_ports din]
set_input_delay 0.5 -clock hsclkin -max [get_ports din] -add_delay
set_input_delay 0.7 -clock lsclkin -min [get_ports din] -add_delay
set_input_delay 2.5 -clock lsclkin -max [get_ports din] -add_delay

set_clock_groups -name muxed_in -physically_exclusive \
  -group [get_clocks "hs*"] \
  -group [get_clocks "ls*"]
```

Shared inputs

This can get confusing, so here is a table that describes the settings and their implications for both noise and path timing:

Switch	Timing Paths	Noise Analysis	Timing Windows
-asynchronous	No	Yes	Infinite
-logically_exclusive	No	Yes	Per Waveforms
-physically_exclusive	No	No	N/A

Shared Inputs

```
report_timing -to lsdin_reg/D -group hsclkin

  Startpoint: din (input port clocked by hsclkin)
  Endpoint: lsdin_reg (falling edge-triggered flip-flop clocked by hsclkin)
  Path Group: hsclkin
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock hsclkin (rise edge)                0.00       0.00
  clock network delay (propagated)         0.00       0.00
  input external delay                     0.50       0.50 f
  din (in)                                 0.00       0.50 f
  lsdin_reg/D (dfnfb1)                     0.00       0.50 f
  data arrival time                                   0.50

  clock hsclkin (fall edge)                0.50       0.50
  clock network delay (propagated)         0.00       0.50
  lsdin_reg/CPN (dfnfb1)                              0.50 f
  library setup time                      -0.12       0.38
  data required time                                  0.38
  -----------------------------------------------------------
  data required time                                  0.38
  data arrival time                                  -0.50
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.12
```

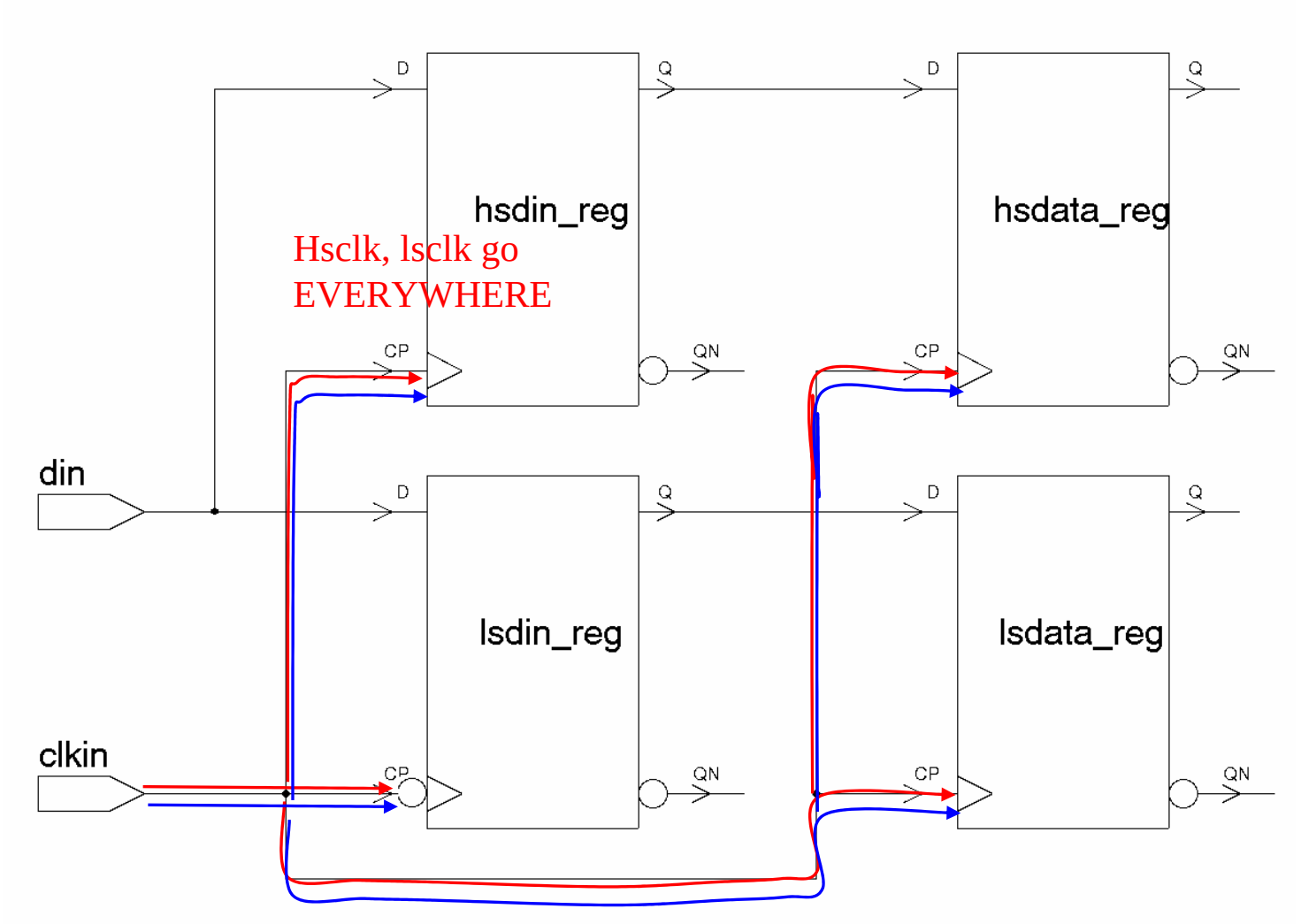
lsdin_reg being checked on hs clock

Shared Inputs

```
report_timing -to lsdata_reg/D -group hsclkin

  Startpoint: lsdin_reg (falling edge-triggered flip-flop clocked by hsclkin)
  Endpoint: lsdata_reg (rising edge-triggered flip-flop clocked by hsclkin)
  Path Group: hsclkin
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock hsclkin (fall edge)                0.50       0.50
  clock network delay (propagated)         0.00       0.50
  lsdin_reg/CPN (dfnfb1)                   0.00       0.50 f
  lsdin_reg/Q (dfnfb1)                     0.32       0.82 r
  lsdata_reg/D (dfnrq1)                    0.00       0.82 r
  data arrival time                                   0.82

  clock hsclkin (rise edge)                1.00       1.00
  clock network delay (propagated)         0.00       1.00
  lsdata_reg/CP (dfnrq1)                              1.00 r
  library setup time                      -0.10       0.90
  data required time                                  0.90
  -----------------------------------------------------------
  data required time                                  0.90
  data arrival time                                  -0.82
  -----------------------------------------------------------
  slack (MET)                                         0.08
```

Worse, its internal paths being checked against hs period

Shared inputs

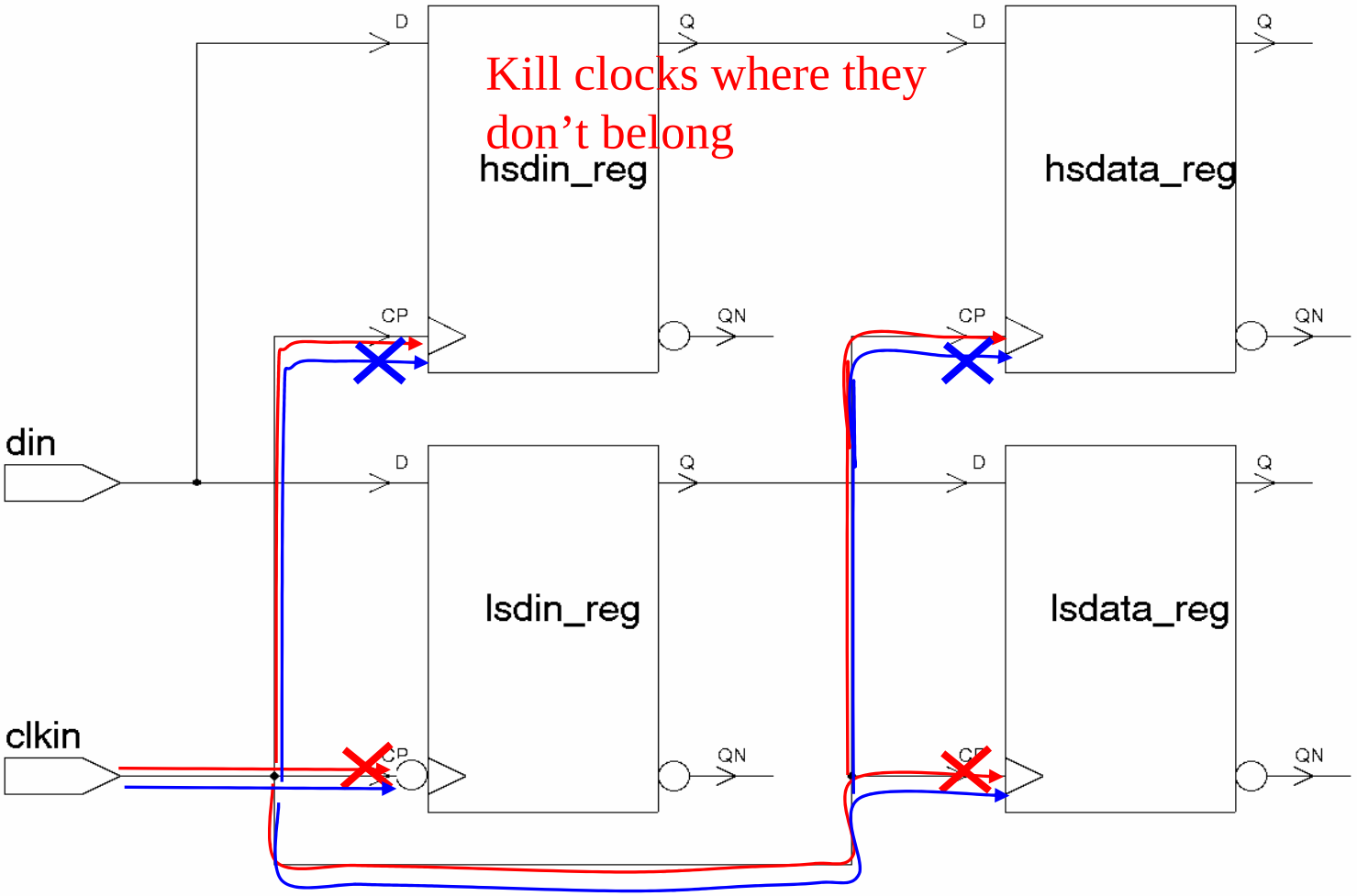


Shared inputs

New feature allows us to kill the clock:

```
set_clock_sense -stop_propagation -clock hsclkin [get_pins
"ls*_reg/CP*"]
set_clock_sense -stop_propagation -clock lsclkin [get_pins
"hs*_reg/CP*"]
```

Shared inputs



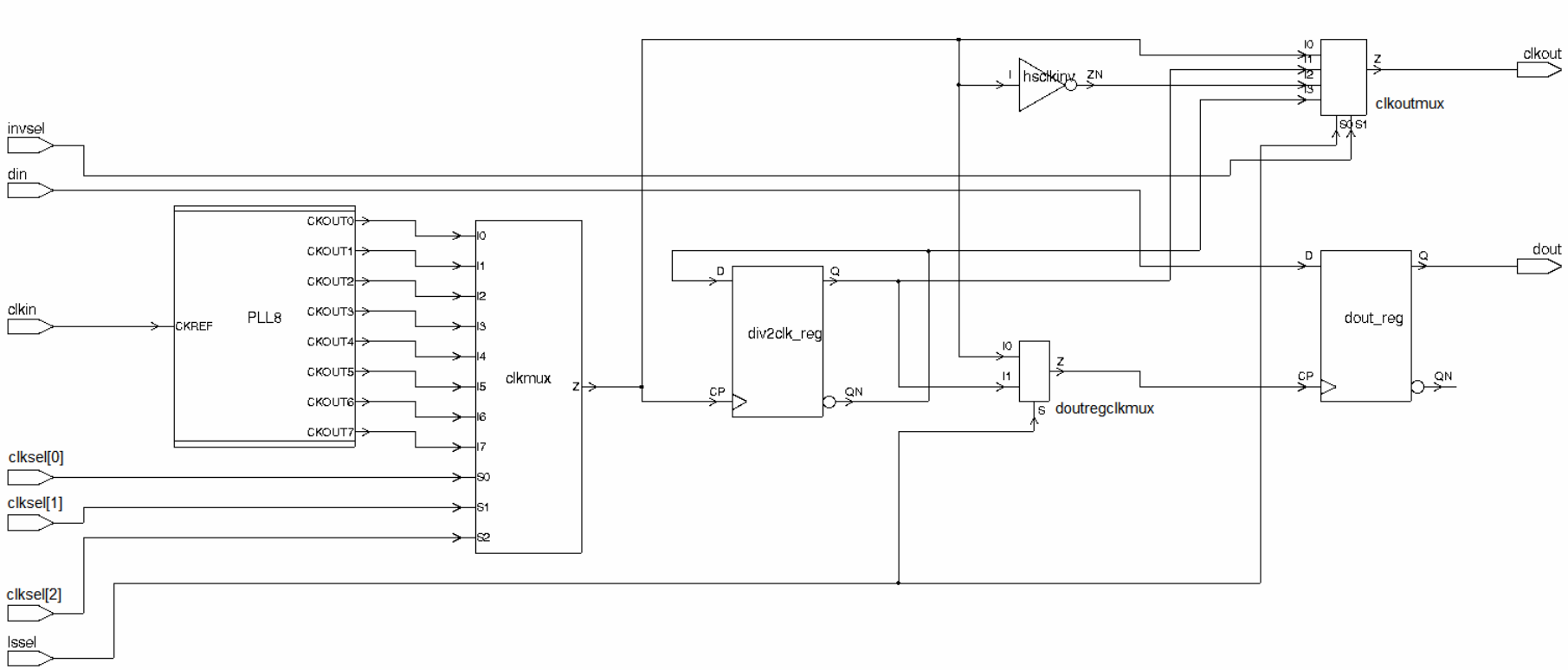
“Bad” paths gone!

```
report_timing -to lsdin_reg/D -group hsclkin

No constrained paths.

1
report_timing -to lsdata_reg/D -group hsclkin

No constrained paths.
```

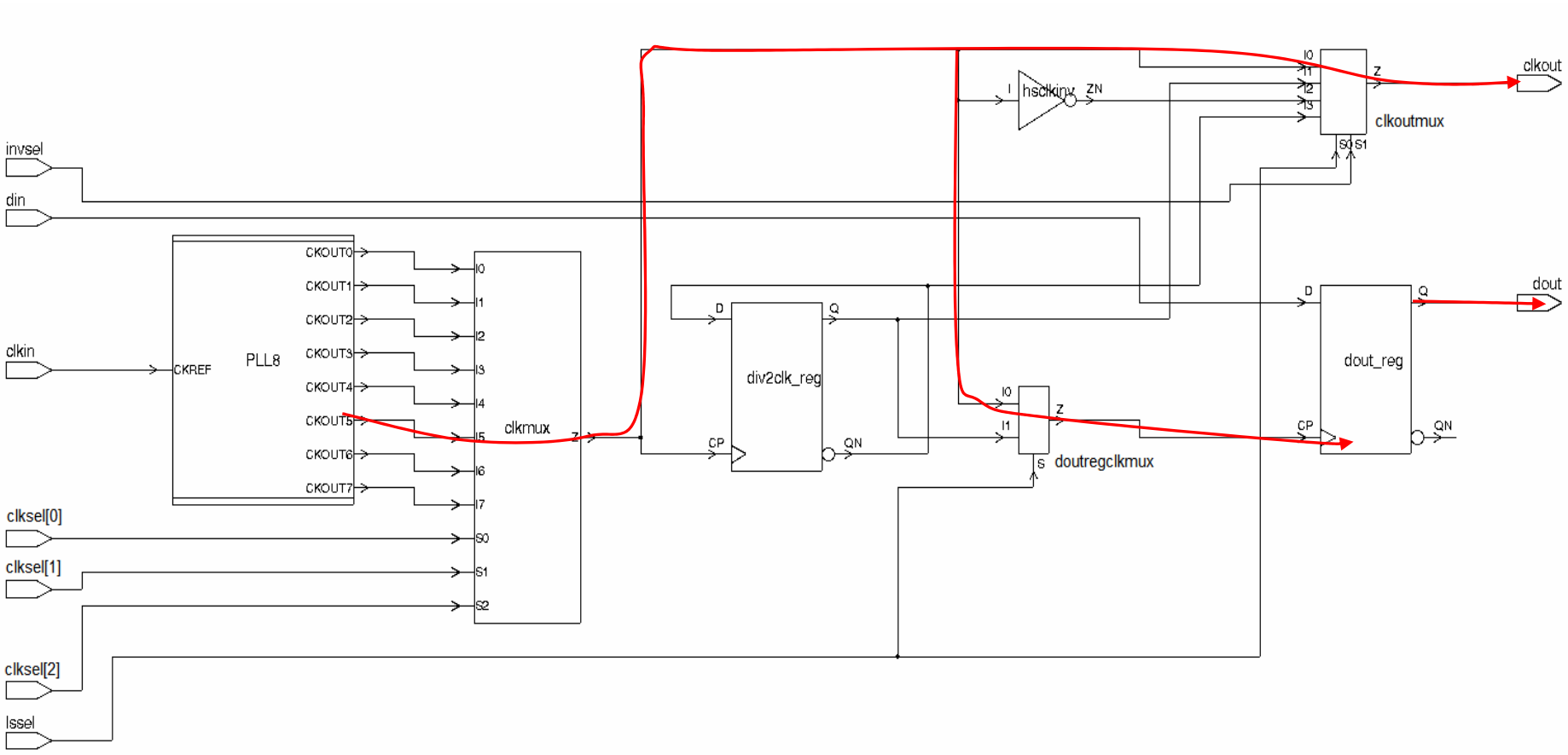
Examples

1. Shared output circuit
2. Shared input circuit
3. Programmable output circuit
4. Conclusion

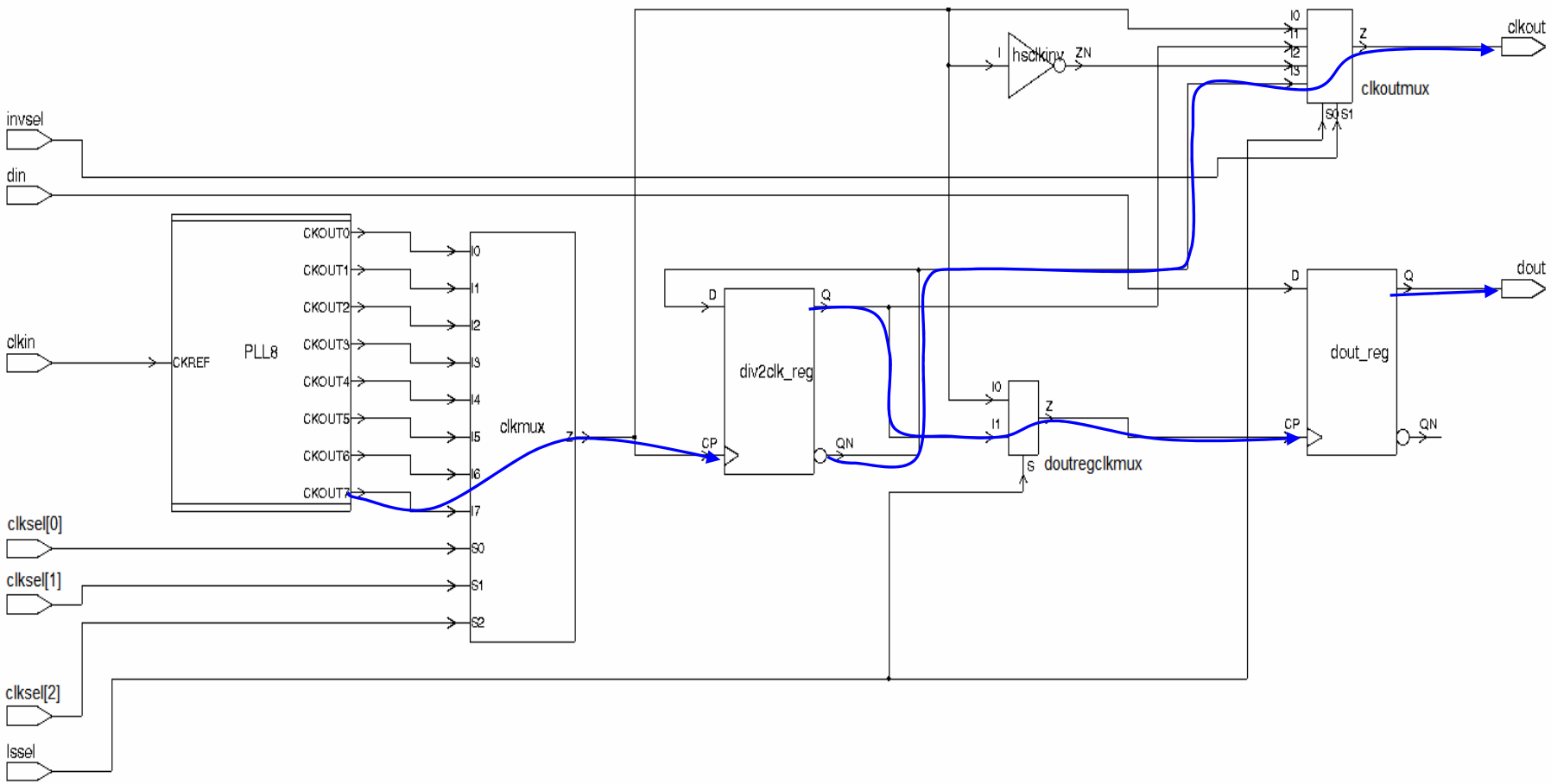
Complex Example



Highspeed, tap 5, noninvert



Lowspeed, tap 7, invert



Complex Example

Want to time multiple high and low speed modes.

There may be multiple copies of the circuit on the die, each running in different modes, but sharing the pll.

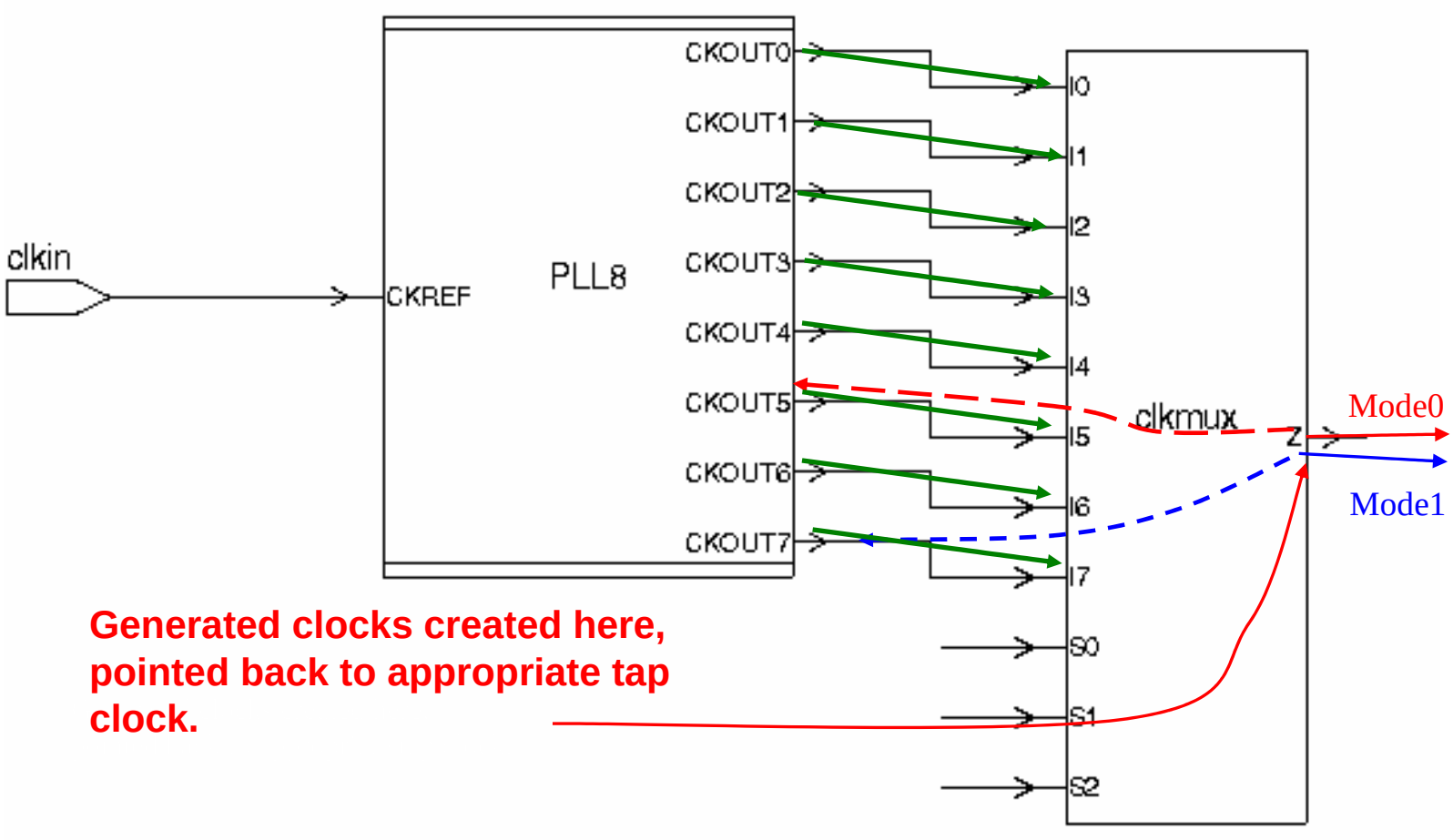
Without mcp, you'd have a huge modes/corners matrix!

Complex Example

First question: How do we handle the PLL taps?

Creating mode clocks as slaves of tap clocks

Create all tap clocks and do div1 (comb) gen'd clocks:



Timing using constraints

Create the tap clocks:

```
for {set phase 0} {$phase < 8} {incr phase} {
  set offset [expr (double($phase) / 8) *
$hsperiod]
  echo "phase: $phase ; offset: $offset"
  create_clock -name hsclk_p${phase} \
    -period $hsperiod \
    -waveform [list $offset [expr $offset +
$hsperiod / 2] ] \
    [get_pins PLL8/CKOUT${phase}]
}
```

See paper for explanation

Tap clocks report

```
****************************************
Report : clock
Design : muxed_phase
Version: Y-2006.06-SP2
Date   : Wed Oct 11 18:33:16 2006
****************************************


Attributes:
    p - Propagated clock
    G - Generated  clock
    I - Inactive   clock

Clock          Period   Waveform            Attrs     Sources
-------------------------------------------------------------------------------
hsclk_p0         4.00   {0 2}               p         {PLL8/CKOUT0}
hsclk_p1         4.00   {0.5 2.5}           p         {PLL8/CKOUT1}
hsclk_p2         4.00   {1 3}               p         {PLL8/CKOUT2}
hsclk_p3         4.00   {1.5 3.5}           p         {PLL8/CKOUT3}
hsclk_p4         4.00   {2 4}               p         {PLL8/CKOUT4}
hsclk_p5         4.00   {2.5 4.5}           p         {PLL8/CKOUT5}
hsclk_p6         4.00   {3 5}               p         {PLL8/CKOUT6}
hsclk_p7         4.00   {3.5 5.5}           p         {PLL8/CKOUT7}
```

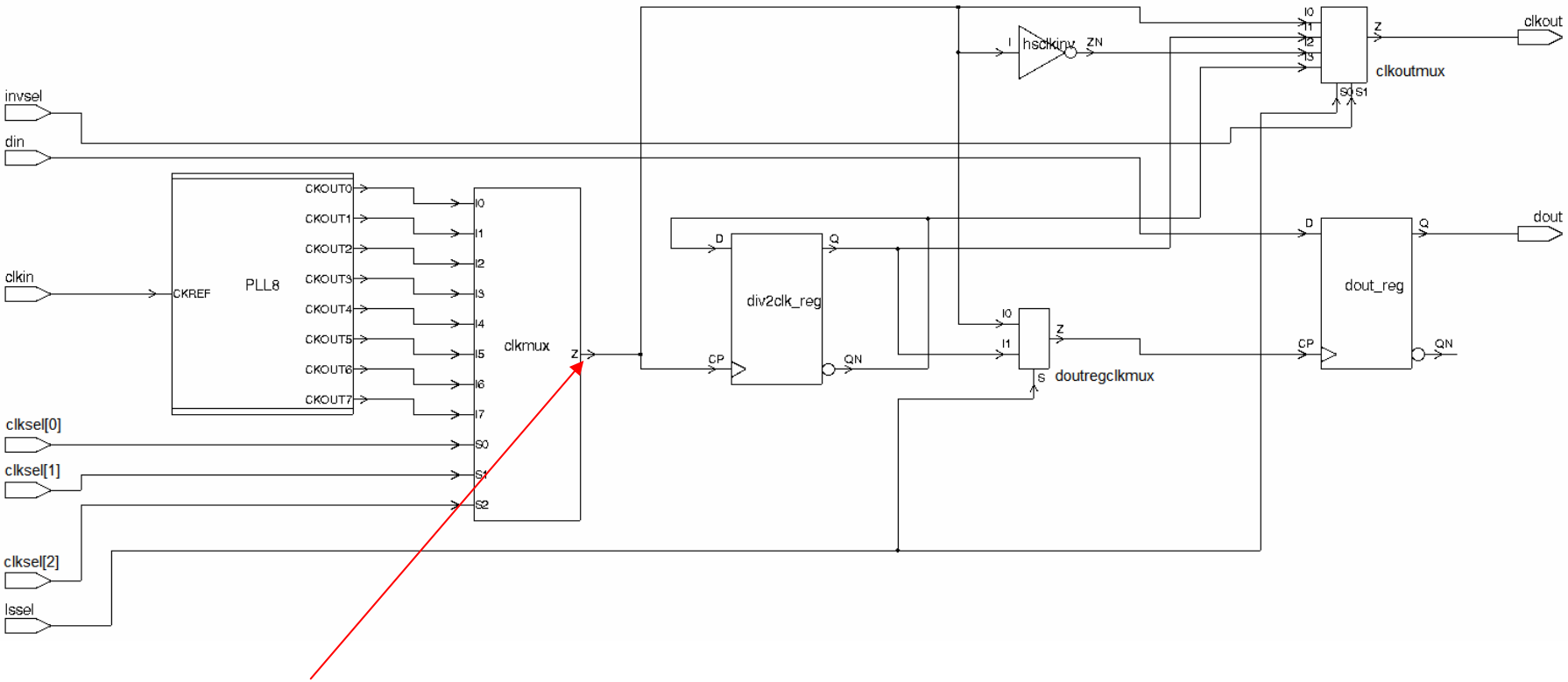
Creating mode clocks on taps

Neat new attribute in PT called “clocks” and a neat new command called “get_clock_network_objects”

Tells what clocks pass through (really, arrive at) a given pin.

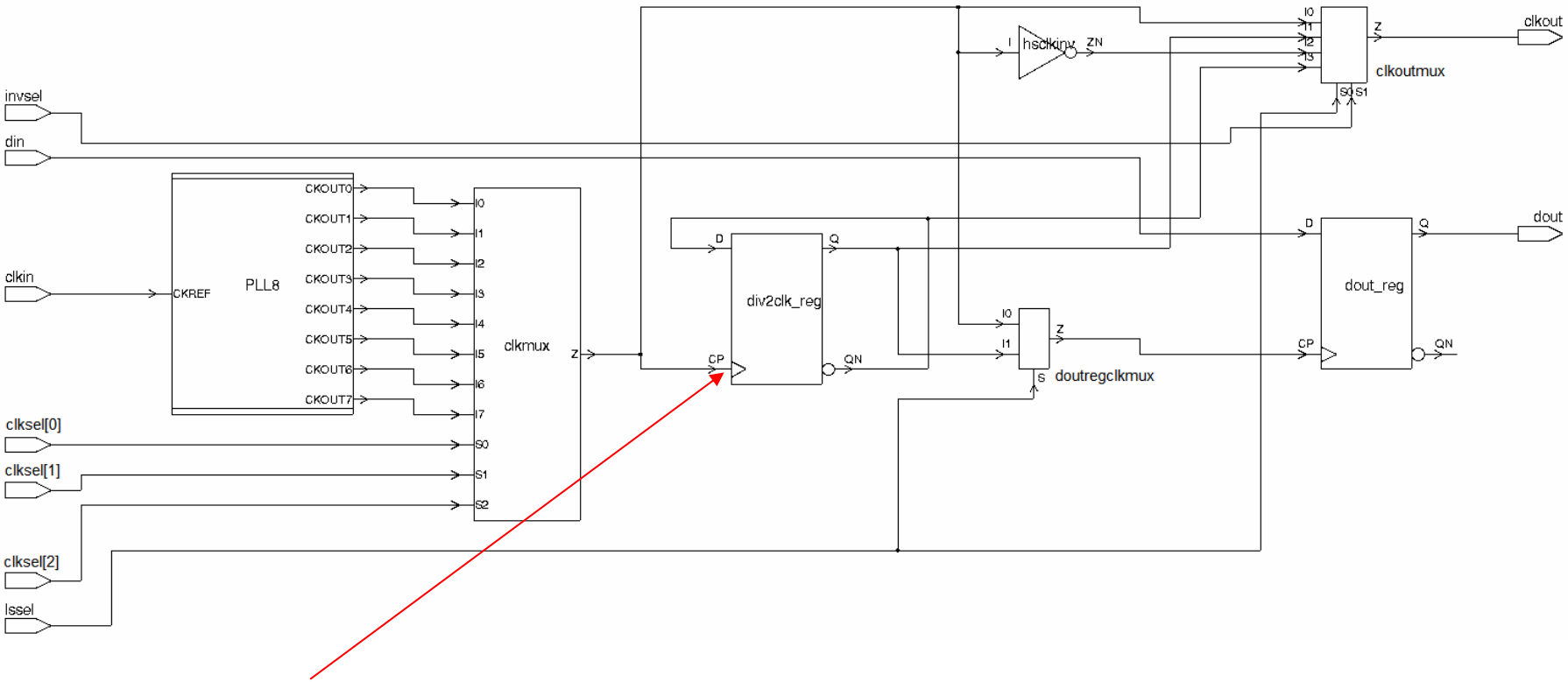
Note: For more info on the clocks attribute, and other tips on tracing clock paths, see solvnet article https://solvnet.synopsys.com/retrieve/009401.html.

“clocks” attribute



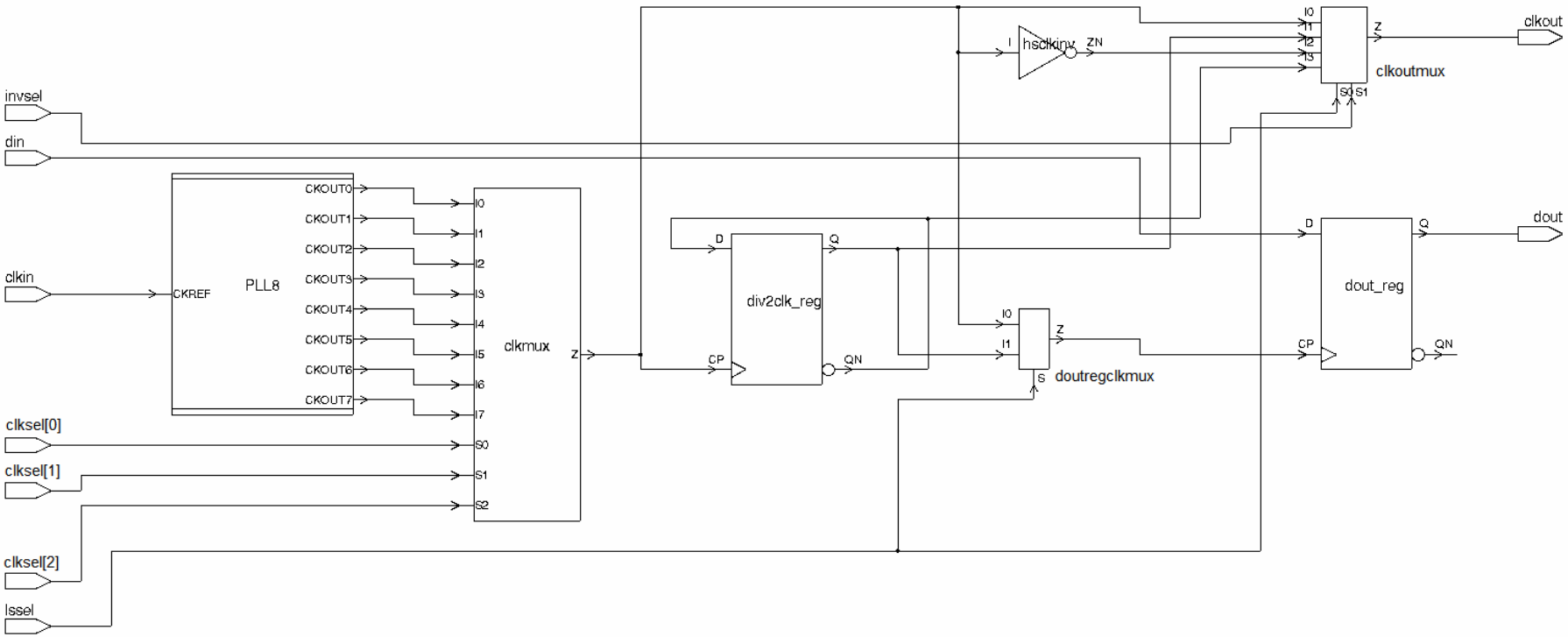

```
pt_shell> get_attribute [get_pins clkmux/Z] clocks
{"hsclk_p6", "hsclk_p7", "hsclk_p5", "hsclk_p4", "hsclk_p0", "hsclk_p3",
"hsclk_p1", "hsclk_p2"}
```

“clocks” attribute




```
pt_shell> get_attribute [get_pins div2clk_reg/CP] clocks
{"hsclk_p6", "hsclk_p7", "hsclk_p5", "hsclk_p4", "hsclk_p0", "hsclk_p3",
"hsclk_p1", "hsclk_p2"}
```

“clocks” attribute



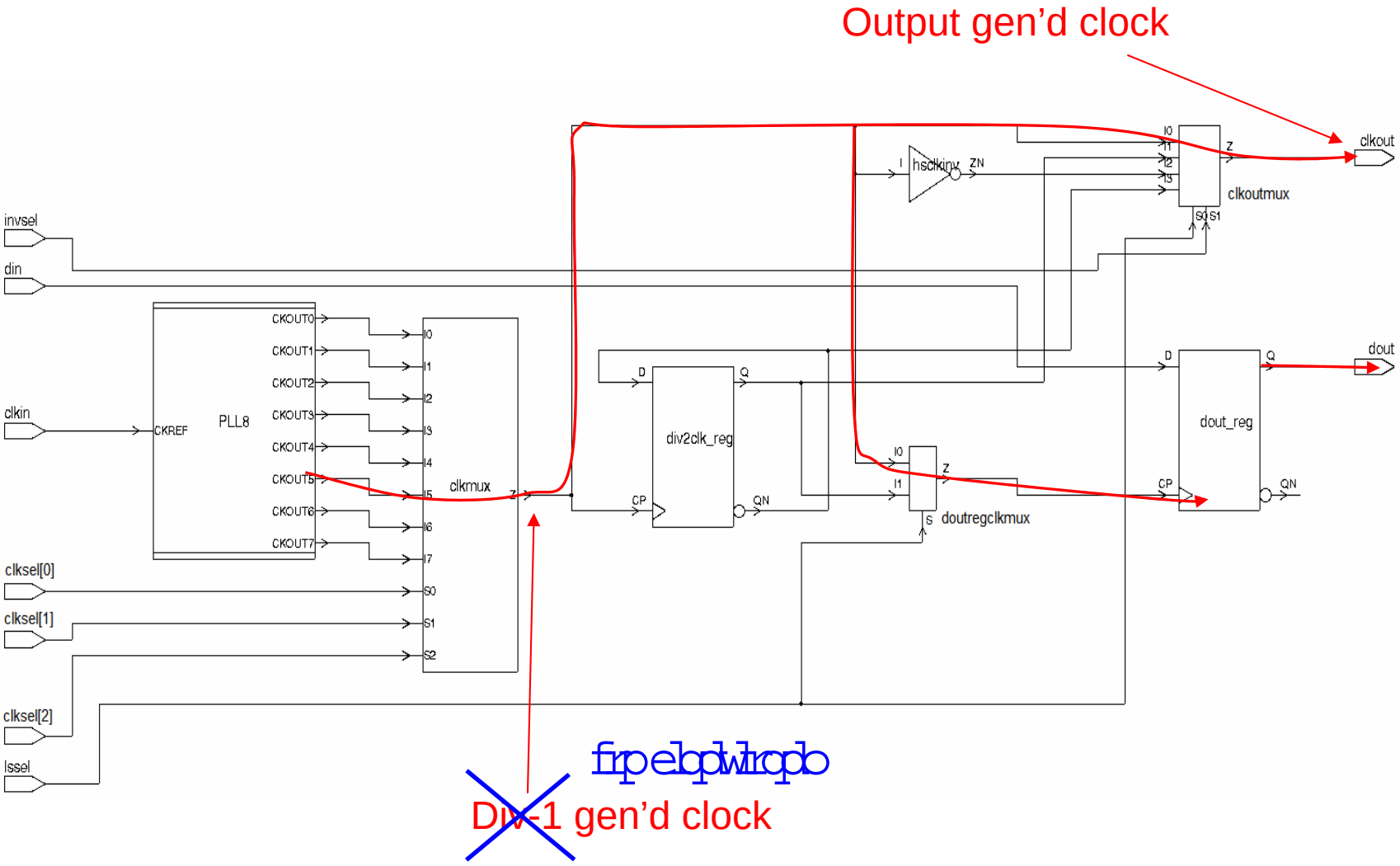

```
pt_shell> get_clock_network_objects -type pin hsclk_p0
{"clkoutmux/I0", "clkmux/Z", "dout_reg/CP", "clkoutmux/I2", "hsclkinv/ZN",
"clkoutmux/Z", "clkout", "doutregclkmux/I0", "doutregclkmux/Z", "div2clk_reg/CP",
"clkmux/I0", "hsclkinv/I"}
```

H1 mode

Define settings for H1 mode:

```
set modeH1(clksel_setting) 5
set modeH1(invsel_setting) 0
set modeH1(lssel_setting) 0
```

H1 mode


Output gen'd clock
clkout
clkoutmux
hsclkinv
invsel
din
clkin
CKREF
PLL8
CKOUT0
CKOUT1
CKOUT2
CKOUT3
CKOUT4
CKOUT5
CKOUT6
CKOUT7
clkmux
div2clk_reg
dout
dout_reg
doutregclkmux
clksel[0]
clksel[1]
clksel[2]
lssel
frpelbplwlrqpbo
Div-1 gen'd clock

H1 mode

Create main mode clock on the clkmux output:

```
create_generated_clock  \
  -name modeH1clk \
  -source [get_attribute [get_clocks hsclk_p${modeH1(clksel_setting)}]
sources] \
  -comb \
  -master_clock hsclk_p${modeH1(clksel_setting)} \
  -add \
  [get_pins clkmux/Z]
```

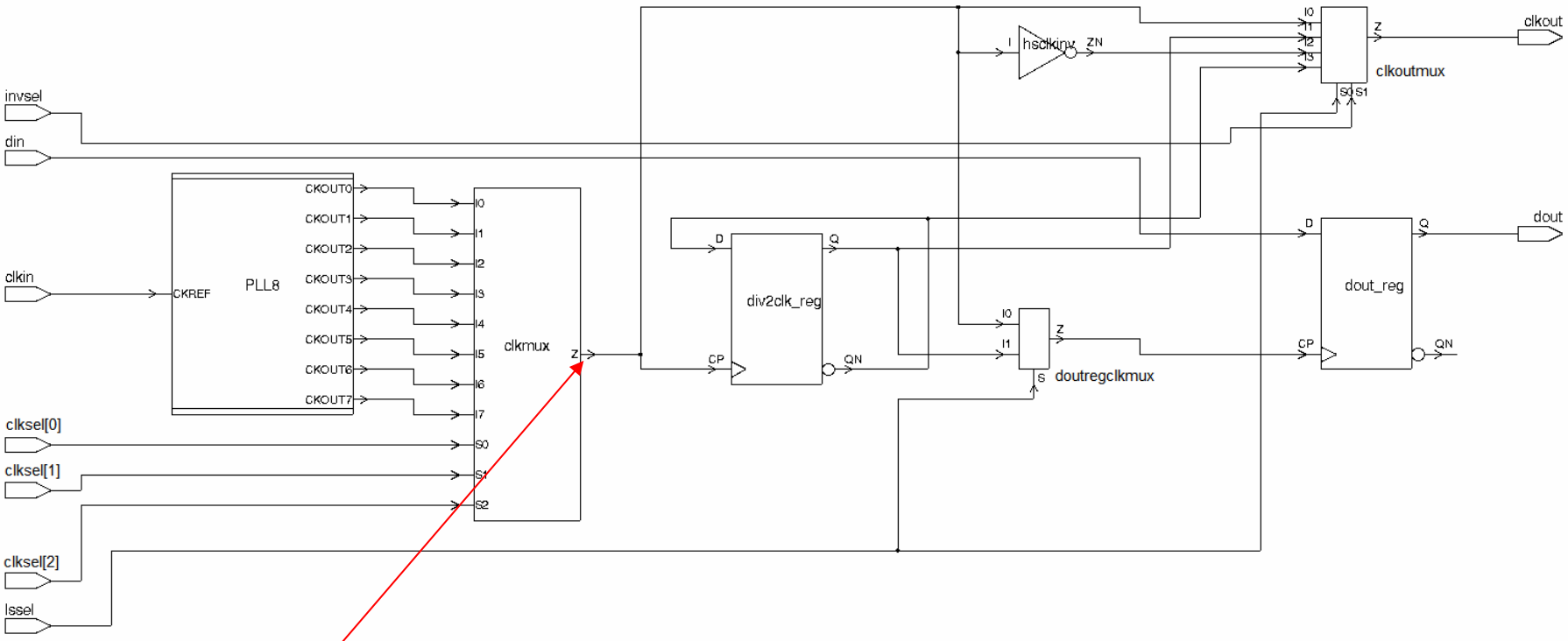
H1 mode

Create main mode clock on the clkmux output:

```
&cmd create_generated_clock  \
  -name modeH1clk \
  -source [get_attribute [get_clocks hsclk_p${modeH1(clksel_setting)}]
sources] \
  -comb \
  -master_clock hsclk_p${modeH1(clksel_setting)} \
  -add \
  [get_pins clkmux/Z]
```

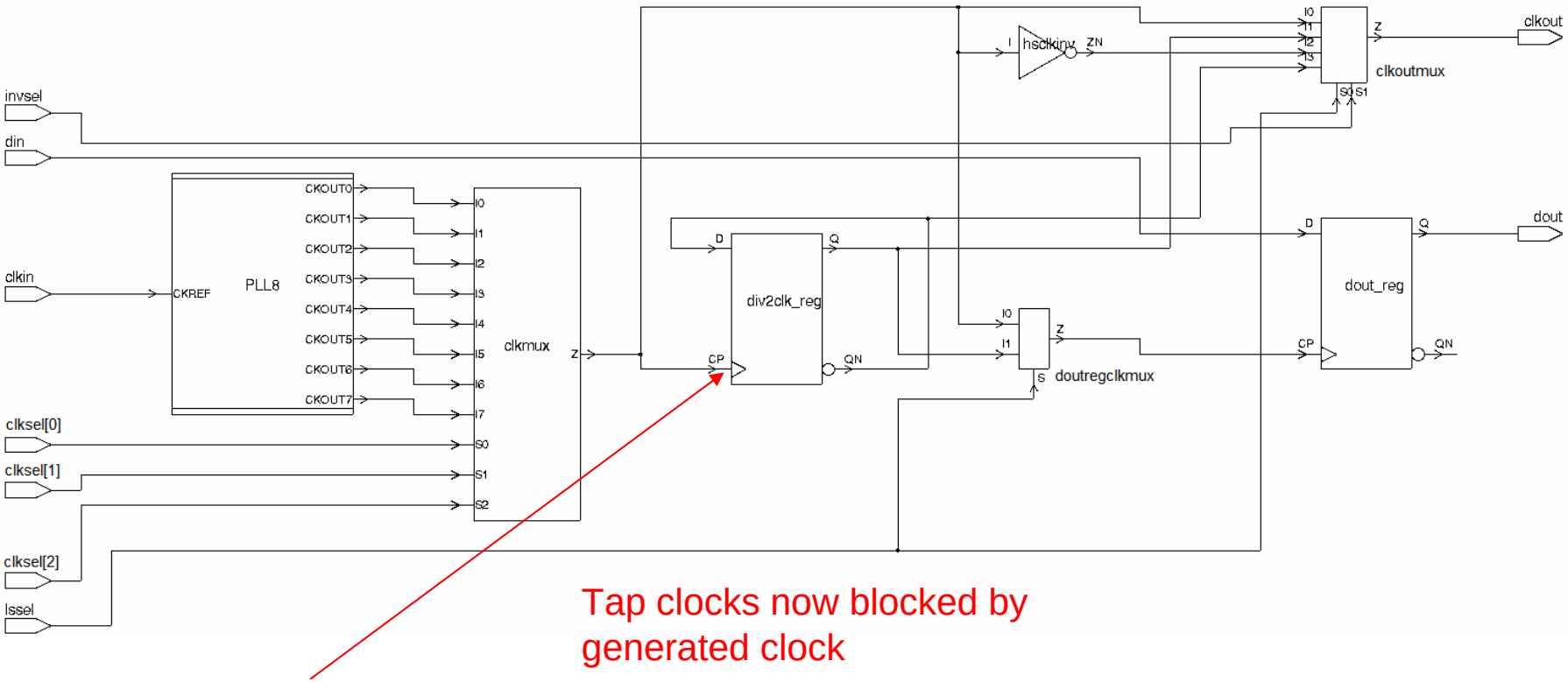
```
Doing command: create_generated_clock -name modeH1clk -source
{ PLL8/CKOUT5 } -comb -master_clock hsclk_p5 -add { clkmux/Z
}
```

“clocks” attribute revisited




```
pt_shell> get_attribute [get_pins clkmux/Z] clocks
{"hsclk_p0", "hsclk_p1", "hsclk_p2", "modeH1clk", "hsclk_p4", "hsclk_p3", "hsclk_p5",
"hsclk_p6", "hsclk_p7"}
```

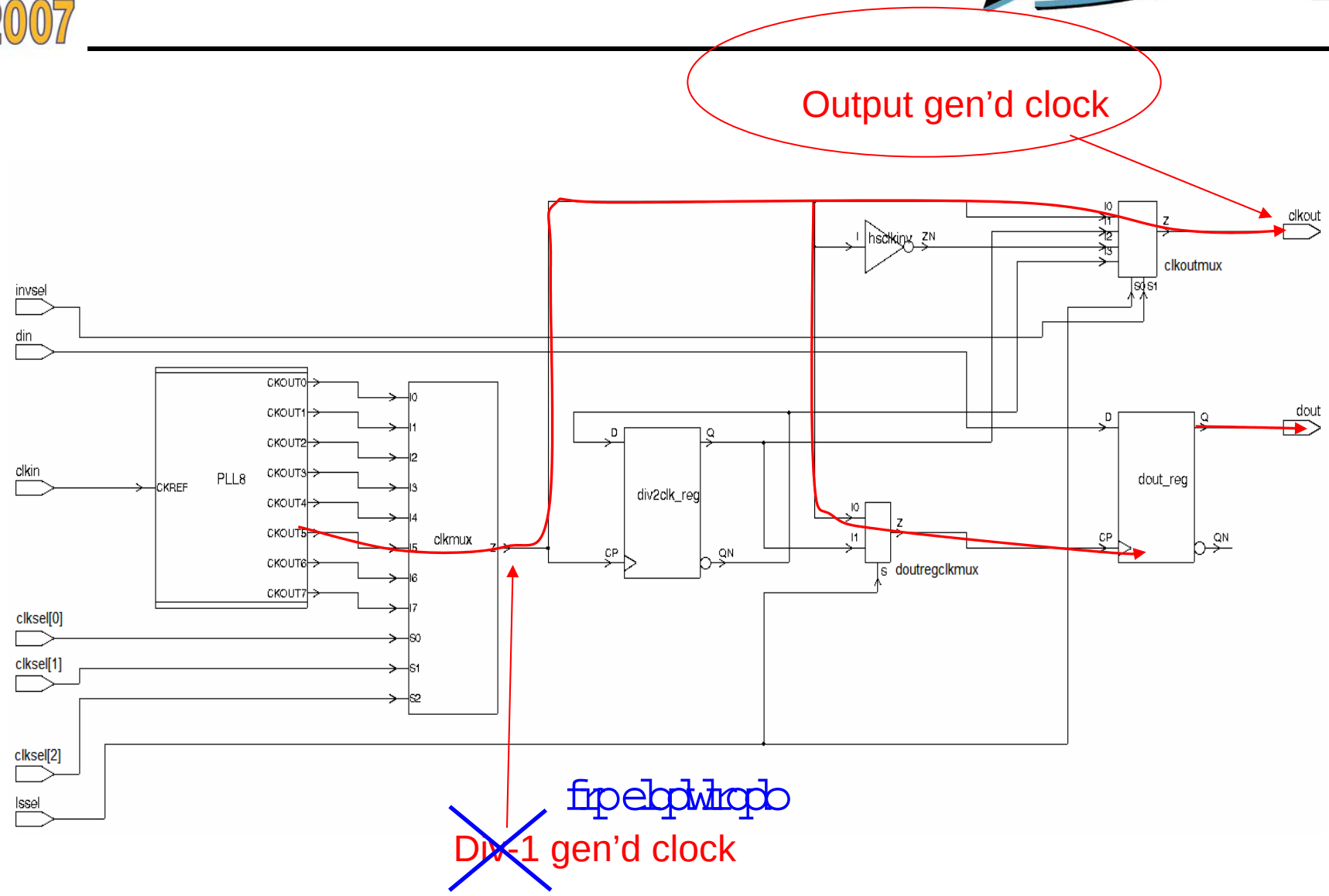
“clocks” attribute revisited



Tap clocks now blocked by generated clock

```
pt_shell> get_attribute [get_pins div2clk_reg/CP] clocks
{"modeH1clk"}
```

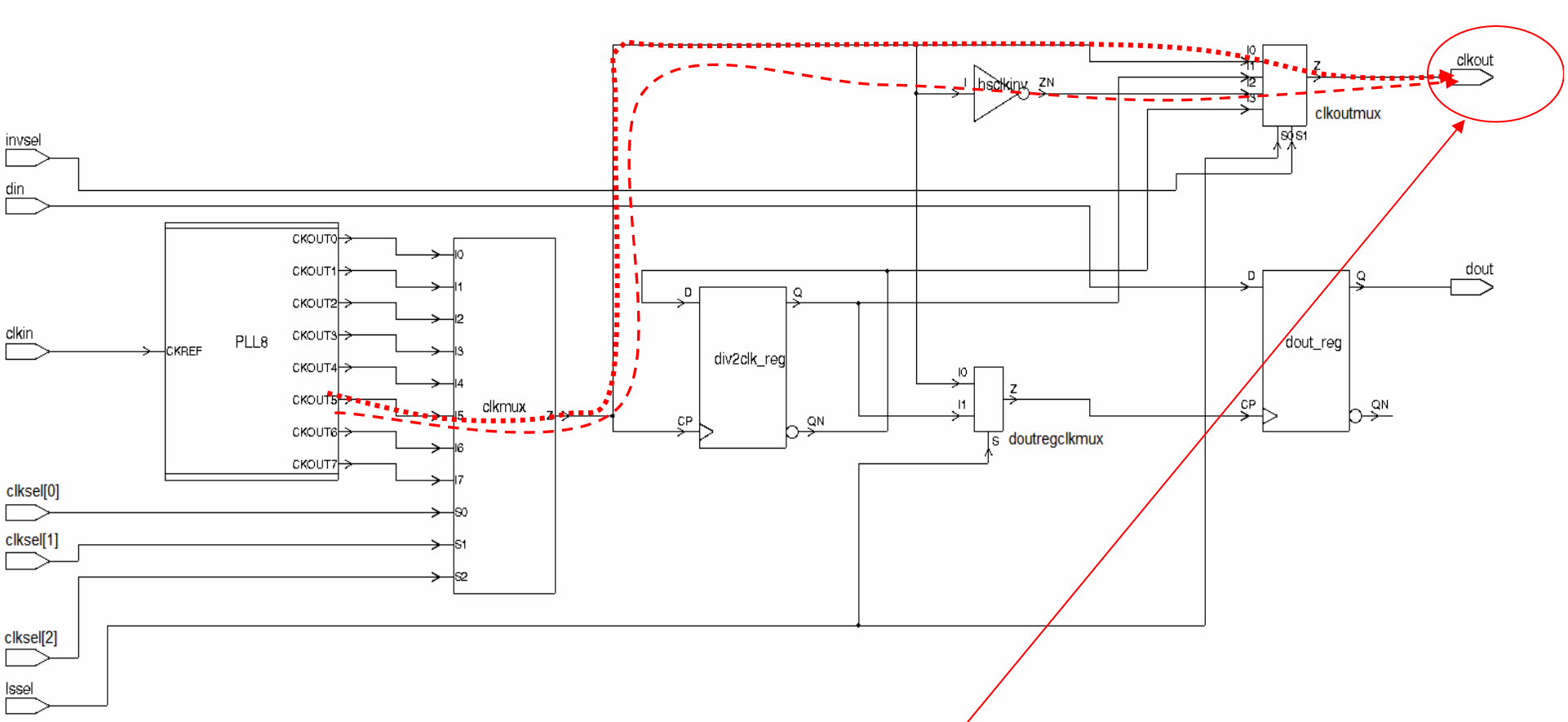
H1 mode


Output gen'd clock
clkout
clkoutmux
hsclkinv
invsel
din
clkin
CKREF
PLL8
CKOUT0
CKOUT1
CKOUT2
CKOUT3
CKOUT4
CKOUT5
CKOUT6
CKOUT7
clkmux
div2clk_reg
dout
dout_reg
doutregclkmux
clksel[0]
clksel[1]
clksel[2]
lssel
frpelbplwlrqpbo
Div-1 gen'd clock

Output clock

```
create_generated_clock  \
  -name modeH1clkout \
  -source [get_attribute [get_clocks modeH1clk] sources] \
  -comb \
  -master_clock modeH1clk \
  -add \
  [get_ports clkout]

set_output_delay -add [expr -1 * 0.1] -clock modeH1clkout -min
[get_ports dout]
set_output_delay 0.5 -clock modeH1clkout -max [get_ports dout]


set_annotated_delay 1.0 -cell -from hsclkinv/I -to hsclkinv/ZN
```

Which path?



create_generated_clock here

Output clock

New for 2006.12 –

Generated clocks that REALLY WORK!!

Output clock

Old behavior:

PT didn’t know which path to take – so it “took both” (promiscuous clock problem)

New behavior (2006.12):

PT figures out (based on –invert switch) whether path ought to be inverted and takes the CORRECT path!

Prior to 2006.12

```
report_timing -delay min -to dout -path full_clock_expanded -input

  Point                                          Incr       Path
  ---------------------------------------------------------------
  clock modeH1clk (rise edge)                    2.50       2.50
  clock hsclk_p5 (source latency)                0.00       2.50
  PLL8/CKOUT5 (DUMMYPLL8)                        0.00       2.50 r
  clkmux/I5 (mx08d1)                             0.00       2.50 r
  clkmux/Z (mx08d1) (gclock source)              0.62       3.12 r
  doutregclkmux/I0 (mx02d0)                      0.00       3.12 r
  doutregclkmux/Z (mx02d0)                       0.20       3.32 r
  dout_reg/CP (dfnrb1)                           0.00       3.32 r
  dout_reg/Q (dfnrb1)                            0.32       3.64 r
  dout (out)                                     0.00       3.64 r
  data arrival time                                         3.64

  clock modeH1clkout (rise edge)                 2.50       2.50
  clock hsclk_p5 (source latency)                0.00       2.50
  PLL8/CKOUT5 (DUMMYPLL8)                        0.00       2.50 f
  clkmux/I5 (mx08d1)                             0.00       2.50 f
  clkmux/Z (mx08d1) (gclock source)              0.59       3.09 f
  hsclkinv/I (inv0d0)                            0.00       3.09 f
  hsclkinv/ZN (inv0d0)                           1.00 *     4.09 r
  clkoutmux/I2 (mx04d0)                          0.00       4.09 r
  clkoutmux/Z (mx04d0)                           0.26       4.35 r
  clkout (out)                                   0.00       4.35 r
  output external delay                          0.10       4.45
  data required time                                        4.45
  ---------------------------------------------------------------
  data required time                                        4.45
  data arrival time                                        -3.64
  ---------------------------------------------------------------
  slack (VIOLATED)                                         -0.82
```

PT is SO confused!

WITH 2006.12

```
report_timing -delay min -to dout -path full_clock_expanded –input

  Path Type: min

  Point                                          Incr       Path
  -----------------------------------------------------------------
  clock modeH1clk (rise edge)                    2.50       2.50
  clock hsclk_p5 (source latency)                0.00       2.50
  PLL8/CKOUT5 (DUMMYPLL8)                        0.00       2.50 r
  clkmux/I5 (mx08d1)                             0.00       2.50 r
  clkmux/Z (mx08d1) (gclock source)              0.62       3.12 r
  doutregclkmux/I0 (mx02d0)                      0.00       3.12 r
  doutregclkmux/Z (mx02d0)                       0.20       3.32 r
  dout_reg/CP (dfnrb1)                           0.00       3.32 r
  dout_reg/Q (dfnrb1)                            0.32       3.64 r
  dout (out)                                     0.00       3.64 r
  data arrival time                                         3.64

  clock modeH1clkout (rise edge)                 2.50       2.50
  clock hsclk_p5 (source latency)                0.00       2.50
  PLL8/CKOUT5 (DUMMYPLL8)                        0.00       2.50 r
  clkmux/I5 (mx08d1)                             0.00       2.50 r
  clkmux/Z (mx08d1) (gclock source)              0.62       3.12 r
  clkoutmux/I0 (mx04d0)                          0.00       3.12 r
  clkoutmux/Z (mx04d0)                           0.30       3.42 r
  clkout (out)                                   0.00       3.42 r
  output external delay                          0.10       3.52
  data required time                                        3.52
  -----------------------------------------------------------------
  data required time                                        3.52
  data arrival time                                        -3.64
  -----------------------------------------------------------------
  slack (MET)                                               0.12
```

Bingo!

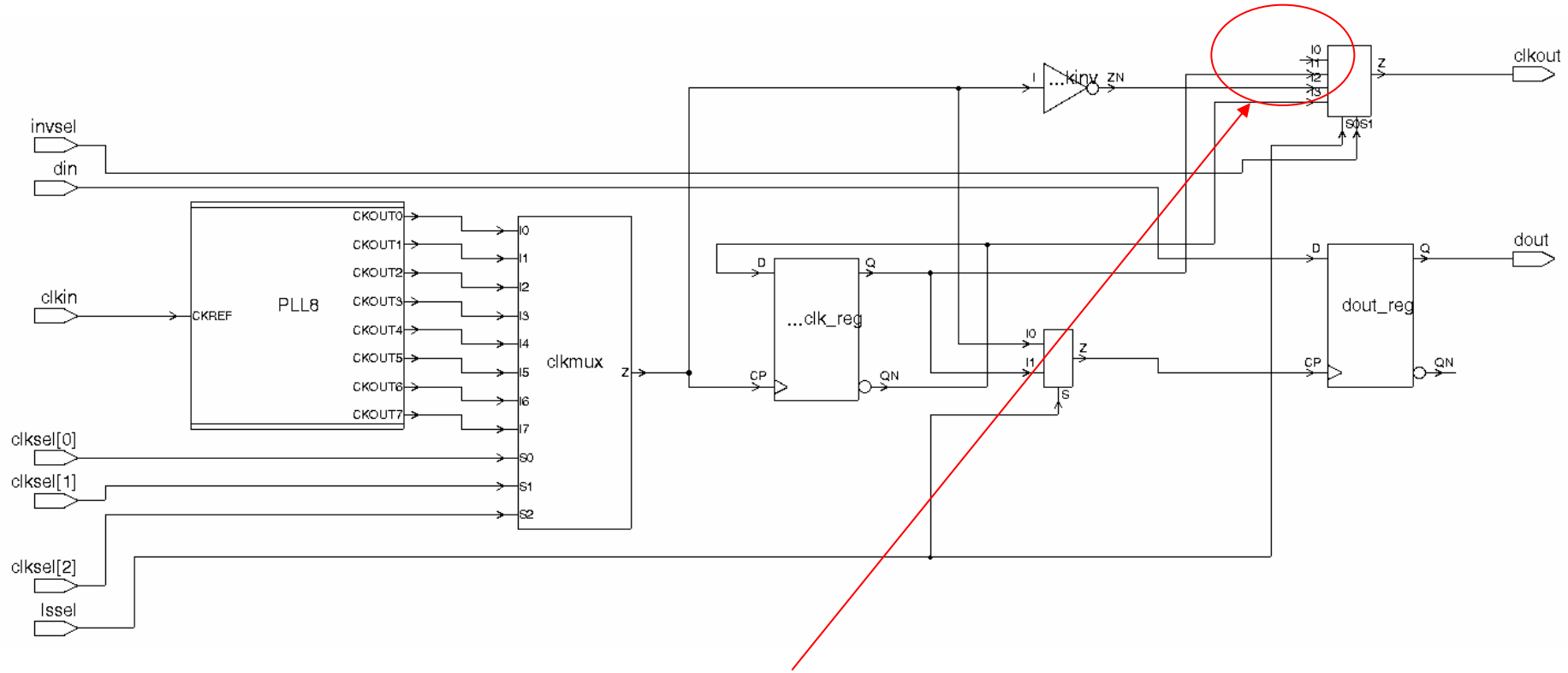
Inverter is gone

Output clock

Soooo,

That happens if the correct path doesn't exist?

Output clock



Non-inverting path now gone

Output clock

If we try to create a non-inverted clock:

```
pt_shell> create_generated_clock -name modeH1clkout -source
[get_attribute [get_clock modeH1clk] sources] -comb -master_clock
modeH1clk -add [get_ports clkout]
1
pt_shell> update_timing
…
Warning: Generated clock 'modeH1clkout' 'rise_edge' is not satisfiable;
zero source latency will be used.
 (UITE-461)
Warning: Generated clock 'modeH1clkout' 'fall_edge' is not satisfiable;
zero source latency will be used.
 (UITE-461)
```

UITE-461 is your FRIEND! If you get this, the path to your generated clock is BROKEN!

Output clock

NOTE:

UITE-461 is currently just a *warning*. Timing will be done with the generated clock having a propagation time of ZERO – so your reports will still come out, but they'll be ALL WRONG!

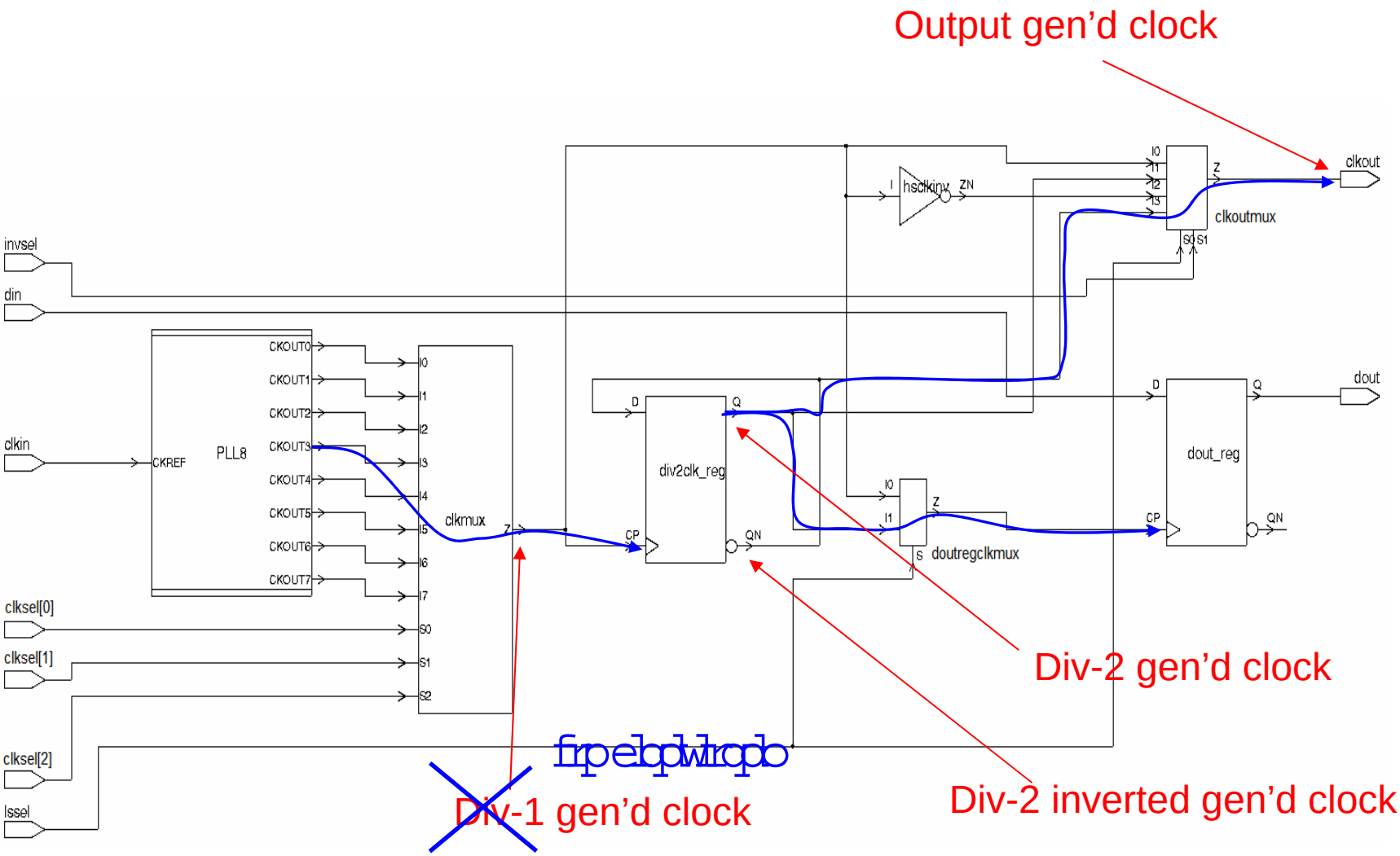
So, watch for UITE-461 !!!

L1 mode

Define settings for L1 mode:

```
set modeL1(clksel_setting) 3
set modeL1(invsel_setting) 0
set modeL1(lssel_setting) 1
```

L1 mode


Output gen'd clock
clkout
clkoutmux
hsclkinv
invsel
din
clkin
CKREF
PLL8
CKOUT0
CKOUT1
CKOUT2
CKOUT3
CKOUT4
CKOUT5
CKOUT6
CKOUT7
clkmux
div2clk_reg
dout
dout_reg
doutregclkmux
clksel[0]
clksel[1]
clksel[2]
lssel
Div-2 gen'd clock
Div-1 gen'd clock
Div-2 inverted gen'd clock

L1 mode

Create main L1 mode clock:

```
create_generated_clock  \
  -name modeL1clk \
  -source [get_attribute [get_clocks hsclk_p${modeL1(clksel_setting)}]
sources] \
  -comb \
  -master_clock hsclk_p${modeL1(clksel_setting)} \
  -add \
  [get_pins clkmux/Z]
```

L1 mode

Create divider clock:

```
create_generated_clock \
  -name modeL1div2clk \
  -source [get_attribute [get_clocks modeL1clk] sources] \
  -divide_by 2 \
  -master_clock modeL1clk \
  -add \
  [get_pins div2clk_reg/Q]
```

L1 mode

Create inverted divider clock:

```
create_generated_clock \
  -name modeL1div2clkN \
  -source [get_attribute [get_clocks modeL1clk] sources] \
  -divide_by 2 \
  -invert \
  -master_clock modeL1clk \
  -add \
  [get_pins div2clk_reg/QN]
```

L1 clocks report

```
report_clock

Attributes:
    p - Propagated clock
    G - Generated  clock
    I - Inactive   clock

Clock           Period   Waveform            Attrs     Sources
-------------------------------------------------------------------------
hsclk_p0          4.00   {0 2}               p         {PLL8/CKOUT0}
hsclk_p1          4.00   {0.5 2.5}           p         {PLL8/CKOUT1}
hsclk_p2          4.00   {1 3}               p         {PLL8/CKOUT2}
hsclk_p3          4.00   {1.5 3.5}           p         {PLL8/CKOUT3}
hsclk_p4          4.00   {2 4}               p         {PLL8/CKOUT4}
hsclk_p5          4.00   {2.5 4.5}           p         {PLL8/CKOUT5}
hsclk_p6          4.00   {3 5}               p         {PLL8/CKOUT6}
hsclk_p7          4.00   {3.5 5.5}           p         {PLL8/CKOUT7}
modeH1clk         4.00   {2.5 4.5}           p, G      {clkmux/Z}
modeH1clkout      4.00   {2.5 4.5}           p, G      {clkout}
modeH2clk         4.00   {1.5 3.5}           p, G      {clkmux/Z}
modeH2clkout      4.00   {3.5 5.5}           p, G      {clkout}
modeL1clk         4.00   {1.5 3.5}           p, G      {clkmux/Z}
modeL1div2clk     8.00   {1.5 5.5}           p, G      {div2clk_reg/Q}
modeL1div2clkN
                  8.00   {5.5 9.5}           p, G      {div2clk_reg/QN}
```

L1 mode

Create output clock :

```
create_generated_clock  \
  -name modeL1clkout \
  -source [get_attribute [get_clocks modeL1div2clk] sources] \
  -comb \
  -master_clock modeL1div2clk \
  -add \
  [get_ports clkout]
```

L1 mode

Set output delays and separate the clocks:

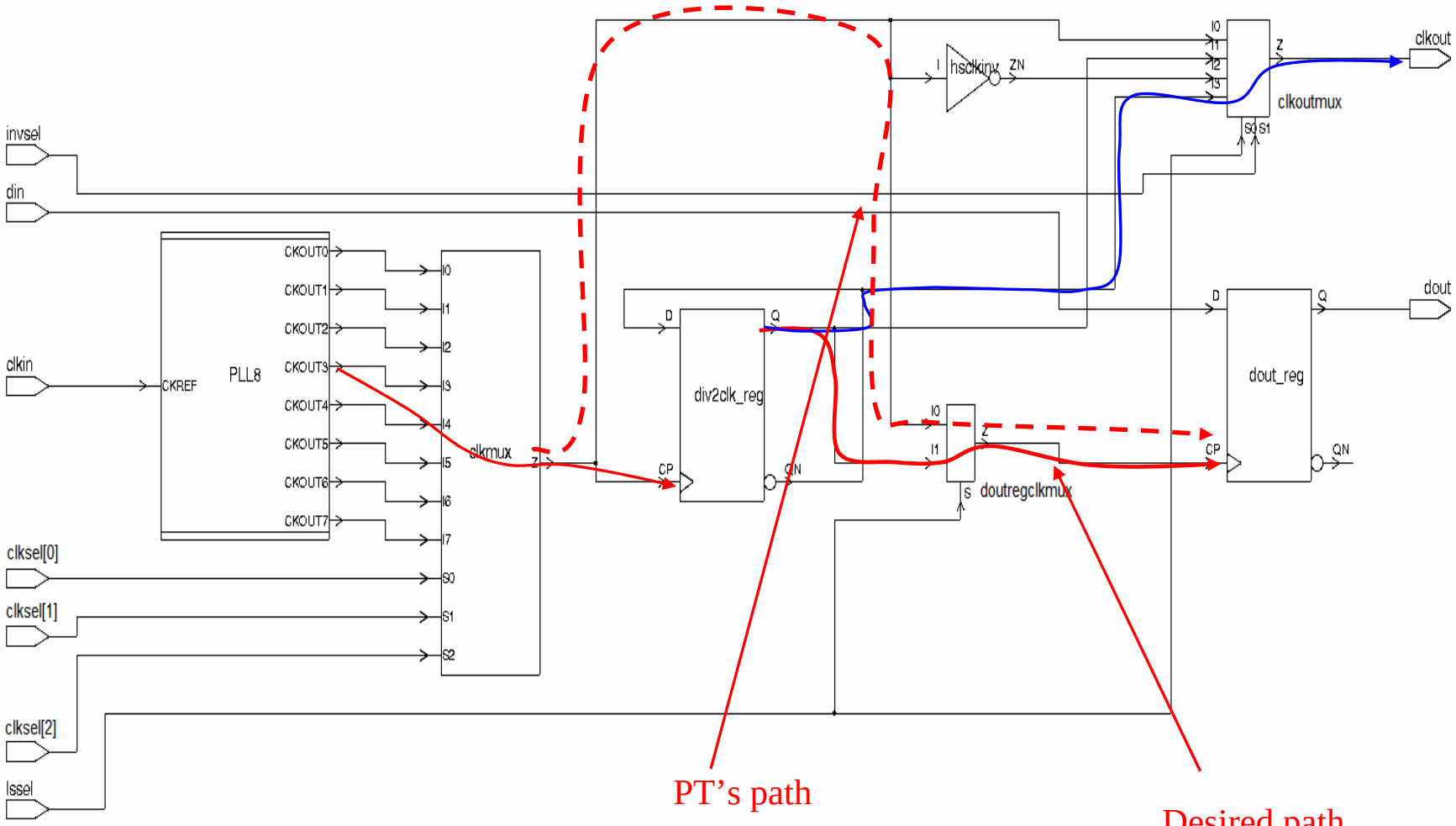
```
set_output_delay -add [expr -1 * 0.25] -clock modeL1clkout -min
[get_ports dout]
set_output_delay -add 1.7 -clock modeL1clkout -max [get_ports dout]

# Separate the modes
set_clock_groups -name muxed_out -logically_exclusive \
  -group [get_clocks modeH1*] \
  -group [get_clocks modeH2*] \
  -group [get_clocks modeL1*]
```

L1 timing report

```
report_timing -delay min -to dout -path full_clock_expanded -input -group modeL1clkout

  Point                                          Incr       Path
  ---------------------------------------------------------------
  clock modeL1clk (rise edge)                    1.50       1.50
  clock hsclk_p3 (source latency)                0.00       1.50
  PLL8/CKOUT3 (DUMMYPLL8)                        0.00       1.50 r
  clkmux/I3 (mx08d1)                             0.00       1.50 r
  clkmux/Z (mx08d1) (gclock source)              0.63       2.13 r
  doutregclkmux/I0 (mx02d0)                      0.00       2.13 r
  doutregclkmux/Z (mx02d0)                       0.20       2.33 r
  dout_reg/CP (dfnrb1)                           0.00       2.33 r
  dout_reg/Q (dfnrb1)                            0.32       2.65 r
  dout (out)                                     0.00       2.65 r
  data arrival time                                         2.65

  clock modeL1clkout (rise edge)                 1.50       1.50
  clock hsclk_p3 (source latency)                0.00       1.50
  PLL8/CKOUT3 (DUMMYPLL8)                        0.00       1.50 r
  clkmux/I3 (mx08d1)                             0.00       1.50 r
  clkmux/Z (mx08d1) (gclock source)              0.63       2.13 r
  div2clk_reg/CP (dfnrb1)                        0.00       2.13 r
  div2clk_reg/Q (dfnrb1) (gclock source)
                                                 0.41       2.54 r
  clkoutmux/I1 (mx04d0)                          0.00       2.54 r
  clkoutmux/Z (mx04d0)                           0.25       2.79 r
  clkout (out)                                   0.00       2.79 r
  output external delay                          0.25       3.04
  data required time                                        3.04
  ---------------------------------------------------------------
  data required time                                        3.04
  data arrival time                                        -2.65
  ---------------------------------------------------------------
  slack (VIOLATED)                                         -0.39
```

Path should go through div2clk_reg!

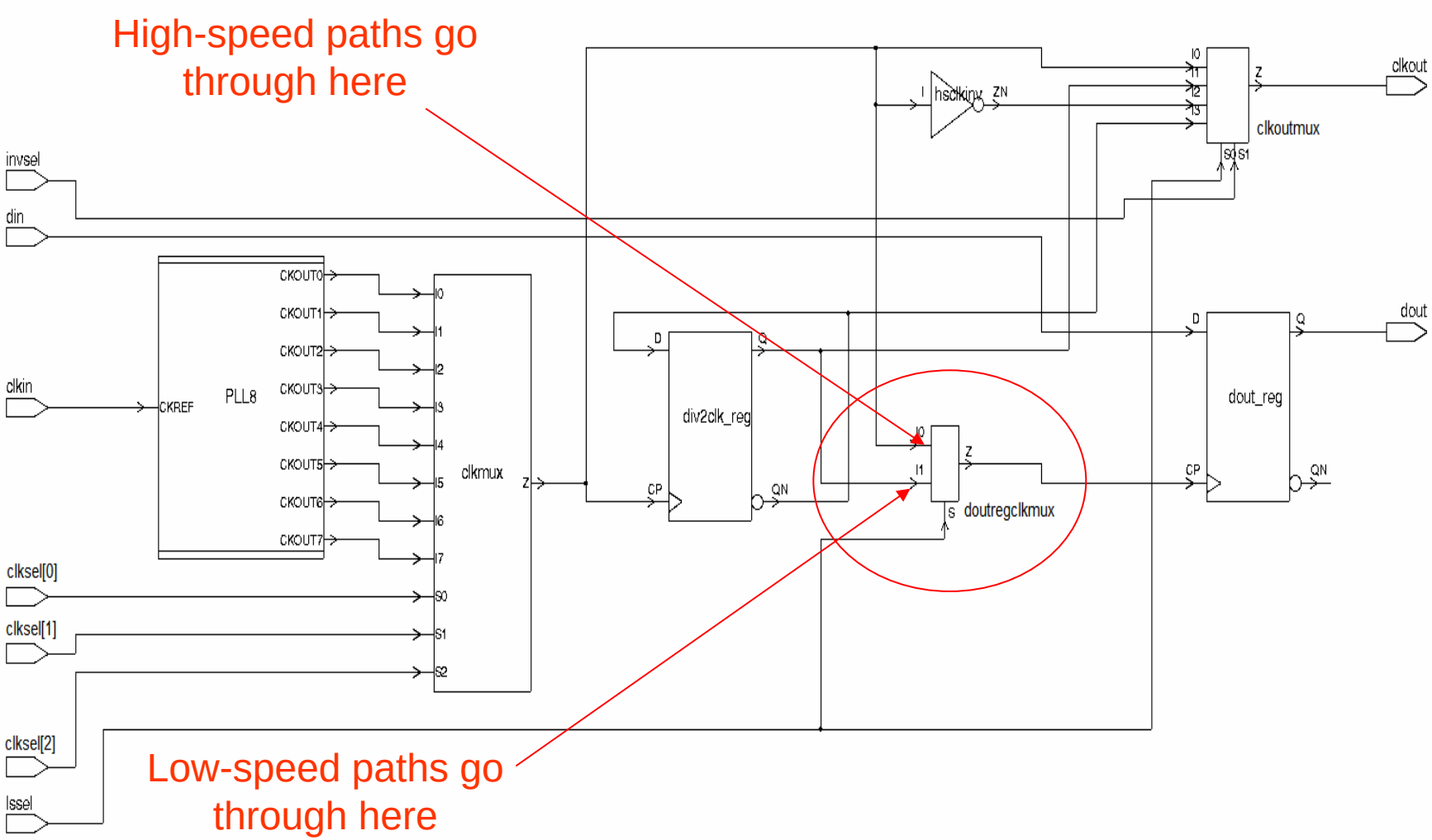
Which path to flop?



L1 mode

This could get very messy with steering clocks, since the new gen'd clock will block other clocks through that point. We would have to create unnecessary steering clocks for all the other clocks just to let them pass.

Which path to flop?



L1 mode

Instead, kill L* clocks going through hs path:

```
set_clock_sense -stop -clock [get_clocks modeL*] [get_pins doutregclkmux/I0]
```

L1 mode min timing report

Point	Incr	Path
clock modeL1div2clk (rise edge)	1.50	1.50
clock hsclk_p3 (source latency)	0.00	1.50
PLL8/CKOUT3 (DUMMYPLL8)	0.00	1.50 r
clkmux/I3 (mx08d1)	0.00	1.50 r
clkmux/Z (mx08d1) (gclock source)	0.63	2.13 r
div2clk_reg/CP (dfnrb1)	0.00	2.13 r
div2clk_reg/Q (dfnrb1) (gclock source)	0.38	2.51 r
doutregclkmux/I1 (mx02d0)	0.00	2.51 r
doutregclkmux/Z (mx02d0)	0.17	2.68 r
dout_reg/CP (dfnrb1)	0.00	2.68 r
dout_reg/Q (dfnrb1)	0.32	3.00 r
dout (out)	0.00	3.00 r
data arrival time		3.00
clock modeL1clkout (rise edge)	1.50	1.50
clock hsclk_p3 (source latency)	0.00	1.50
PLL8/CKOUT3 (DUMMYPLL8)	0.00	1.50 r
clkmux/I3 (mx08d1)	0.00	1.50 r
clkmux/Z (mx08d1) (gclock source)	0.63	2.13 r
div2clk_reg/CP (dfnrb1)	0.00	2.13 r
div2clk_reg/Q (dfnrb1) (gclock source)	0.41	2.54 r
clkoutmux/I1 (mx04d0)	0.00	2.54 r
clkoutmux/Z (mx04d0)	0.25	2.79 r
clkout (out)	0.00	2.79 r
output external delay	0.25	3.04
data required time		3.04
data required time		3.04
data arrival time		-3.00
slack (VIOLATED)		-0.04

L1 mode

What about the H* clocks going through I1?

H* div2 clocks for noise

```
# Create both pos and neg divided clocks for noise
create_generated_clock \
  -name modeH1div2clk \
  -source [get_attribute [get_clocks modeH1clk] sources] \
  -divide_by 2 \
  -master_clock modeH1clk \
  -add \
  [get_pins div2clk_reg/Q]

create_generated_clock \
  -name modeH1div2clkN \
  -source [get_attribute [get_clocks modeH1clk] sources] \
  -divide_by 2 \
  -invert \
  -master_clock modeH1clk \
  -add \
  [get_pins div2clk_reg/QN]
```

L1 mode

So, also kill H* clocks going through ls path:

```
set_clock_sense -stop -clock [get_clocks modeH*] [get_pins doutregclkmux/I1]
```

Conclusion

Multiclock propagation – it's not just for gearheads anymore!

Use 2006.12! It's got lots of new features!

We need this stuff in DC !!

Thank you !