DIGILENT®

A National Instruments Company

SD Card Interface Timing

Analysis, Implementation and Results

Ioan Catuna

www.digilentinc.com Jan. 2020

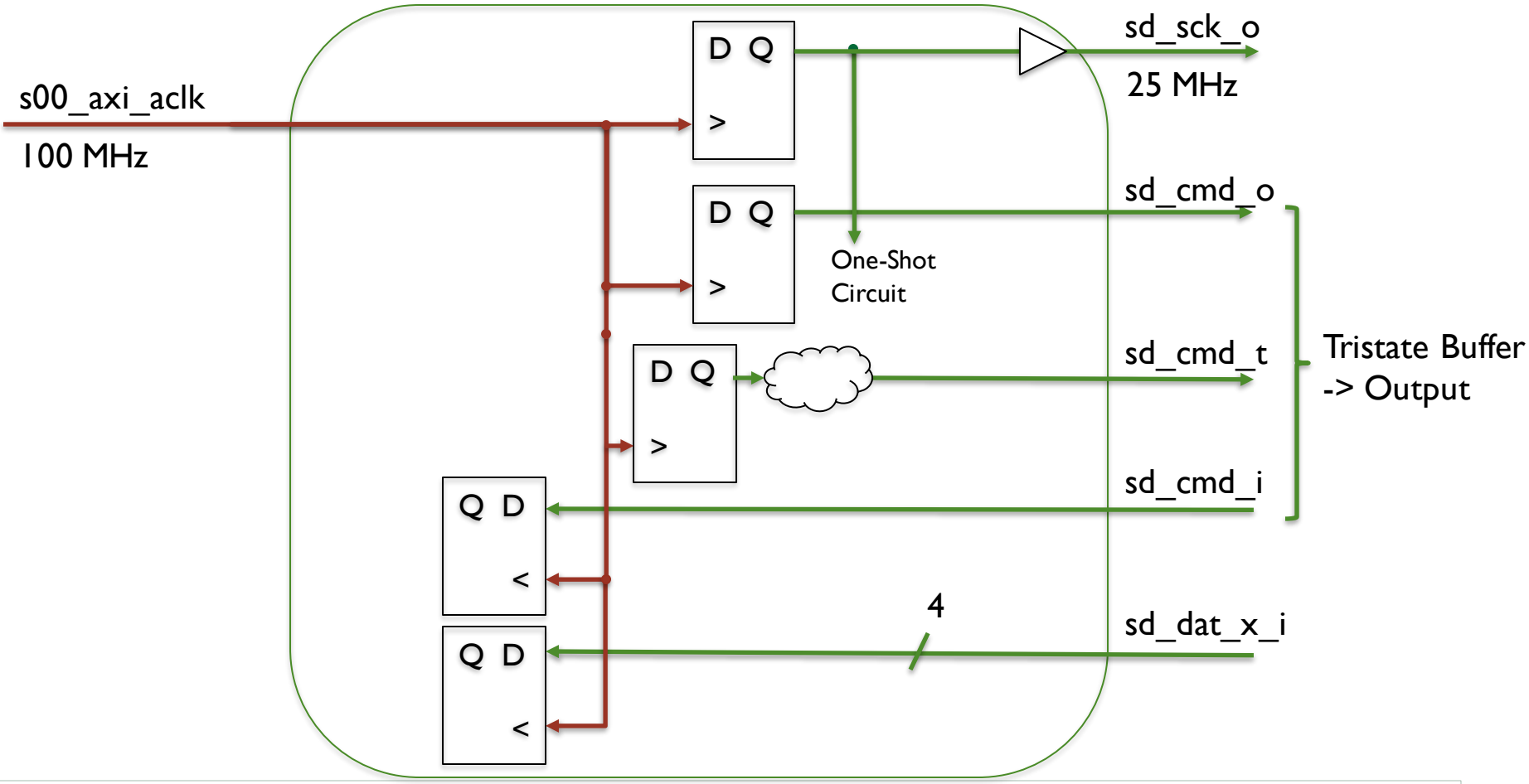
Contents

- SD Card Interface Design
- Timing Paths
- Timing Constraints
- Timing Analysis Results
- References

Contents

- **SD Card Interface Design**
- Timing Paths
- Timing Constraints
- Timing Analysis Results
- References

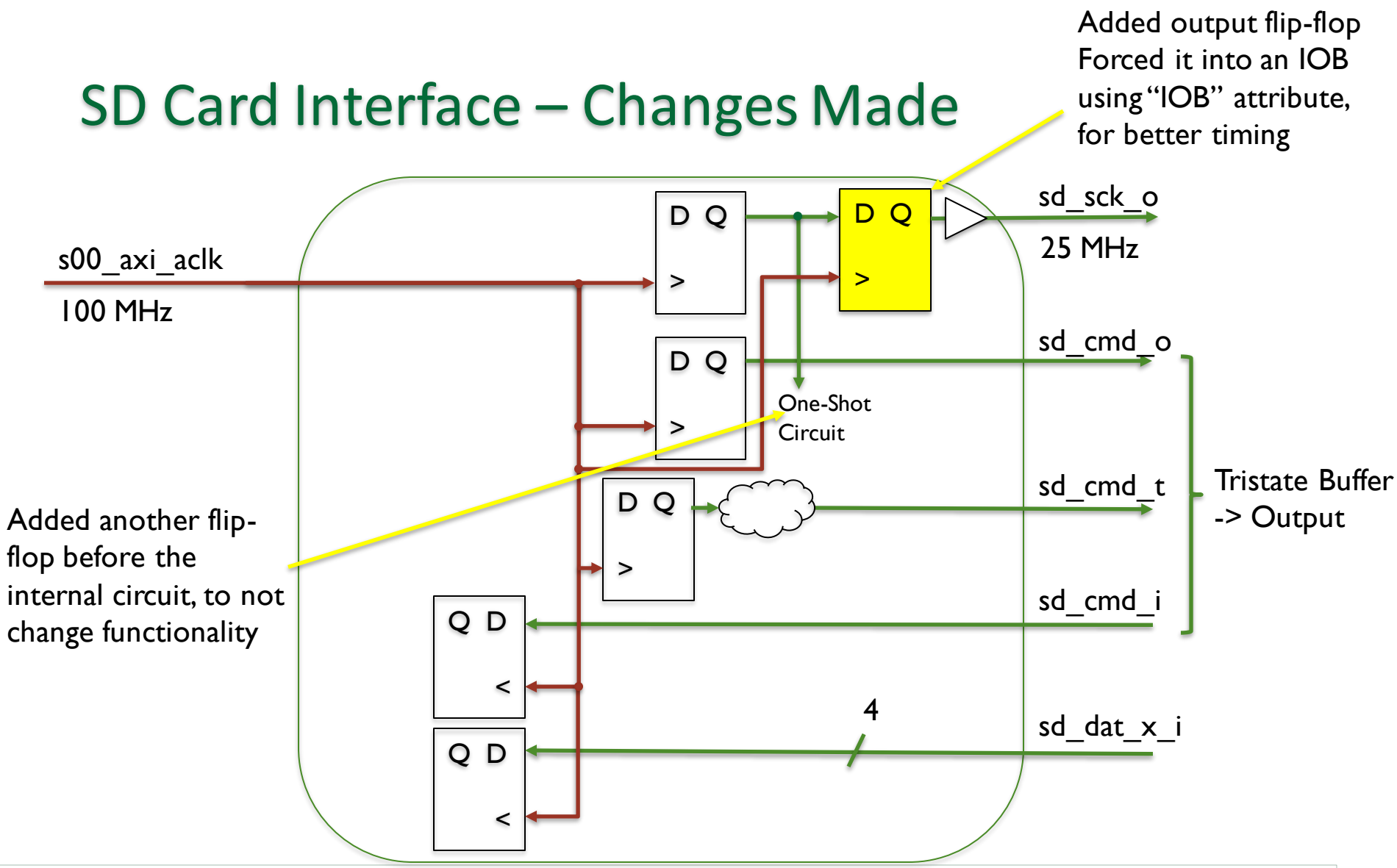
SD Card Interface – Original Design



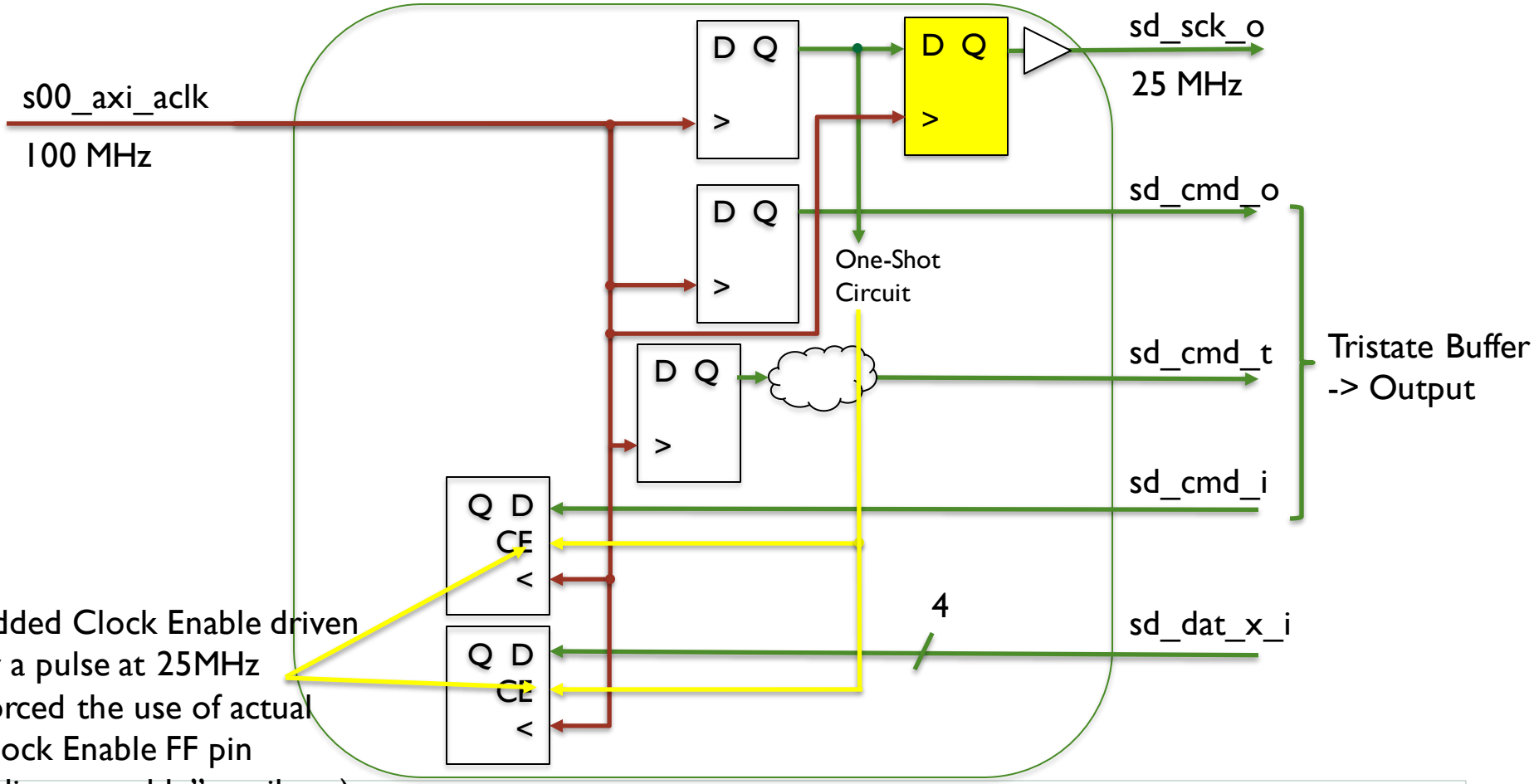
SD Card Interface – Original Design Problems

- Original design had many data integrity issues when reading from the SD card
  - Occurring too often to be metastability issues
  - More likely timing issues, or timing + metastability
- Original design had no timing constraints on the SD Card interface
  - Decided to review interface design & add constraints

SD Card Interface – Changes Made



SD Card Interface – Changes Made



Added Clock Enable driven by a pulse at 25MHz
Forced the use of actual Clock Enable FF pin
("direct_enable" attribute)

DIGILENT®
A National Instruments Company

Contents

- SD Card Interface Design
- **Timing Paths**
- Timing Constraints
- Timing Analysis Results
- References

SD Card Input Timing - Default Speed

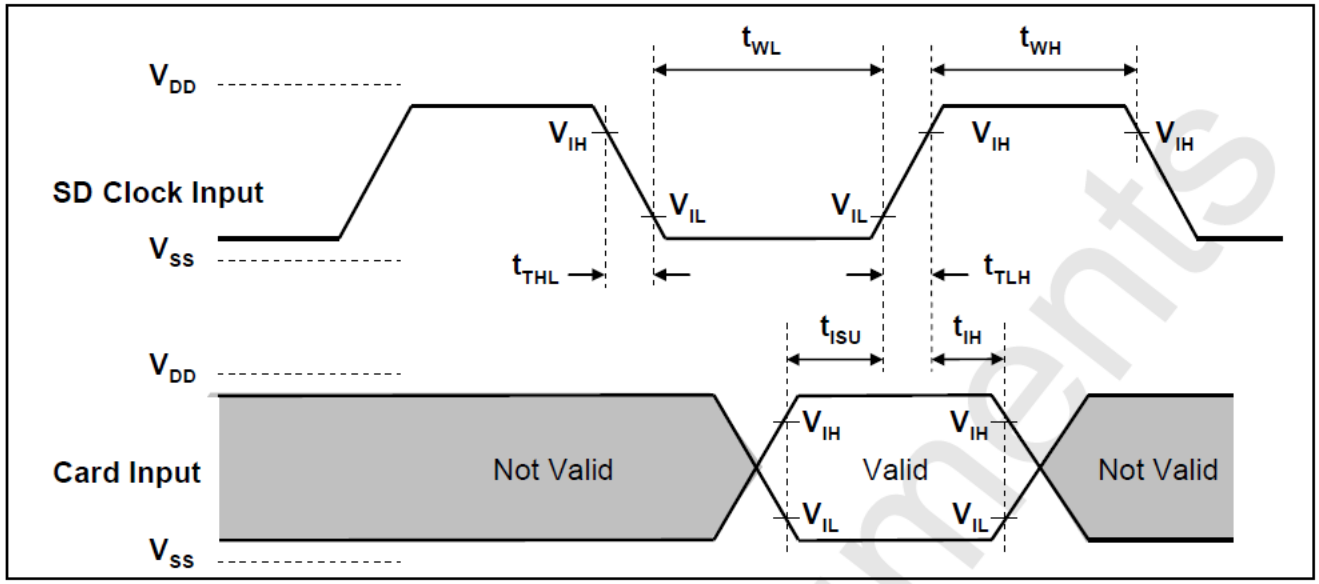


Figure 6-10 : Card Input Timing (Default Speed Card)

Inputs CMD, DAT (referenced to CLK)					
Input set-up time	t_{ISU}	5		ns	$C_{CARD} \leq$ 10 pF (1 card)
Input hold time	t_{IH}	5		ns	$C_{CARD} \leq$ 10 pF (1 card)

DIGILENT®
A National Instruments Company

SD Card Output Timing - Default Speed

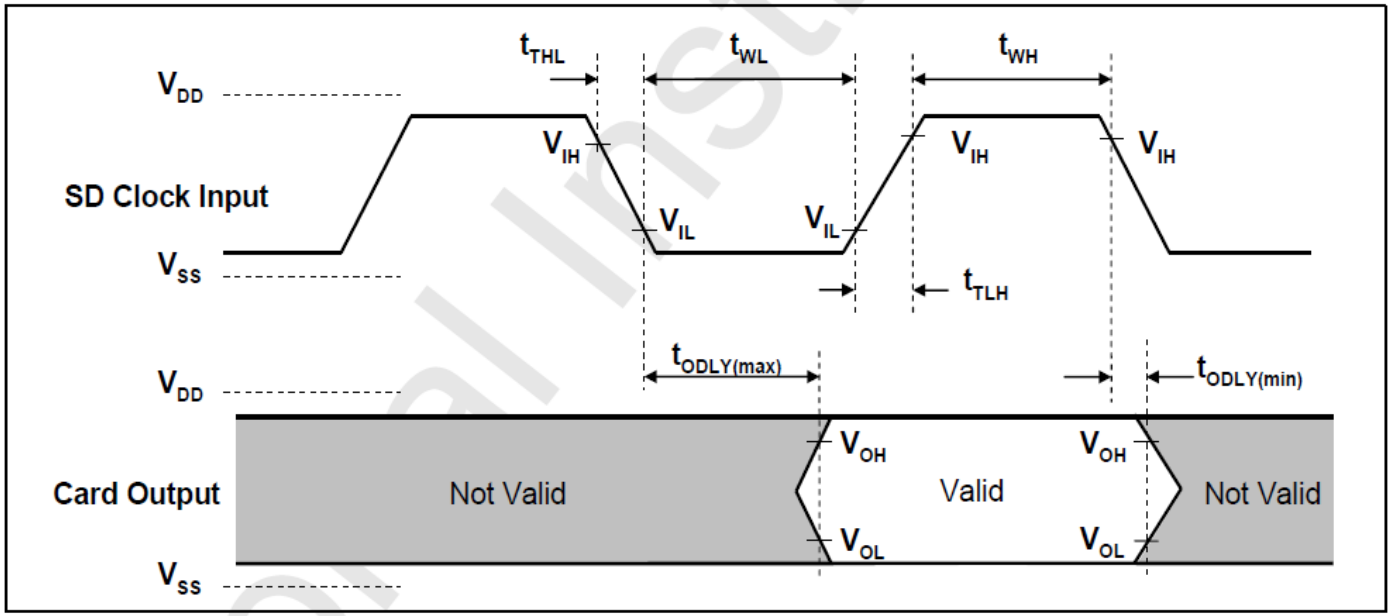
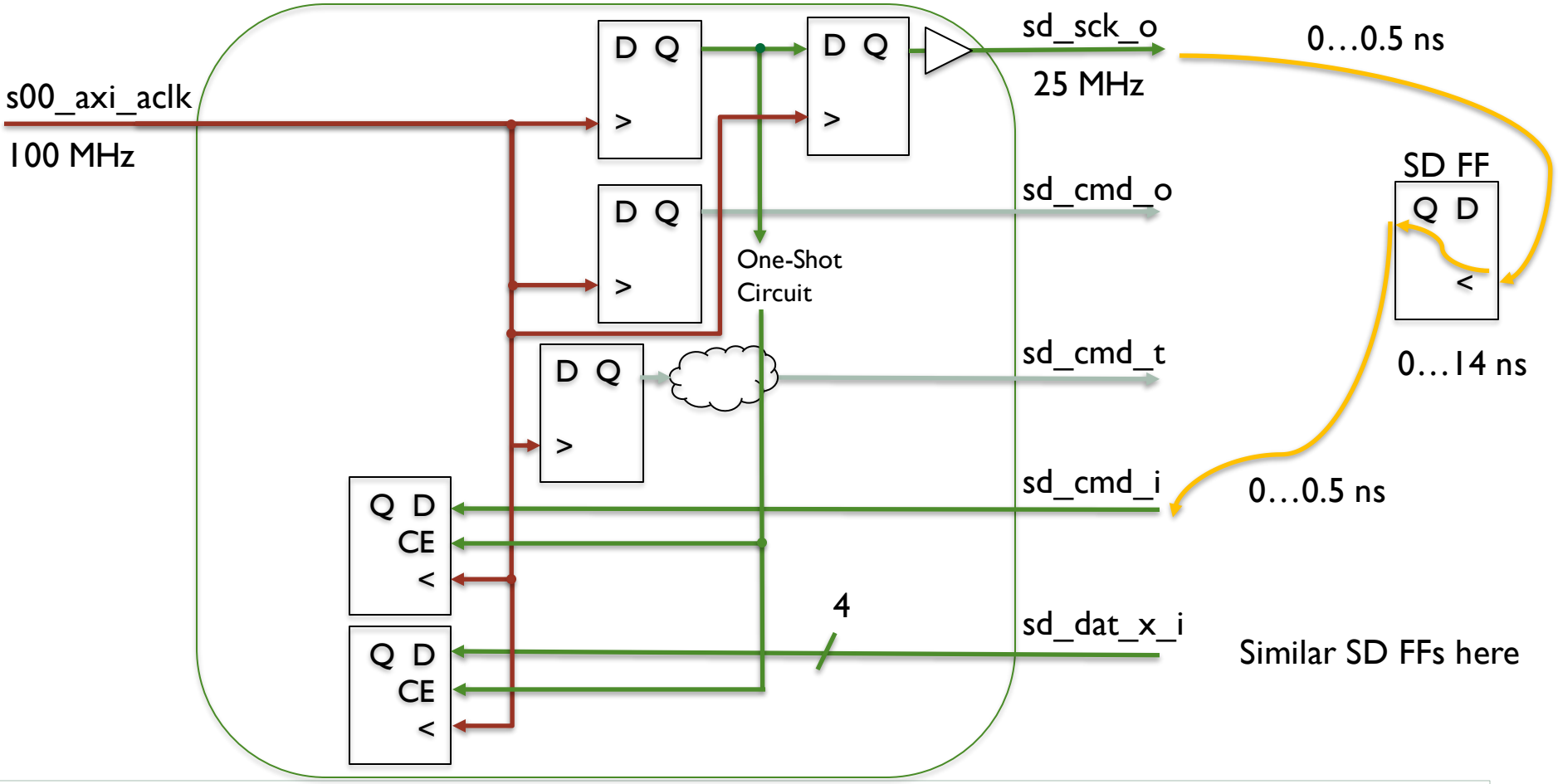


Figure 6-11 : Card Output Timing (Default Speed Mode)

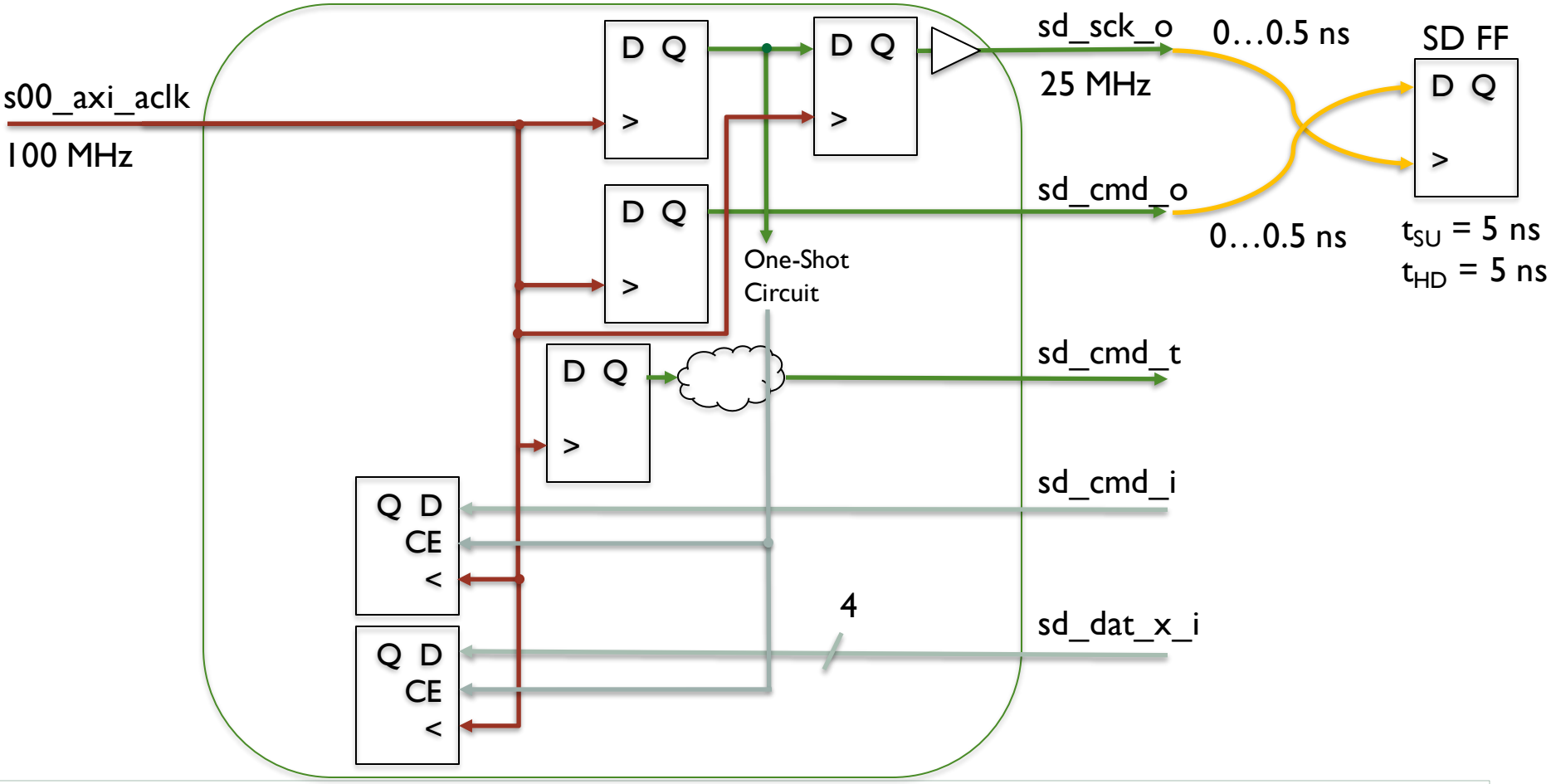
Outputs CMD, DAT (referenced to CLK)					
Output Delay time during Data Transfer Mode	t_{ODLY}	**0**	**14**	**ns**	$C_L \leq$ **40 pF (1 card)**
Output Delay time during Identification Mode	t_{ODLY}	**0**	**50**	**ns**	$C_L \leq$ **40 pF (1 card)**

DIGILENT®
A National Instruments Company

Timing Paths – Inputs to the Core
System Synchronous Inputs



Timing Paths – Outputs from the Core
Source Synchronous Outputs



Contents

- SD Card Interface Design
- Timing Paths
- **Timing Constraints**
- Timing Analysis Results
- References

Initial Considerations

- SD Card Interface Core is an IP to be integrated in different projects
- Must define its constraints at IP level, not at project top-level
- Following constraints based on Xilinx QSPI IP constraints

Challenges with Constraints at IP Level

- Don’t know the input clock name
  - Must detect it
- Don’t know top-level port names
  - Must use IP port names instead
  - Use `get_ports` command

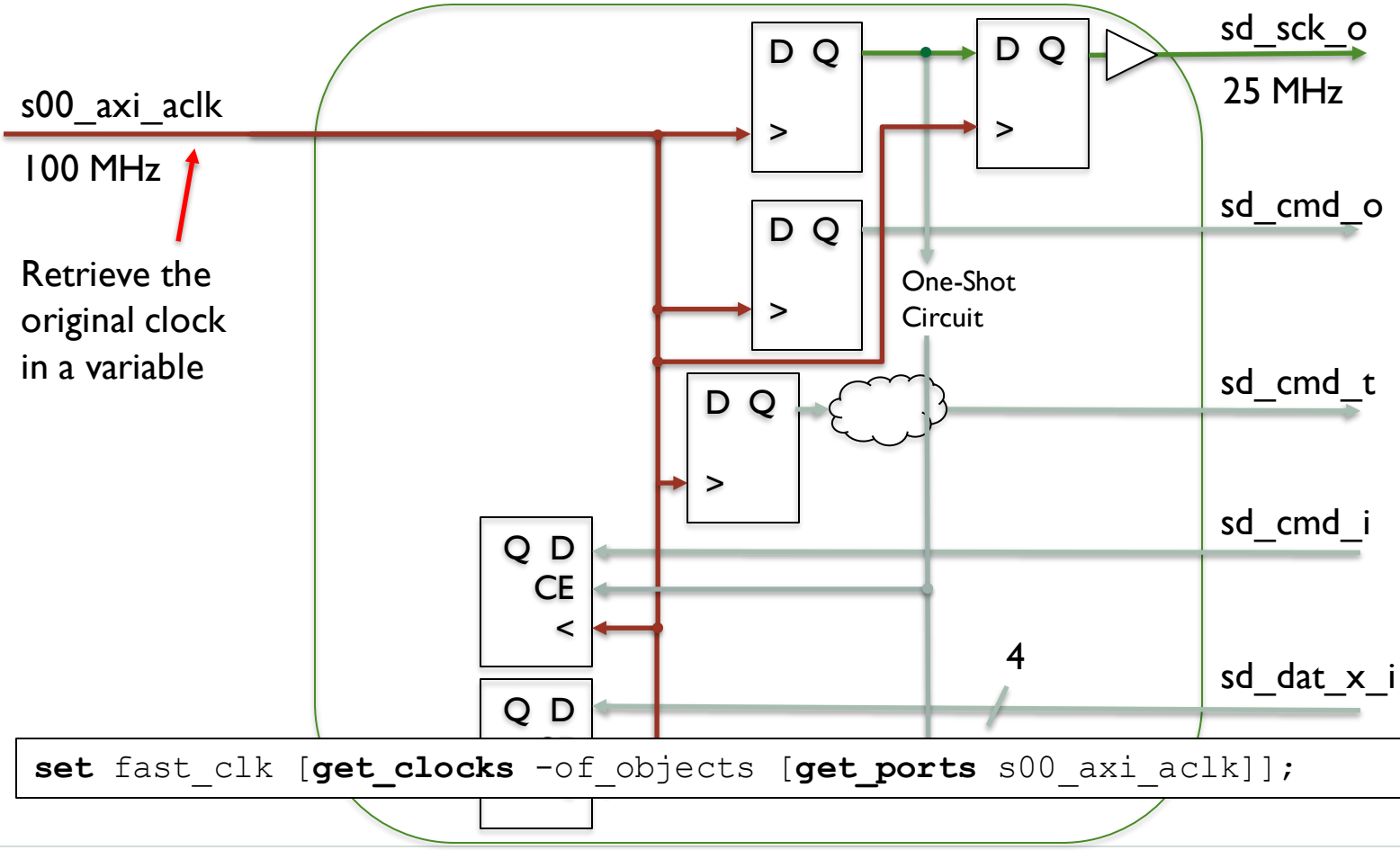
Challenges with Constraints at IP Level

- For `set_input_delay` and `set_output_delay` constraints, we must find the FPGA top-level ports
- What if IBUFs/OBUFs are not defined inside the IP (quite typical)?
  - Vivado converts a `get_ports` command inside the IP into a `get_pins` command at top-level
  - `get_pins` does not automatically "jump" over IBUFs/OBUFs
  - Must use `-prop_thru_buffers` attribute

Challenges with Constraints at IP Level

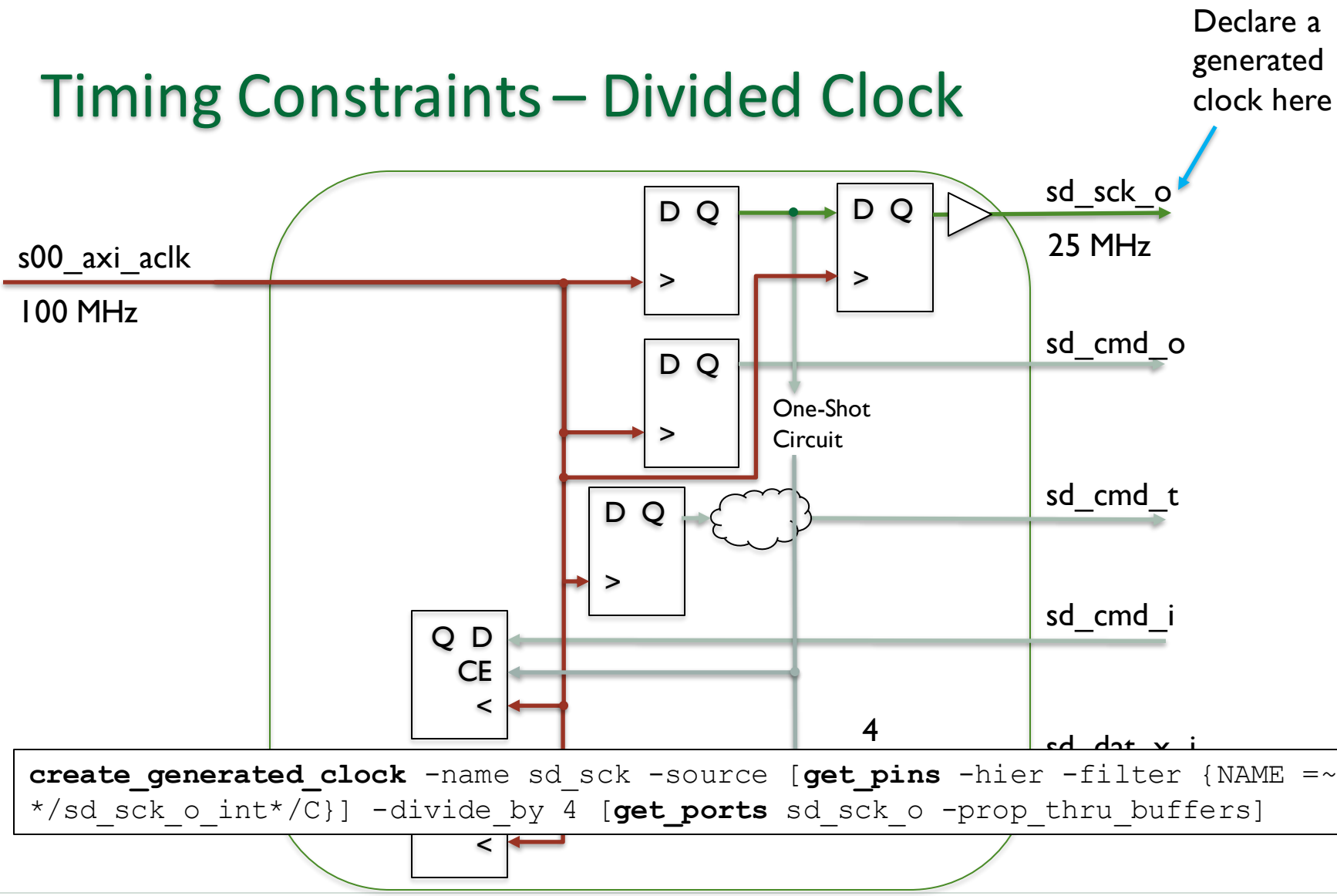
- For the IP XDC file, set processing order in Vivado to LATE
  - Thus making sure upper-level constraints are already in place (e.g. input clocks)

Timing Constraints – Original Clock




```
set fast_clk [get_clocks -of_objects [get_ports s00_axi_aclk]];
```

Timing Constraints – Divided Clock




```
create_generated_clock -name sd_sck -source [get_pins -hier -filter {NAME =~ */sd_sck_o_int*/C}] -divide_by 4 [get_ports sd_sck_o -prop_thru_buffers]
```

Timing Constraints – SD Card Cmd & Data Lines

- Vivado timing analysis considers the SD Card interface to be synchronous to sd_sck
- The circuit inside the SD Card core is synchronous to fast_clk
- We automatically have **inter-clock paths**
- They need to be properly analyzed by the tools
- We need to tell the tools **how** to properly analyze them

Timing Constraints – Between the 2 Clocks

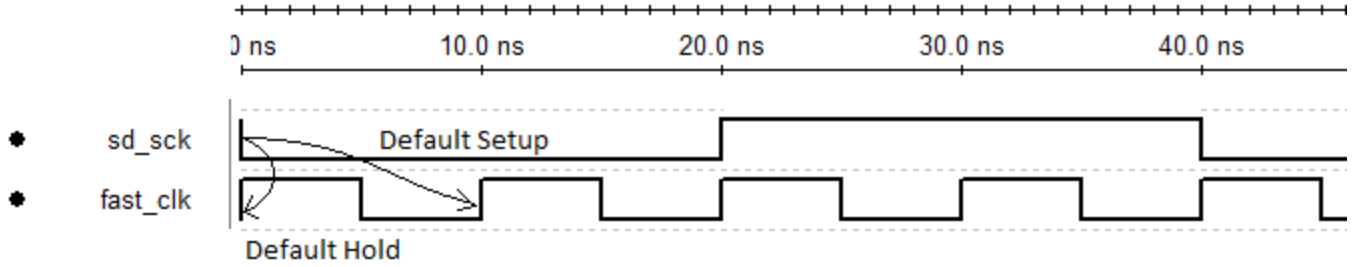
- We are dealing with multicycle paths
  - “sd_sck” domain – to – “fast_clk” domain
  - “fast_clk” domain – to – “sd_sck” domain
- In the SD Core, data is always generated and read in the “fast_clk” domain
- “sd_sck” is only used outside the FPGA

Timing Constraints – Between the 2 Clocks

- XDC syntax offers the `set_multicycle_path -from -to` constraint
  - The `-setup` path multiplier is by default specified versus the destination clock
  - The `-hold` path multiplier is by default specified versus the source clock
- We need to force the tools to consider "fast_clk" as reference for both setup and hold analysis, for both directions
  - Use `-start` (for setup) and `-end` (for hold) attributes where needed

Timing Constraints – From SD Clock to Fast Clock

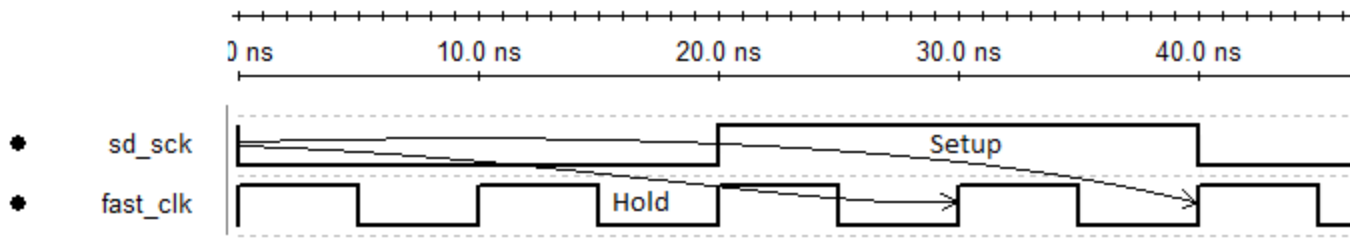
- Use `-end` attribute for `-hold` analysis!
- By default, setup is analyzed on the next destination clock edge
- Hold is analyzed one clock edge before



Timing Constraints – From SD Clock to Fast Clock

- We can specify another clock edge instead
  - For setup analysis, we need to move forward on the timing diagram

```
set_multicycle_path 4 -setup -from [get_clocks sd_sck] -to $fast_clk
```

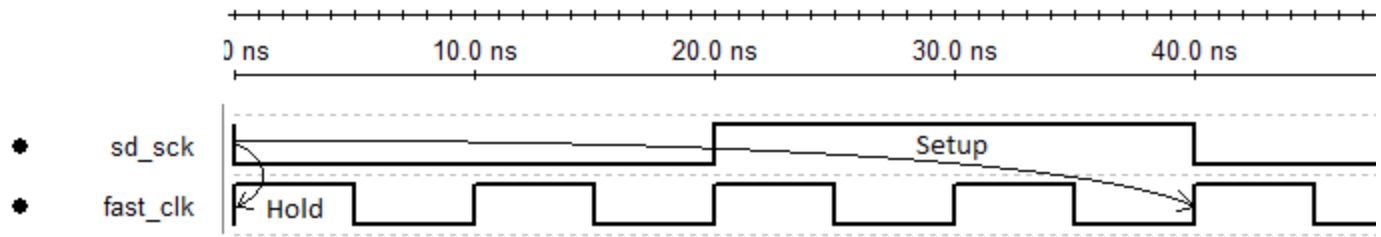


- Setup analysis is now done on the correct edge
- Hold analysis needs more work

Timing Constraints – From SD Clock to Fast Clock

- For hold analysis, we need to move backward on the timing diagram

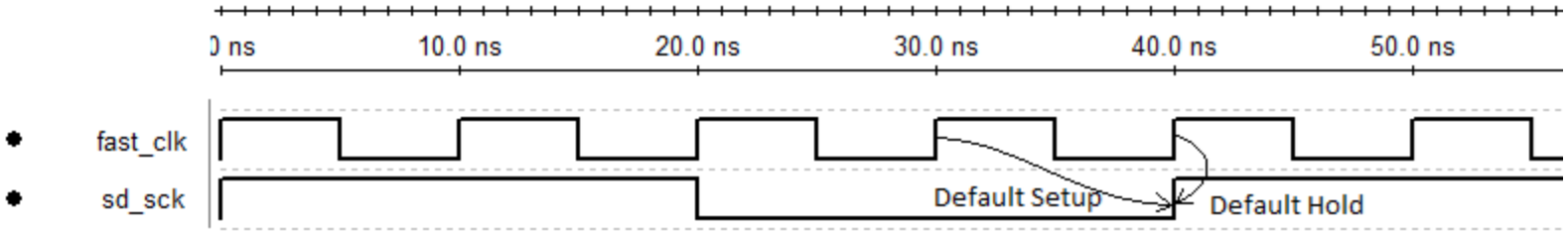
```
set_multicycle_path 3 -hold -end -from [get_clocks sd_sck] -to $fast_clk
```



- Both setup analysis and hold analysis now look good

Timing Constraints – From Fast Clock to SD Clock

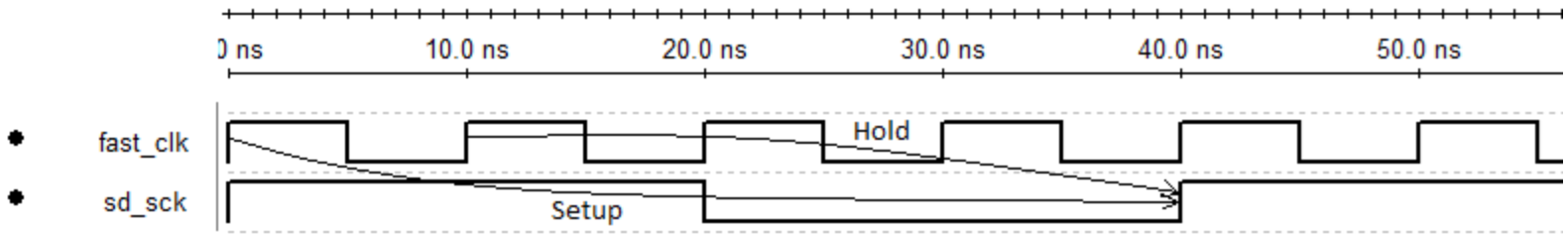
- Use `-start` attribute for `-setup` analysis!
- By default, setup and hold analysis is analyzed in the following way:



Timing Constraints – From Fast Clock to SD Clock

- We can specify another clock edge instead
  - For setup analysis, we now move backward on the timing diagram

```
set_multicycle_path 4 -setup -start -from $fast_clk -to [get_clocks sd_sck]
```

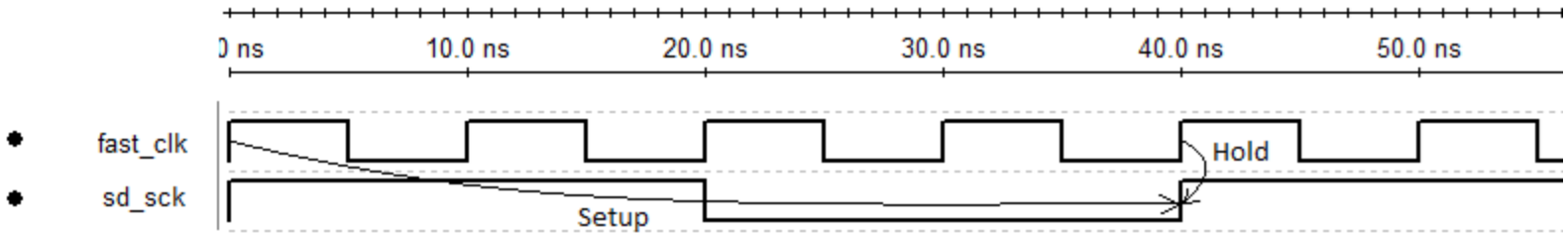


- Setup analysis is now done on the correct edge
- Hold analysis needs more work

Timing Constraints – From Fast Clock to SD Clock

- For hold analysis, we now move forward on the timing diagram

```
set_multicycle_path 3 -hold -from $fast_clk -to [get_clocks sd_sck]
```



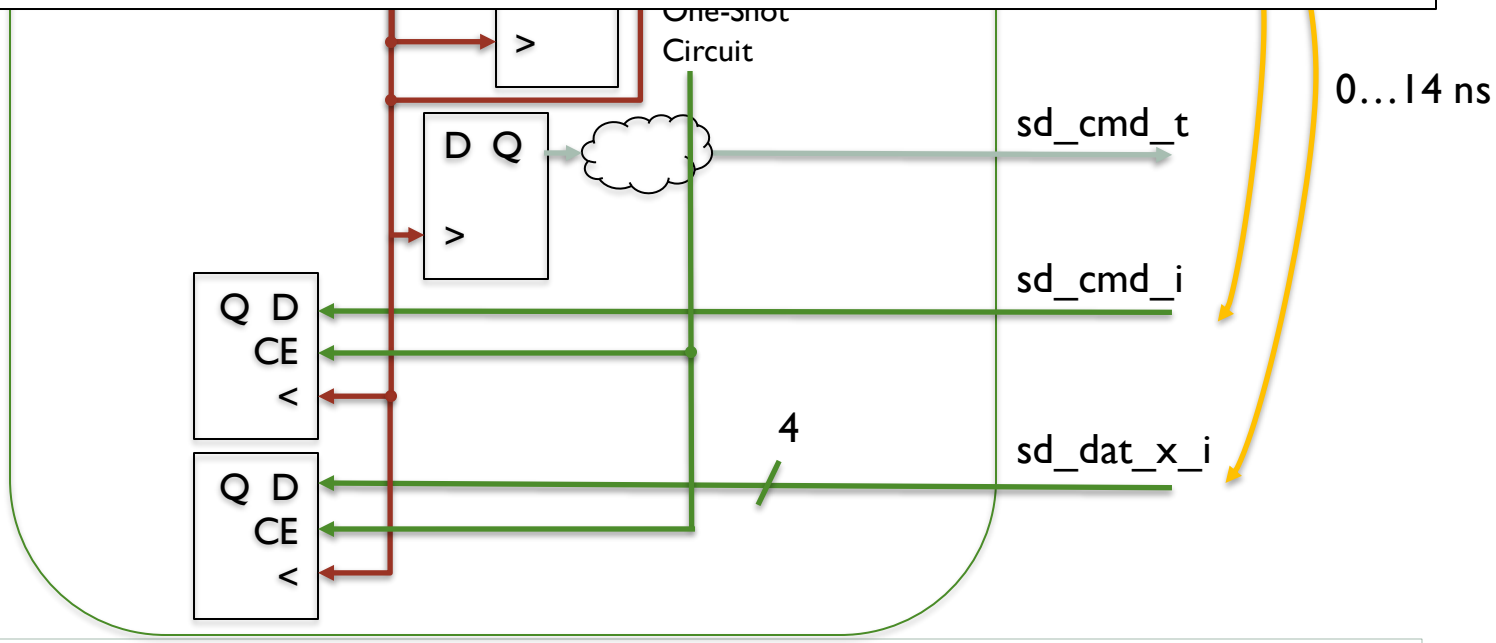
- Both setup analysis and hold analysis now look good

Timing Paths – Inputs to the Core
System Synchronous Inputs

```
set_input_delay -clock sd_sck -max -clock_fall [expr $trce_dly_max +
$tODLY_max_falling + $trce_dly_max] [get_ports sd_cmd_i -prop_thru_buffers]

set_input_delay -clock sd_sck -min -clock_fall [expr $trce_dly_min +
$tODLY_min_falling + $trce_dly_min] [get_ports sd_cmd_i -prop_thru_buffers]

# The same for all sd_dat_x_i inputs
```

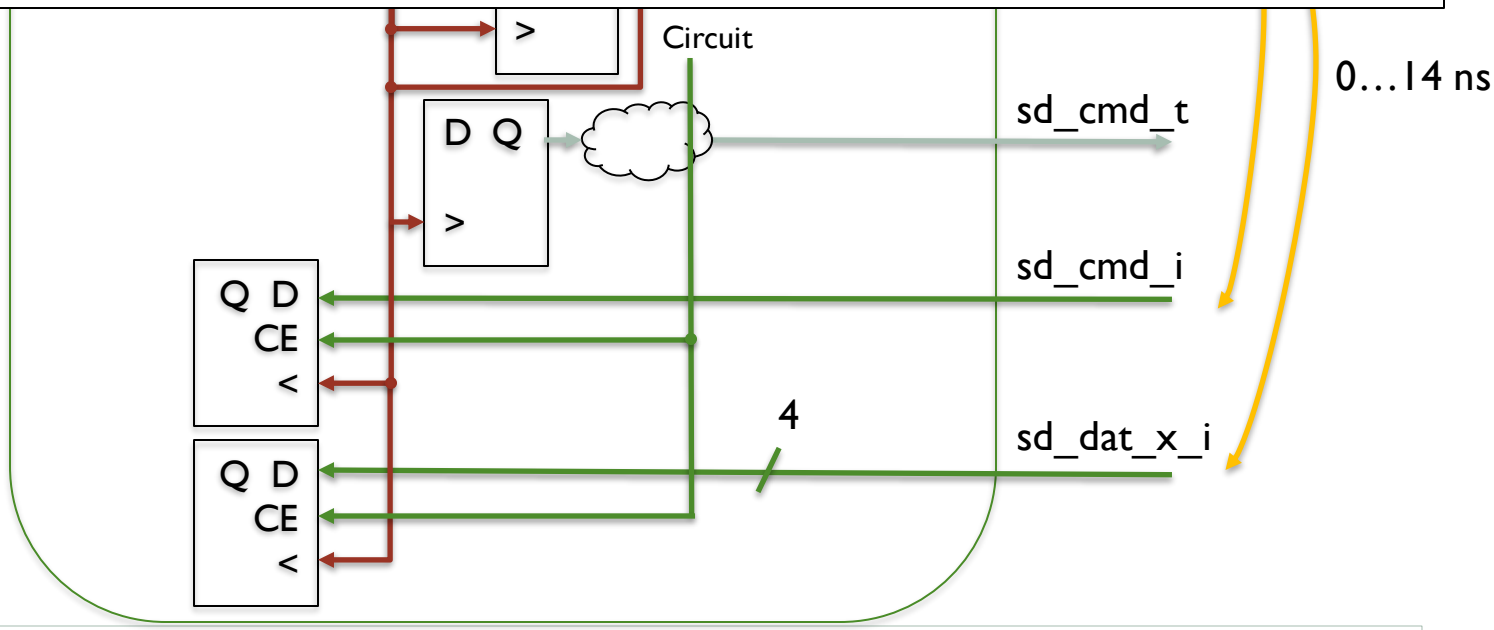


Timing Paths – Inputs to the Core
System Synchronous Inputs

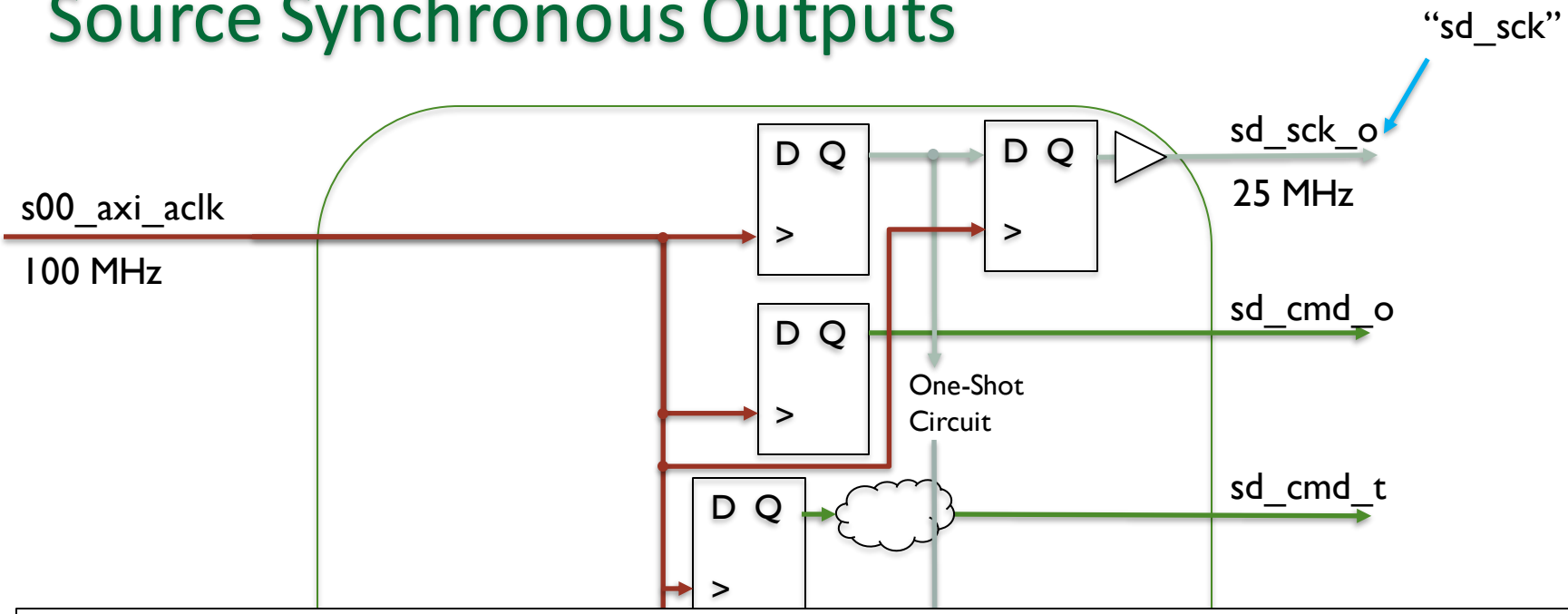
```
set_input_delay -clock sd_sck -max -clock_fall [expr 0.5 + 14 + 0.5]
[get_ports sd_cmd_i -prop_thru_buffers]

set_input_delay -clock sd_sck -min -clock_fall [expr 0 + 0 + 0] [get_ports
sd_cmd_i -prop_thru_buffers]

# The same for all sd_dat_x_i inputs
```



Timing Constraints – Command Outputs Source Synchronous Outputs

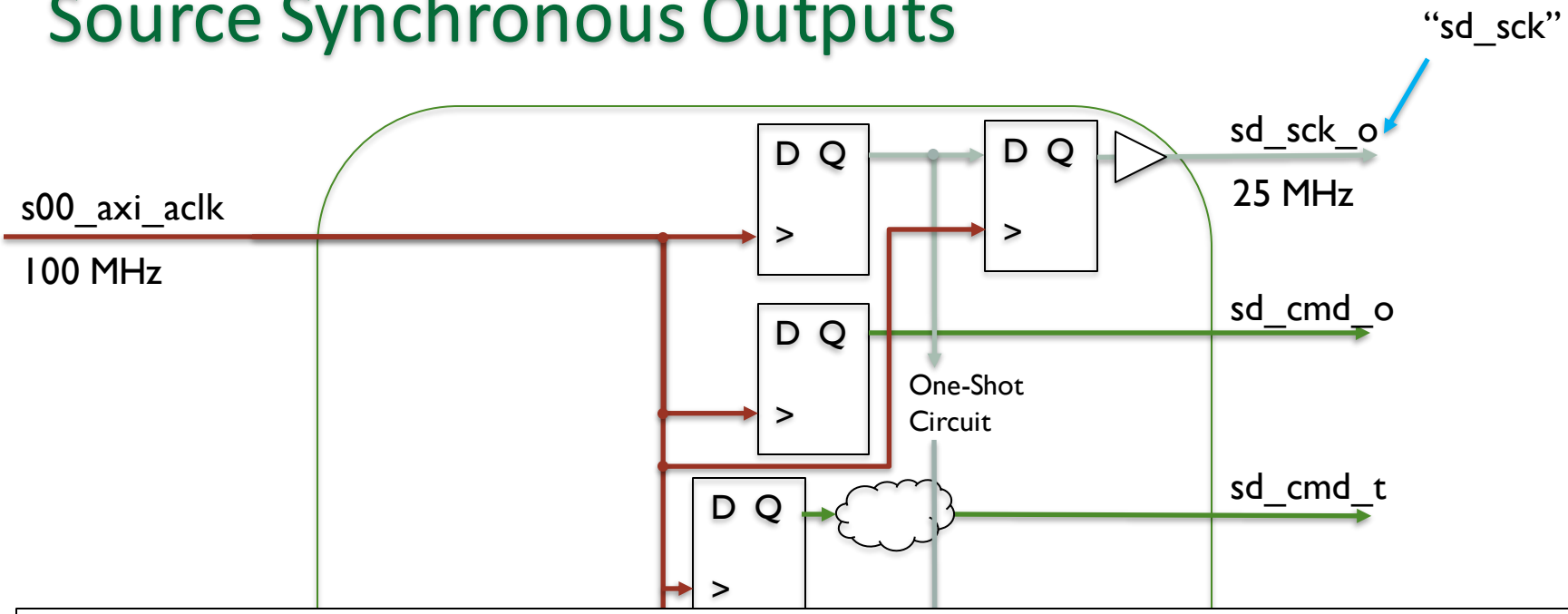



```
set_output_delay -clock sd_sck -max [expr $trce_dly_max + $tISU_min_rising -
$trce_dly_min] [get_ports sd_cmd_o -prop_thru_buffers]

set_output_delay -clock sd_sck -min [expr $trce_dly_min - ($trce_dly_max +
$tIH_min_rising)] [get_ports sd_cmd_o -prop_thru_buffers]

# The same for sd_cmd_t output
```

Timing Constraints – Command Outputs Source Synchronous Outputs




```
set_output_delay -clock sd_sck -max [expr 0.5 + 5 - 0] [get_ports sd_cmd_o -
prop_thru_buffers]

set_output_delay -clock sd_sck -min [expr 0 - (0.5 + 5)] [get_ports sd_cmd_o
-prop_thru_buffers]

# The same for sd_cmd_t output
```

Contents

- SD Card Interface Design
- Timing Paths
- Timing Constraints
- **Timing Analysis Results**
- References

Timing Analysis - Results

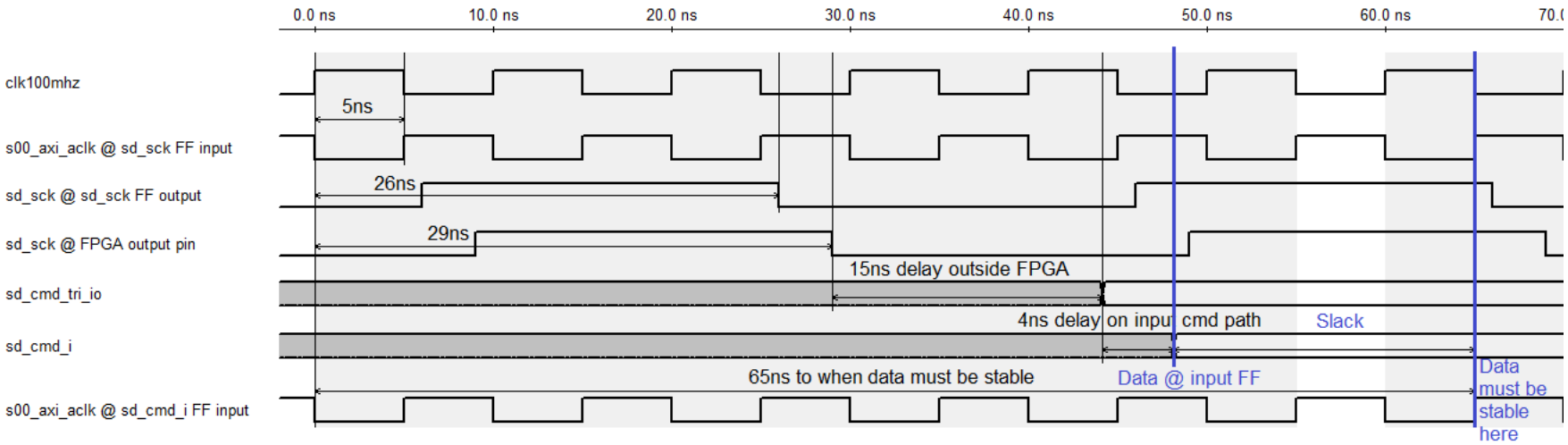
- TA results obtained with the 9 January 2020 Nexys A7 100T BIST
  - Using Vivado 2018.2

Timing Analysis - Results

- sd_cmd_o output worst-case slacks:
  - Setup Slack: 17.042 ns
  - Hold Slack: 0.479 ns – marginal, but OK
- sd_cmd_i input worst-case slacks:
  - Setup Slack: 16.475 ns
  - Hold Slack: 2.627 ns

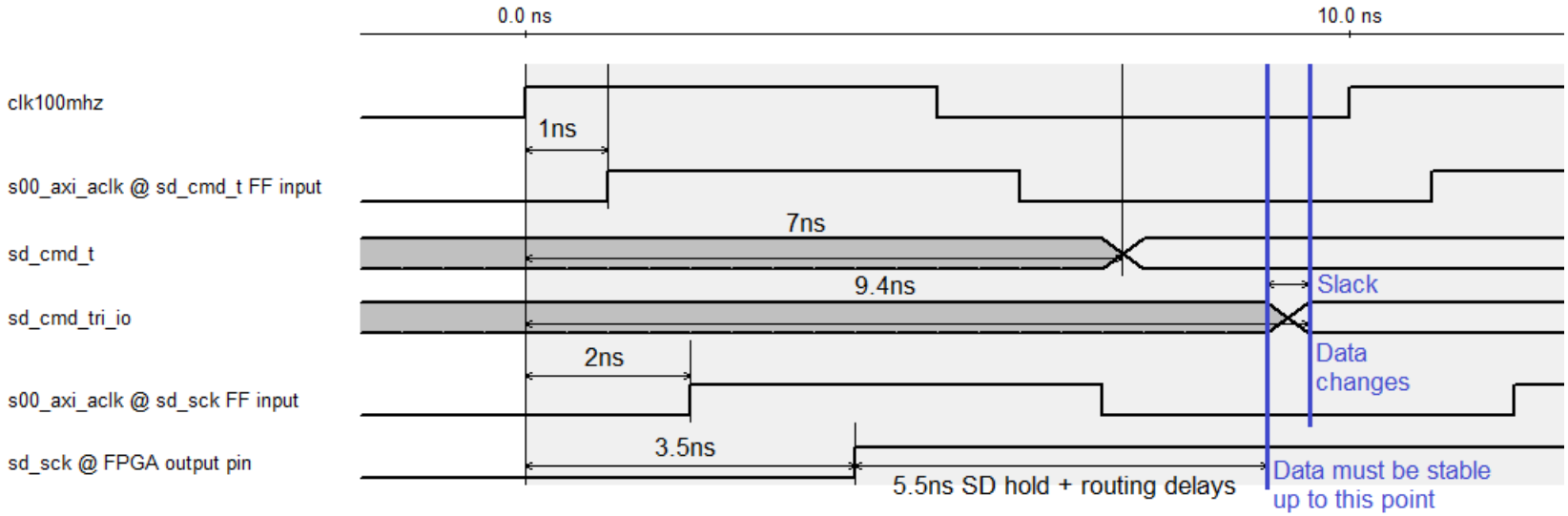
Timing Analysis – Input Setup Report

- Let's analyze a bit the sd_cmd_i input setup timing report



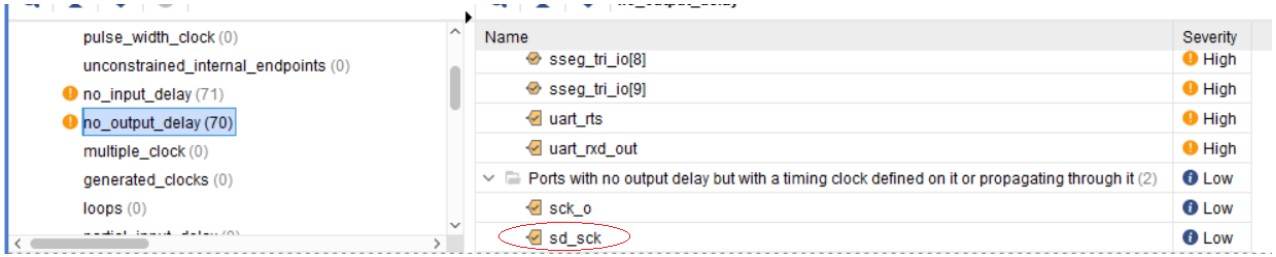
Timing Analysis – Output Hold Report

- Let's analyze a bit the sd_cmd_t output hold timing report



Good Results

- SD Card IP timing is met (on both 100T and 50T)
- No errors or regular warnings regarding the IP's timing
  - Just one critical warning (Low Severity): "Ports with no output delay but with a timing clock defined on it or propagating through it":



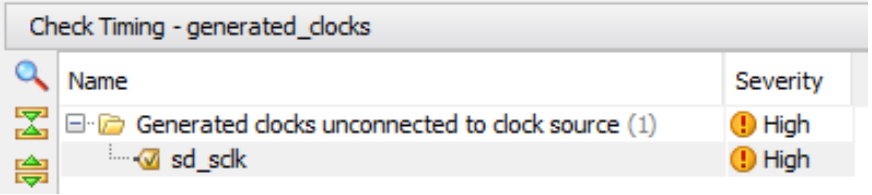
  - We could add a set of set_max_delay/set_min_delay constraints for it
- Tested the updated IP in hardware
  - No more data integrity issues

Vivado 2016.4 Possible Bug #1

- Clock relationship between fast_clk and sd_sck is not always correctly detected during timing analysis
  - The delay between them is not taken into account during analysis
- May be related to problem on next slide
- Problem does not appear in 2018.2

Vivado 2016.4 Possible Bug #2

- Unexplained warning appearing only in timing report



  - It appears only in complex project(s) like the SWORD BIST
  - It does not appear in a simpler test project using the same SD Card Interface IP
  - The clock source IS actually connected to the generated clock
  - The generated clock IS connected to an output pin
  - Does not appear in 2018.2

Contents

- SD Card Interface Design
- Timing Paths
- Timing Constraints
- Timing Analysis
- **References**

References

- Vivado Design Suite Tcl Command Reference Guide, UG835 (v2016.4) November 30, 2016, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug835-vivado-tcl-commands.pdf

- Vivado Design Suite User Guide - Using Constraints, UG903 (v2016.4) November 30, 2016, https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug903-vivado-using-constraints.pdf

- AXI Quad SPI v3.2, LogiCORE IP Product Guide, Vivado Design Suite, PG153 April 5, 2017, https://www.xilinx.com/support/documentation/ip_documentation/axi_quad_spi/v3_2/pg153-axi-quad-spi.pdf

Questions?