ADIUVO
ENGINEERING AND TRAINING, LTD.

Tackling Timing Analysis

Adam@adiuvoengineering.com

Objective

The objective of this session are:

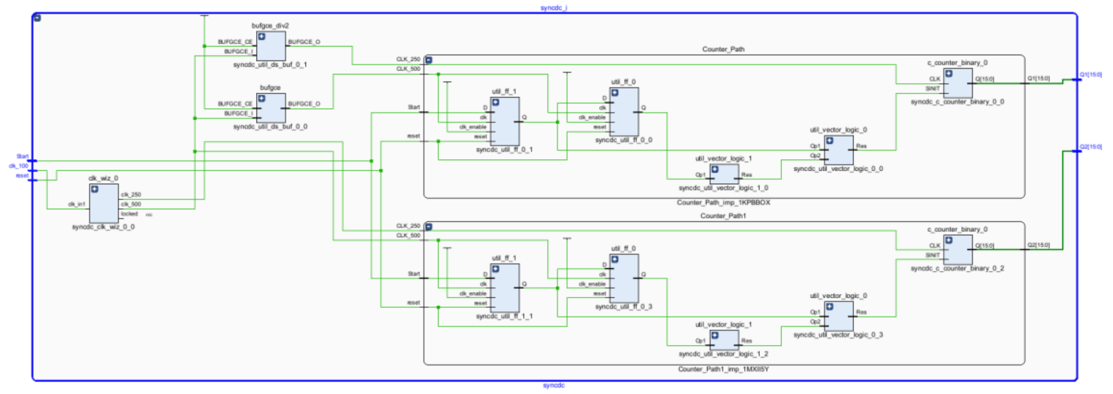
1. Introduce the concept of timing closure
2. Discuss clocking and clock domain crossing challenges
3. Introduce timing constraints
4. Demonstrate through a worked example who to achieve a baseline timing closure

What is timing closure

Designing the RTL correctly is just the start. To successfully deploy a design, we need to demonstrate it closes timing.

What is timing closure? Simple answer is when the design meets all timing constraints.

But defining the correct timing constraints and performing the analysis's can be complex and time consuming if not done correctly.



Objective of Timing closure

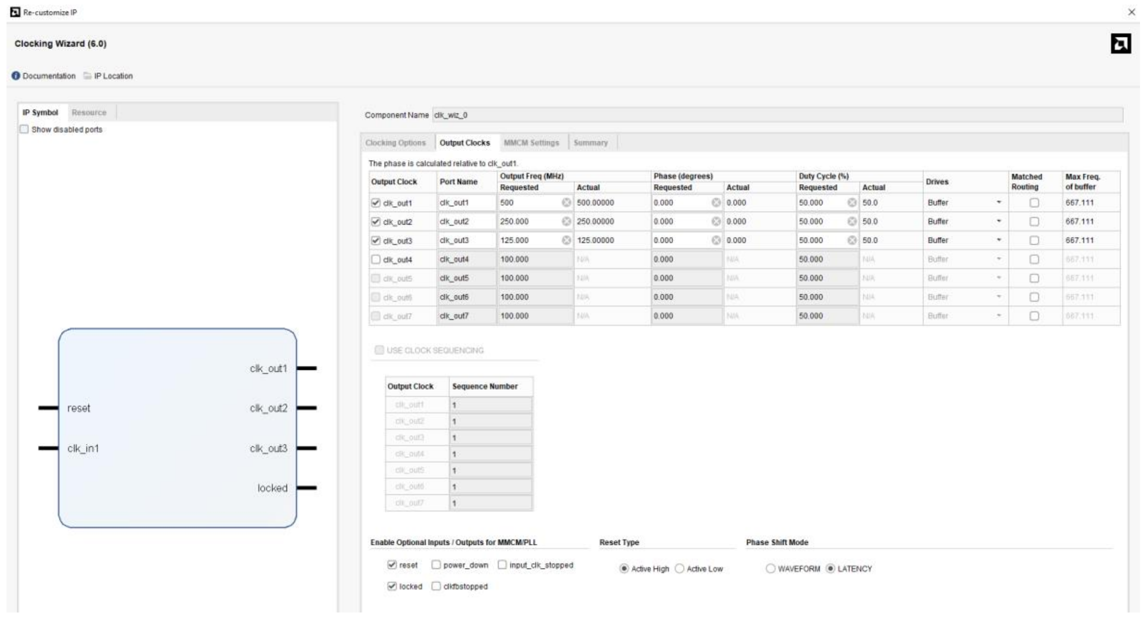
- All active clock pins are reached by a clock definition.
- All active path endpoints have requirement with respect to a defined clock (setup/hold/recovery/removal).
- All active input ports have an input delay constraint.
- All active output ports have an output delay constraint.
- Timing exceptions are correctly specified.

Timing Success

- Setup/Recovery (max delay analysis): WNS > 0 ns and TNS = 0 ns

- Hold/Removal (min delay analysis): WHS > 0 ns and THS = 0 ns

- Pulse Width: WPWS > 0 ns and TPWS = 0 ns


Re-customize IP

Clocking Wizard (6.0)

Documentation   IP Location

IP Symbol   Resource

☐ Show disabled ports

Component Name: clk_wiz_0

Clocking Options | Output Clocks | MMCM Settings | Summary

The phase is calculated relative to clk_out1.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Matched Routing	Max Freq. of buffer
		Requested	Actual	Requested	Actual	Requested	Actual			
✓ clk_out1	clk_out1	500	500.00000	0.000	0.000	50.000	50.0	Buffer	☐	667.111
✓ clk_out2	clk_out2	250.000	250.00000	0.000	0.000	50.000	50.0	Buffer	☐	667.111
✓ clk_out3	clk_out3	125.000	125.00000	0.000	0.000	50.000	50.0	Buffer	☐	667.111
☐ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	☐	667.111
☐ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	☐	667.111
☐ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	☐	667.111
☐ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	☐	667.111

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Enable Optional Inputs / Outputs for MMCM/PLL: ✓ reset ☐ power_down ☐ input_clk_stopped ✓ locked ☐ clkfbstopped

Reset Type: [✓ Active High] Active Low

Phase Shift Mode: WAVEFORM [✓ LATENCY]

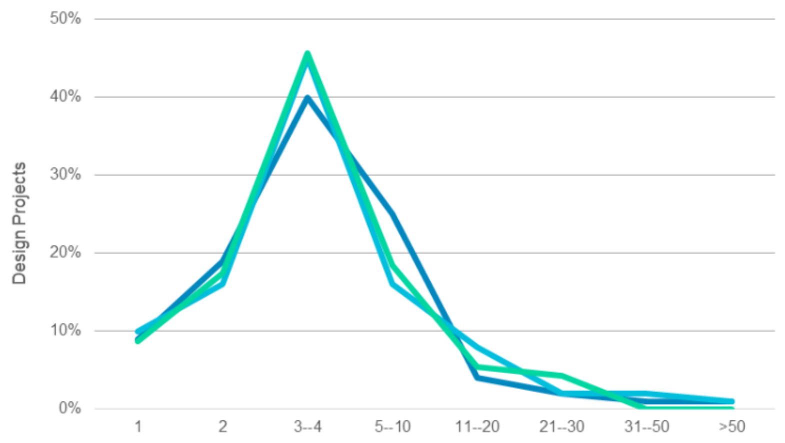
Challenge for Most Projects

- Wilson Group FPGA survey runs every 2 years.
- Latest results in 2022 show average number of clocks 3 -4
- Considering use of standards like AXI, AXI Lite etc. it is easy to achieve several clocks

Number of Asynchronous Clock Domain on FPGA Projects

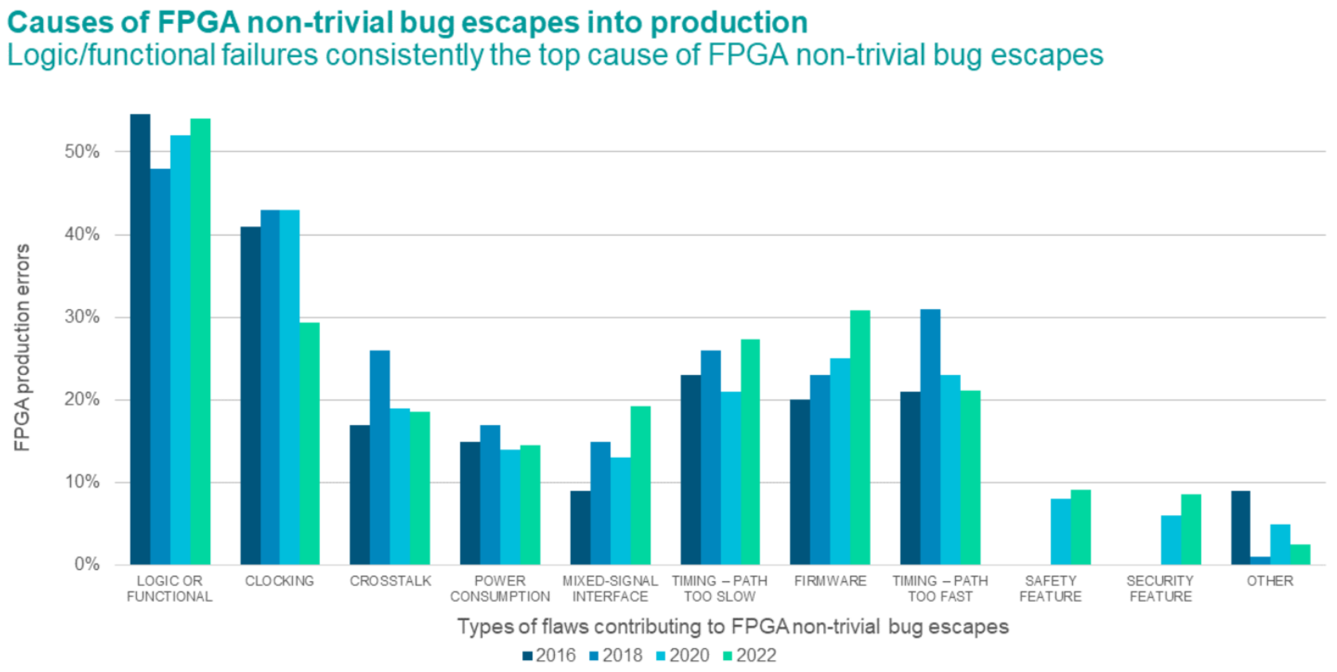
3 - 4

Average number of clock domains on FPGA designs



Impacts on project

- 70% of FPGA projects behind schedule
- 84% have a non-trivial bug arise in production
- Second leading cause of issue is timing / clocking related



Clocking

ADIUVO
ENGINEERING AND TRAINING, LTD.

Clocking

FPGA Designs are Synchronous!

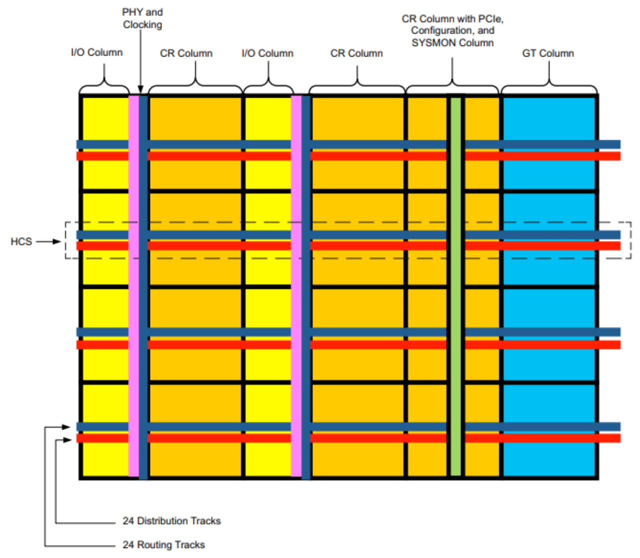
Clocks are high fan out, as such dedicated pins are required to be used.

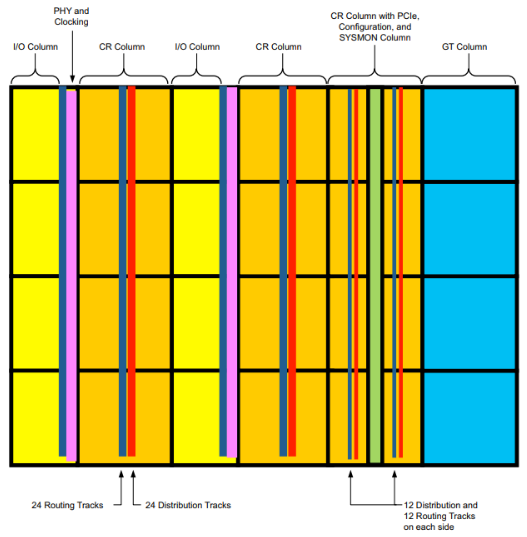
FPGA typically will have multiple clock regions

- Each Region contains CLBs, DSP and BRAM

Clock pins are places in IO banks

- UltraScale / UltraScale+ GC pins or global clock pins
- Seven Series CC and GC pins – GC global clocks CC restricted to close CR
- Special Clock pins – e.g., Byte-Lane Clocks (DBC and QBC) typical in memory applications





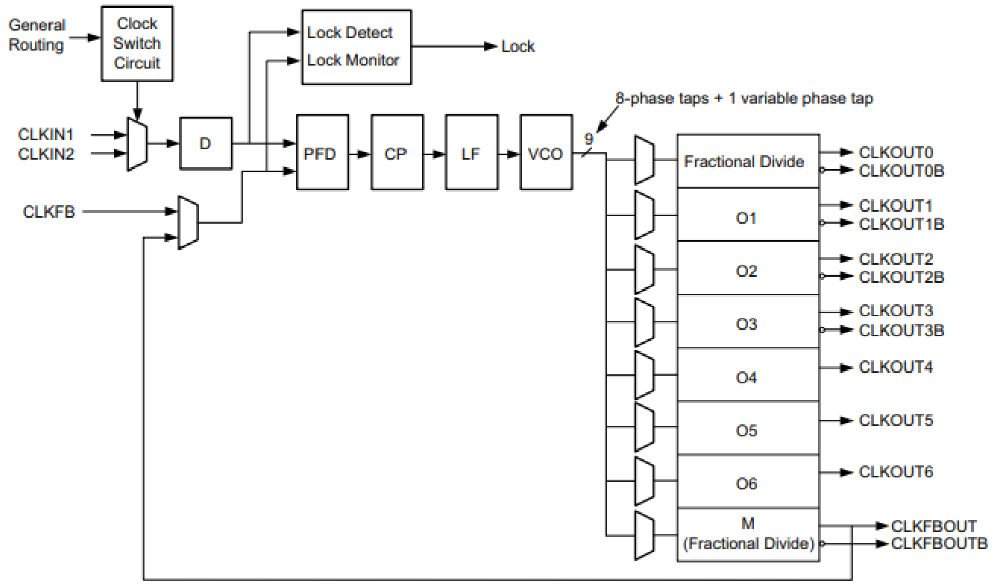
PHY and Clocking
I/O Column
CR Column
I/O Column
CR Column
CR Column with PCIe, Configuration, and SYSMON Column
GT Column
24 Routing Tracks
24 Distribution Tracks
12 Distribution and 12 Routing Tracks on each side

Clocking Resources

FPGA have a range of clock resources to simply solutions

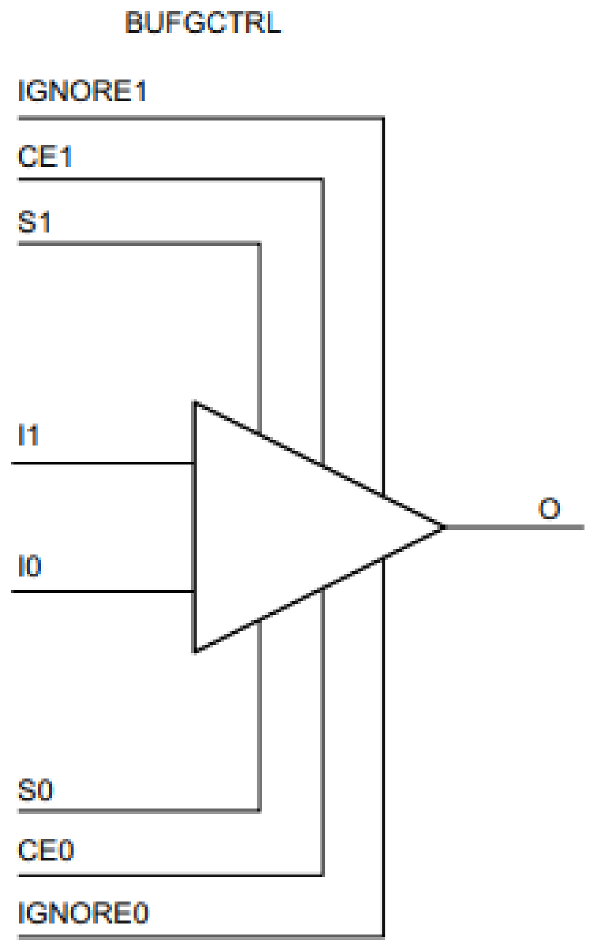
1. Clock Buffers (e.g., bufgctl)

2. Clock Management Tiles
   - Mixed Mode Clock Manager (MMCM) & 2 Phase Locked Loops



Clocking Resources

- Bufgctl - drive the routing and distribution resources across the entire device.
- Can switch between 2 clocks seamlessly.
- Basis of many structures including
- Bufgce_1, Bufgmux etc.
- Read Seven Series, UltraScale clocking guides



Constraints

What are constraints?

Constraints allow you to relay additional information about your specific design and its implementation to the synthesis and implementation tools.

Constraints can be split into two categories: those used during synthesis and implementation—timing constraints, for instance—and those used exclusively for implementation—pin placement, for example.

The constraints we will use within Vivado are based upon the industry-standard SDC or Synopsis Design Constraints format with additional AMD-specific constraints to specify implementation details. Because this is an AMD specific constraint’s file, the file prefix is XDC and not SDC

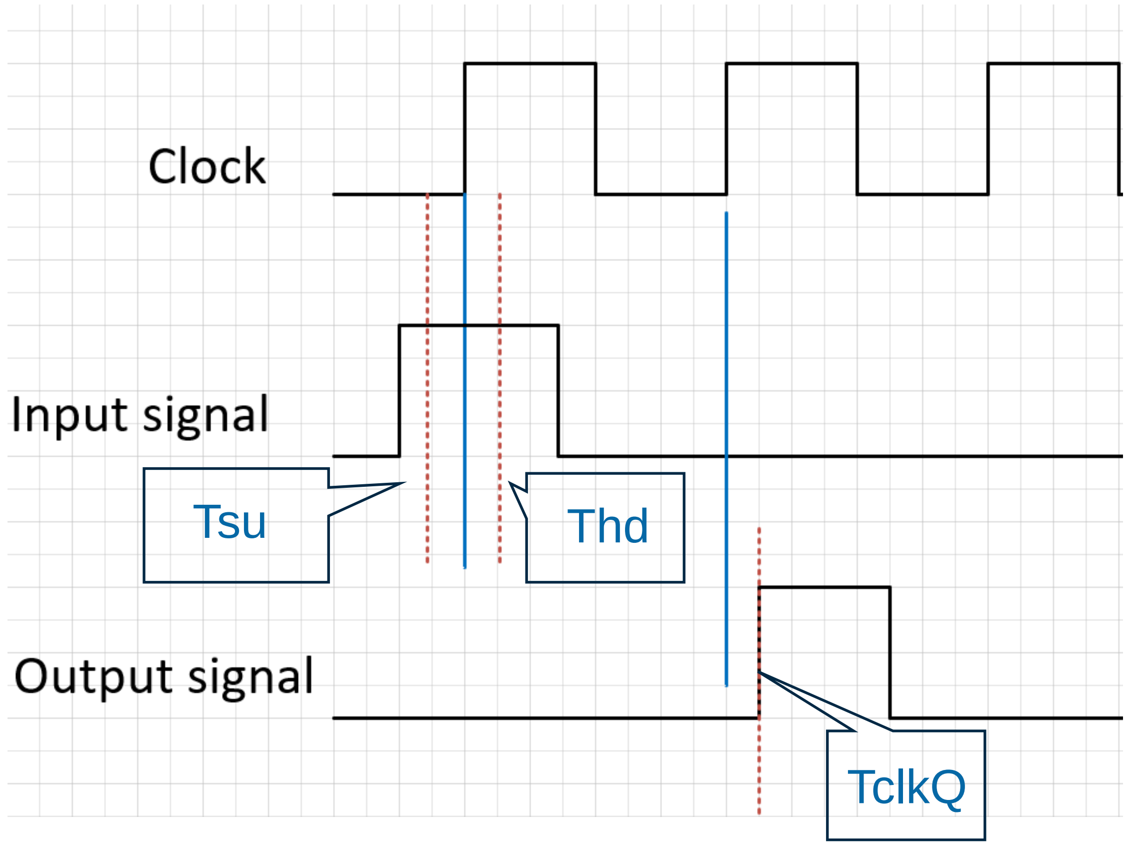
Constraints

When we declare our constraints, we must do so in the following order:

- **Timing constraints** – The timing relationships required for correct operation
- **Timing Exceptions** – Having first defined the timing relationships, we then define any exceptions to those constraints, if these exceptions exist (e.g., for multi cycle paths). Note that we cannot define an exception unless we first describe what the norm is.
- **Implementation constraints** – Constraints used in the design's placement and routing. These constraints help to achieve the desired results (e.g., pin out and location constraints).

Clock Constraints

- Clock provides a regular timing reference.
- Key aspects we need to understand are
  - Tsu – Set up time, the arrival time of the signal prior to the clock edge.
  - Thd – Hold time, the time the signal remains stable after the clock edge.
  - TclkQ -Clock to output, the time it takes following a clock edge for the output to change.



Clock Constraints

Vivado encompasses three concepts for clocks, which are defined slightly differently:

- **Primary Clocks** – those that enter through an I/O pin
- **Generated Clocks** – those which are generated automatically via an internal PLL or by the design (for instance, dividing a clock by two with a flip-flop). With generated clocks, one describes how the master clock (either a prime or other generated clock) modifies the waveform
- **Virtual Clocks** – These are not attached to anything within the design netlist but can be used for I/O timing

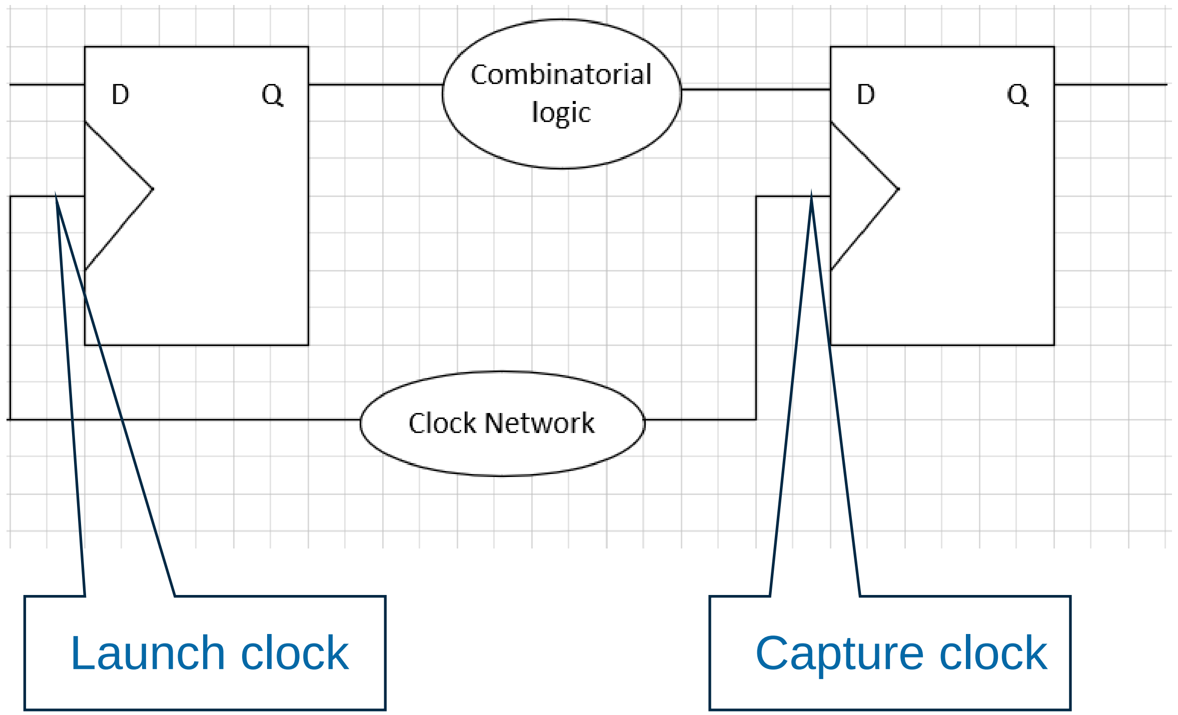
Primary and virtual clocks are created using the command:

create_clock –period –name –waveform [source objects]

Clock Constraints

Common Definitions used in clock constraints

- The launch clock is associated with the source that launches the data.
- The capture clock is associated with the destination flip-flop that captures the data. The set-up requirement is the clock's capture edge time minus the launch edge time.
- Note that these clocks can be different—they can have different phases or different clock periods.
- If different launch and capture clocks are used, then the set-up requirement is the smaller of the two when you consider the worst-case relationships between the two clocks.



Clock Constraints

Hold time is the time that the data must be stable and unchanging after the capture clock edge. The hold time is the greater of the current capture edge minus the next launch edge. If the launch and capture clocks are different, then worst-case timing between clocks becomes an important consideration.

Vivado analyzes both the set-up and hold times and provides the results of these analysis. The resulting report shows the design slack. Positive slack means that the register will not clock in a wrong value or go metastable. Negative slack means metastability is an issue because of a timing violation.

Clock Constraints

Vivado defines clocks as being within one of three categories:

- **Synchronous Clocks** – Synchronous clocks have a predictable timing/phase relationship, which is normally the case for a primary clock and its generated clocks because they share a common root clock and will therefore have a common period.

- **Asynchronous Clocks** – Asynchronous clocks have no predictable timing/phase relationship, which is normally the case for different primary clocks (and the clocks generated from these primary clocks). Asynchronous clocks have different roots.

- **Unexpandable Clocks** – Two clocks are unexpandable if a common period cannot be determined over 1000 clock cycles. If a common clock period cannot be established, then Vivado uses the worst case set-up relationship over the 1000 cycles. However, there is no guarantee that this relationship truly represents the actual worst case. That estimate is just the best that Vivado can do with the information provided.

Clock Domain Crossing

ADIUVO
ENGINEERING AND TRAINING, LTD.

What is a Clock Domain?

Same clock domain if:

- Same source and integer multiple frequencies

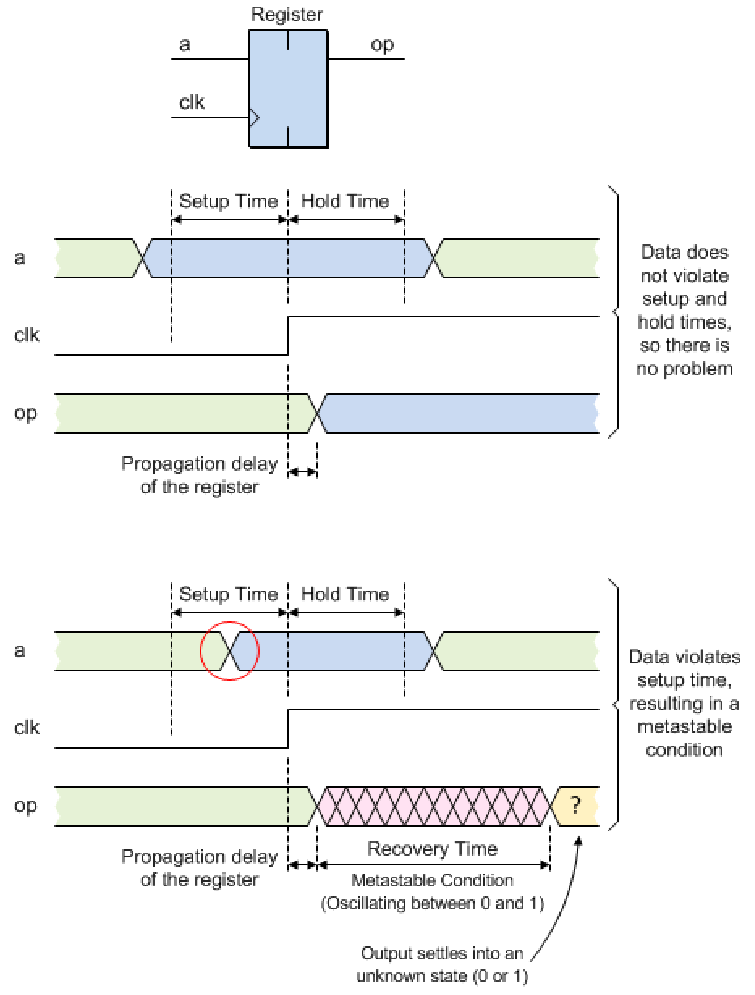
Different clock domain if:

- Different source
  - **Even if frequency specification is the same!**
  - **All specifications have error bars**
- Not integer multiple frequencies, even if same source

- By default, Vivado assumes all clocks are in the same domain – Unless you tell it otherwise! Not telling it clock relationships can make a significantly longer implementation times as Vivado tries to close timing which is impossible

Metastability

- One issue which can arise with incorrect domain crossing is metastability
- This can lead to corruption of data or incorrect behaviour
- Occurs when a flip flops set up or hold time is violated


Register
a
op
clk
Setup Time
Hold Time
a
clk
op
Data does not violate setup and hold times, so there is no problem
Propagation delay of the register
Setup Time
Hold Time
a
clk
op
?
Data violates setup time, resulting in a metastable condition
Propagation delay of the register
Recovery Time
Metastable Condition (Oscillating between 0 and 1)
Output settles into an unknown state (0 or 1)

Clock Domain Crossing

Several Techniques which can be used depending upon what needs to be transferred

- Two stage synchroniser – Ideal for single bit data
- Grey Code Synchroniser – Encodes data bus in grey code and transfer between domains – Ideal for counters as input to be converted to grey code can only decrement / increment by one from previous value
- Hand shake synchroniser – Transfers data bus between two clock domains using handshake signals
- Pulse synchroniser – transfer pulse from one clock domain to another
- Asynchronous FIFO – transfers data from one domain to another, useful for high throughput / burst transfers

Clock Domain Crossing

To support reset functionality across clock domains we may need the following synchronisers structures.

- Asynchronous Reset Synchroniser – enables asynchronous assertion and synchronous de-assertion.
- Synchronous Reset Synchroniser – synchronises a synchronous reset to another clock domain.

ADIUVO
ENGINEERING AND TRAINING, LTD.

CDC in AMD

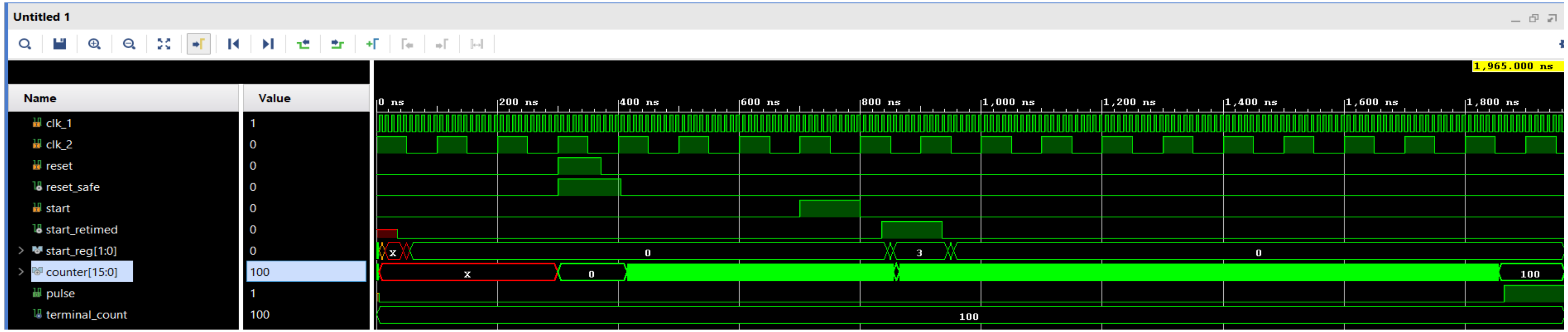
AMD Parameterised Macros (XPM) – provide CDC structures

Use registers optimised for CDC in the fabric

- Registers located close together and have small set up and hold windows

Described within UG953 Libraries Guide

Described within UG 974 UltraScale Architecture Libraries


Untitled 1

Name	Value
clk_1	1
clk_2	0
reset	0
reset_safe	0
start	0
start_retimed	0
start_reg[1:0]	0
counter[15:0]	100
pulse	1
terminal_count	100

Timing Closure Approach

Baseline

Objective – Achieve the simplest set of timing constraints

Post Synthesis

- Use Timing wizard to define constraints – Skip IO constraints
- Validate with report methodology

Optimize Design + Report Timing Summary

- Resolve WNS
- Validate with report methodology

Place Design + Report Timing Summary

- Resolve WNS
- Resolve Large Hold WHS > -0.5ns

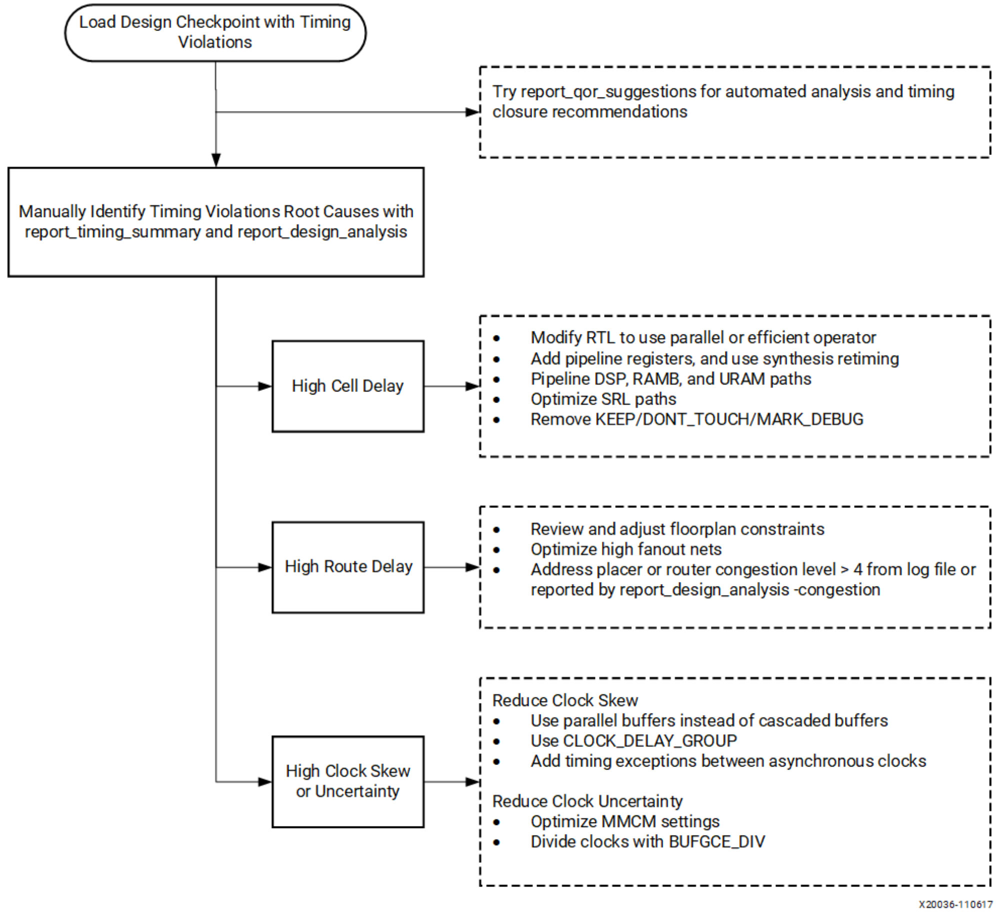
Route Design

- Resolve all timing violations
- Validate with methodology and DRC Checks

Analyzing Timing Violations

As we go through these stages, we need some guidance and pointers

- Before routing, review only violations over 0.5 ns
- After routing, start with the worst violation.



Load Design Checkpoint with Timing Violations
Try report_qor_suggestions for automated analysis and timing closure recommendations
Manually Identify Timing Violations Root Causes with report_timing_summary and report_design_analysis
High Cell Delay
• Modify RTL to use parallel or efficient operator
• Add pipeline registers, and use synthesis retiming
• Pipeline DSP, RAMB, and URAM paths
• Optimize SRL paths
• Remove KEEP/DONT_TOUCH/MARK_DEBUG
High Route Delay
• Review and adjust floorplan constraints
• Optimize high fanout nets
• Address placer or router congestion level > 4 from log file or reported by report_design_analysis -congestion
High Clock Skew or Uncertainty
Reduce Clock Skew
• Use parallel buffers instead of cascaded buffers
• Use CLOCK_DELAY_GROUP
• Add timing exceptions between asynchronous clocks
Reduce Clock Uncertainty
• Optimize MMCM settings
• Divide clocks with BUFGCE_DIV
X20036-110617

Leverage Reports

Vivado can provide several reports which can be very useful

- Quality of Result Analysis - An assessment score that is indicative of how likely your design is to meet performance targets. Also provides flow guidance.

- Quality of Result Suggestions - Identifies issues with a design and offers solutions in the form of tool switches, properties that influence tool behavior on cells, and recommends text modifications where it is not possible to automate a solution

- Design Analysis Report - Information on timing path characteristics, design interconnect complexity, and congestion

Design Analysis Report

Logic Delay

- Are there many levels of logic?
- Are there any constraints or attributes that prevent logic optimization?
- Does the path include a cell with high logic delay such as block RAM or DSP?
- Is the path requirement too tight for the current path topology?

Net Delay

- Are there any high fanout nets in the path?
- Are the cells assigned to several Pblocks that can be placed far apart?
- Are the cells placed far apart
- Is there a missing pipeline register in a block RAM or DSP cell?

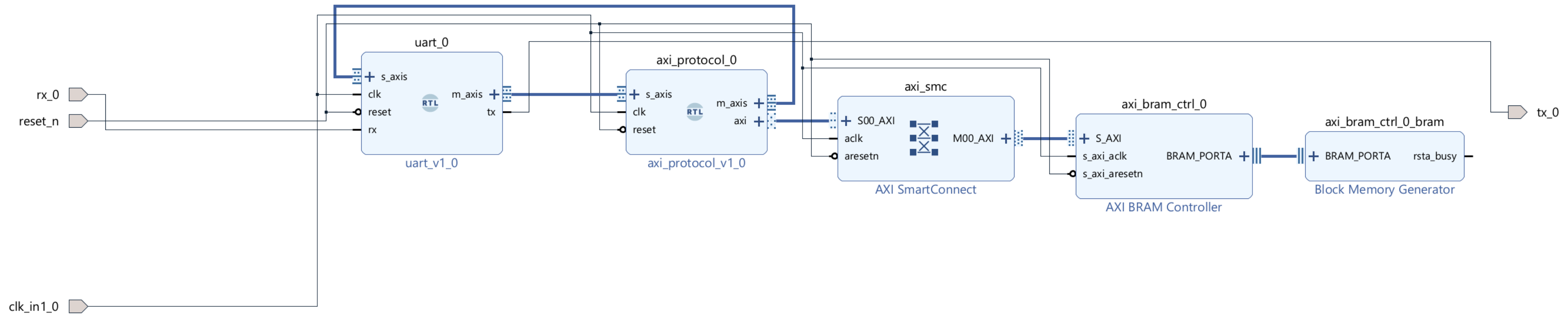
High Skew

- Is it a clock domain crossing path?
- Are the clocks synchronous or asynchronous?
- Is the path crossing I/O columns?

Timing Closure Example

Example Design

Provides UART access to AXI Network using AXI Stream, Protocol Decoding and then AXI Peripherals to complete the design.



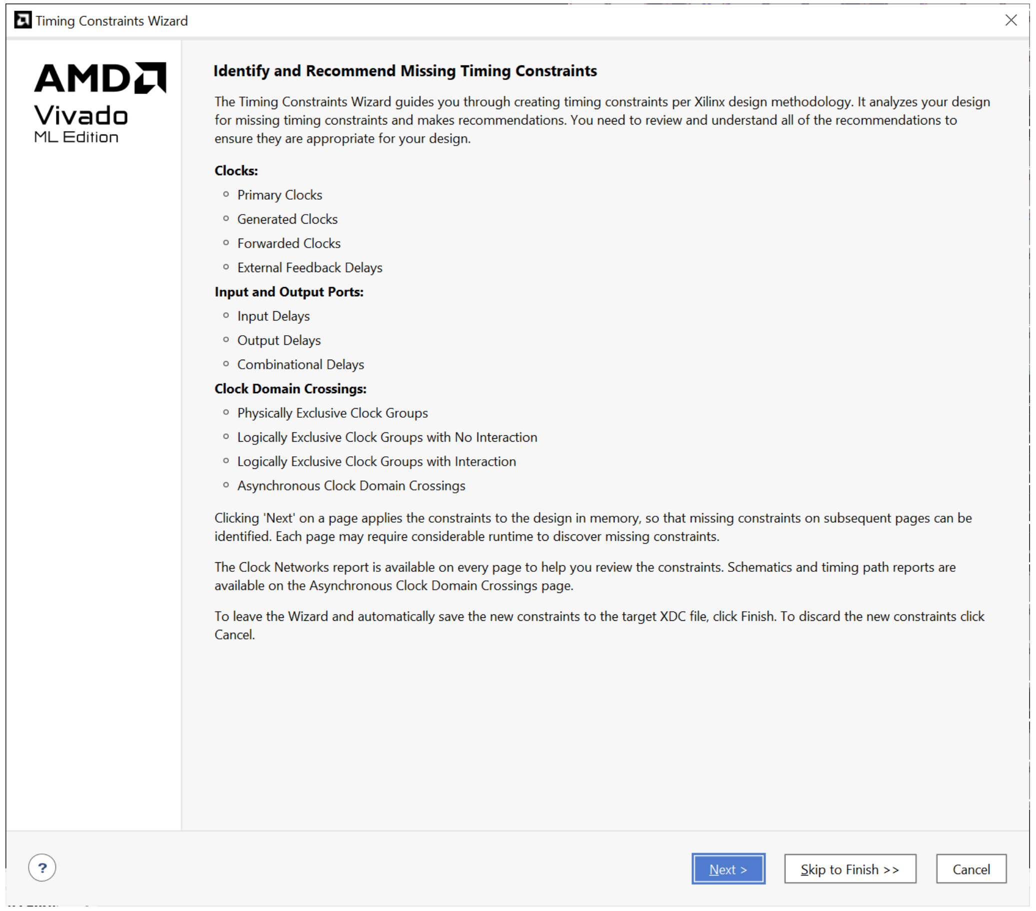
Example Design

Based on a Hackster project.

Original project ran at 100MHz

For this example, we are going to make it run at 200MHz in an Arty XC7S25 device.

Starting point is synthesising design and then opening the synthesis view.



Identify and Recommend Missing Timing Constraints

The Timing Constraints Wizard guides you through creating timing constraints per Xilinx design methodology. It analyzes your design for missing timing constraints and makes recommendations. You need to review and understand all of the recommendations to ensure they are appropriate for your design.

Clocks:
- Primary Clocks
- Generated Clocks
- Forwarded Clocks
- External Feedback Delays

Input and Output Ports:
- Input Delays
- Output Delays
- Combinational Delays

Clock Domain Crossings:
- Physically Exclusive Clock Groups
- Logically Exclusive Clock Groups with No Interaction
- Logically Exclusive Clock Groups with Interaction
- Asynchronous Clock Domain Crossings

Clicking 'Next' on a page applies the constraints to the design in memory, so that missing constraints on subsequent pages can be identified. Each page may require considerable runtime to discover missing constraints.

The Clock Networks report is available on every page to help you review the constraints. Schematics and timing path reports are available on the Asynchronous Clock Domain Crossings page.

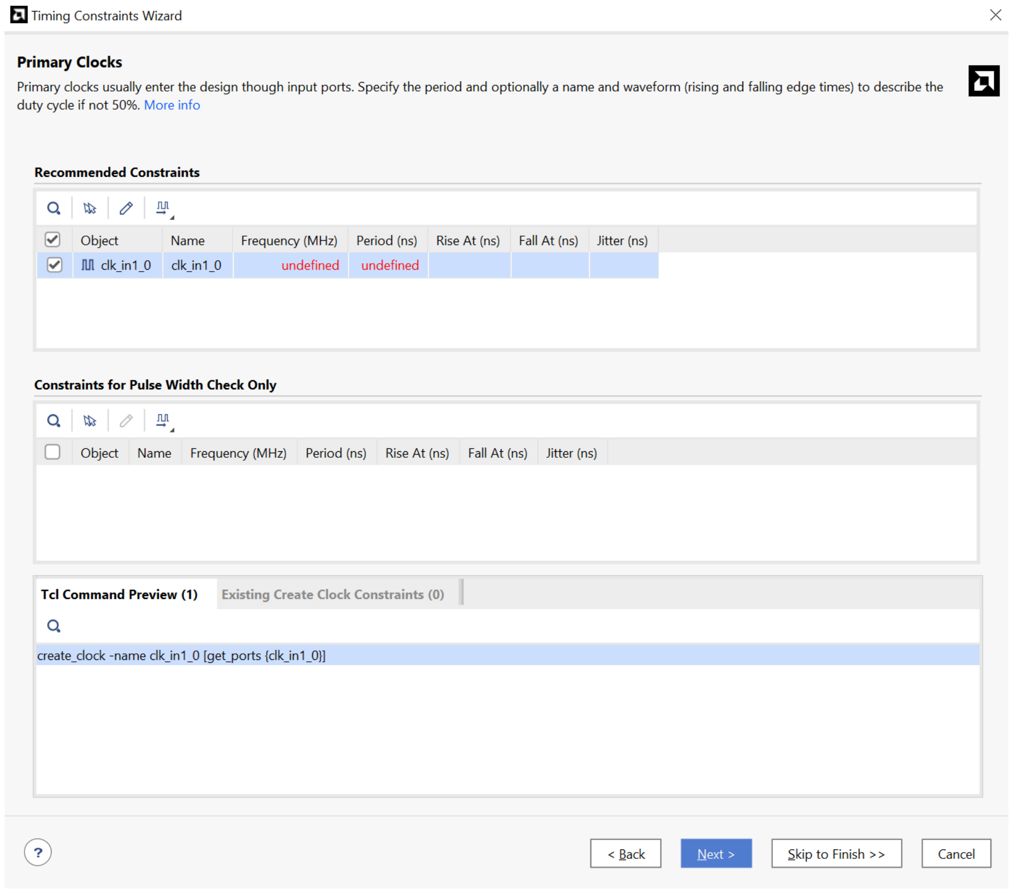
To leave the Wizard and automatically save the new constraints to the target XDC file, click Finish. To discard the new constraints click Cancel.

Create Initial Timing Constraints

Use Timing Constraints Wizard to create the initial constraints

Target Frequency is 200 Mhz

As we are creating baseline skip to the end once completed



Primary Clocks

Primary clocks usually enter the design though input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. More info

Recommended Constraints

☑	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
☑	clk_in1_0	clk_in1_0	undefined	undefined			

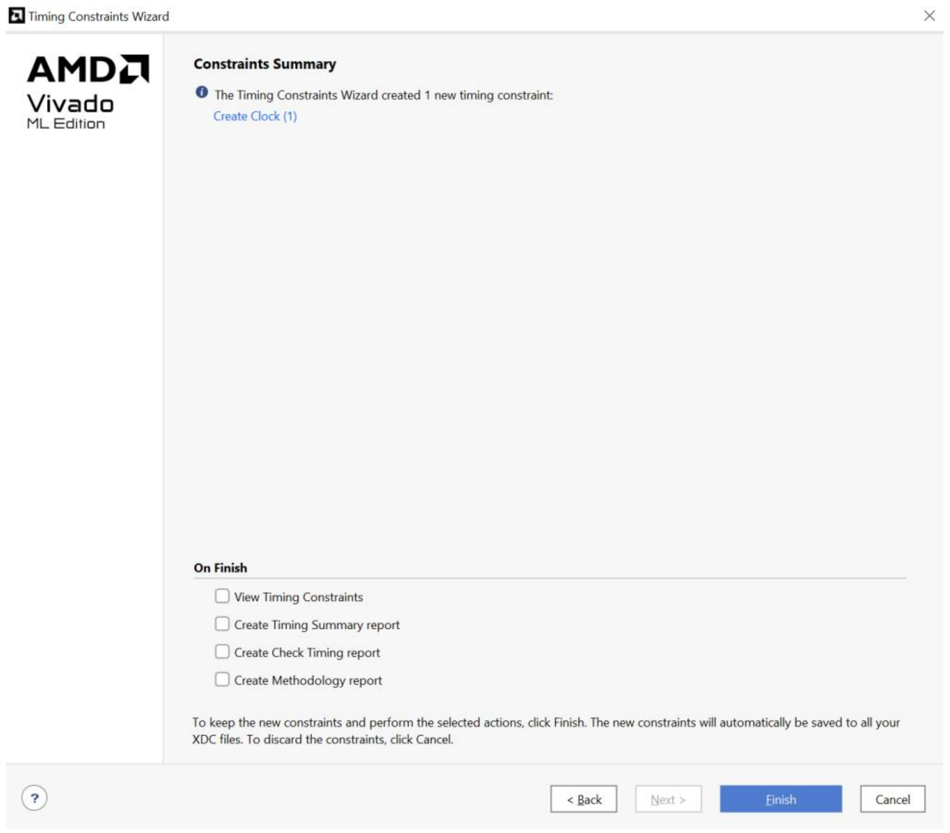
Constraints for Pulse Width Check Only

☐	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)

```
create_clock -name clk_in1_0 [get_ports {clk_in1_0}]
```

Creating timing constraints

Finish the constraints wizard and if you desire examine the timing constraints which have been created in the XDC file.



Constraints Summary

The Timing Constraints Wizard created 1 new timing constraint:
Create Clock (1)

On Finish

- [ ] View Timing Constraints
- [ ] Create Timing Summary report
- [ ] Create Check Timing report
- [ ] Create Methodology report

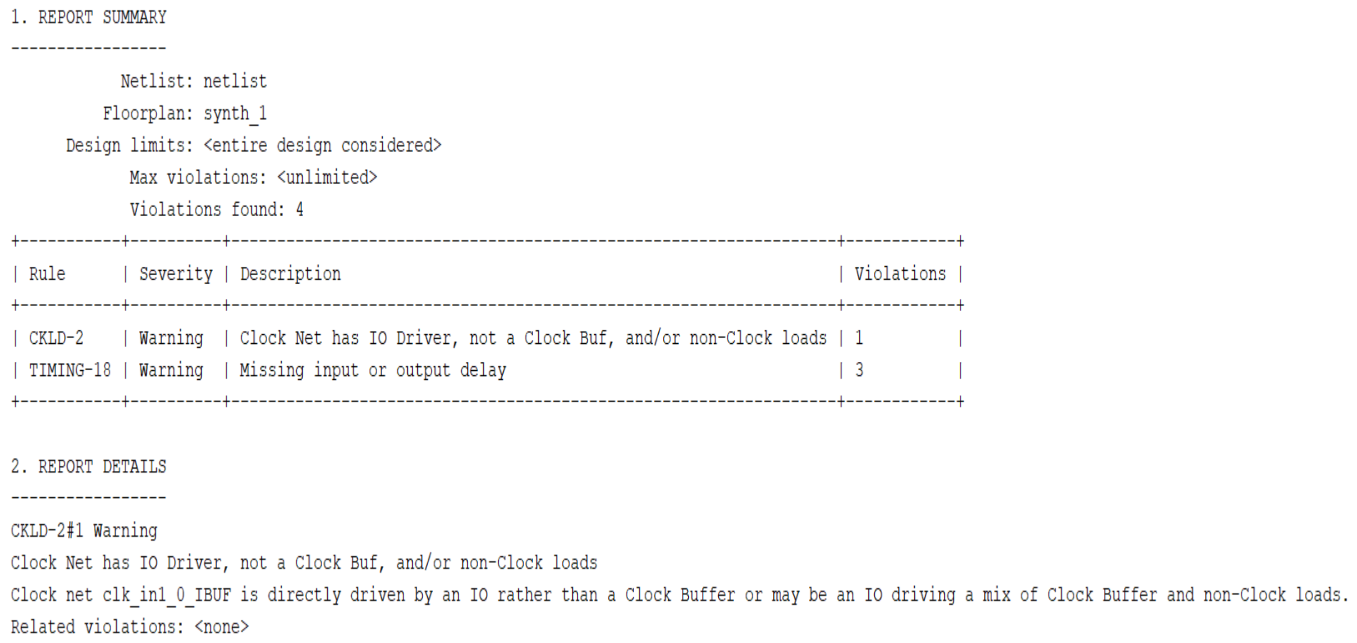
To keep the new constraints and perform the selected actions, click Finish. The new constraints will automatically be saved to all your XDC files. To discard the constraints, click Cancel.

Report Methodolody

Run Synthesis once this completes.

Run the methodology report.

This report flags an initial problem that the clock signal is not assigned to a global buffer



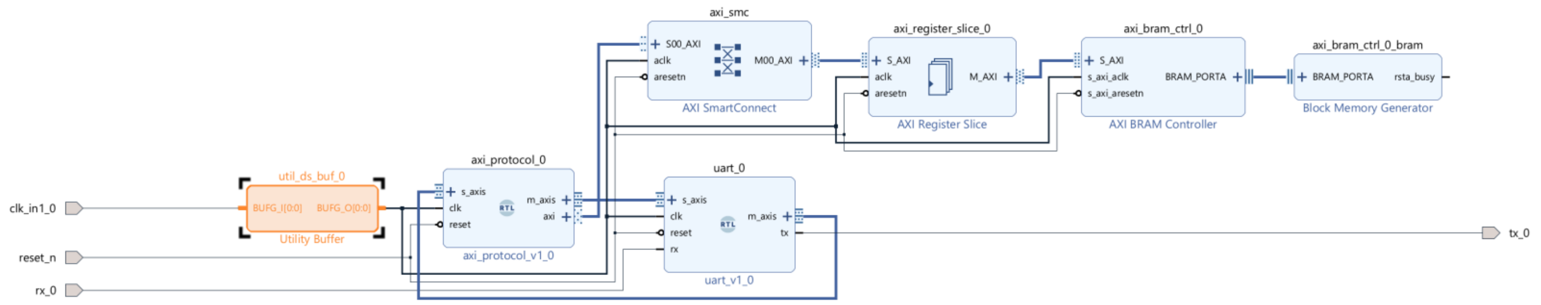
```
1. REPORT SUMMARY
-----------------
            Netlist: netlist
          Floorplan: synth_1
      Design limits: <entire design considered>
             Max violations: <unlimited>
             Violations found: 4
+-----------+----------+--------------------------------------------------------------------+------------+
| Rule      | Severity | Description                                                        | Violations |
+-----------+----------+--------------------------------------------------------------------+------------+
| CKLD-2    | Warning  | Clock Net has IO Driver, not a Clock Buf, and/or non-Clock loads | 1          |
| TIMING-18 | Warning  | Missing input or output delay                                      | 3          |
+-----------+----------+--------------------------------------------------------------------+------------+

2. REPORT DETAILS
-----------------
CKLD-2#1 Warning
Clock Net has IO Driver, not a Clock Buf, and/or non-Clock loads
Clock net clk_in1_0_IBUF is directly driven by an IO rather than a Clock Buffer or may be an IO driving a mix of Clock Buffer and non-Clock loads.
Related violations: <none>
```

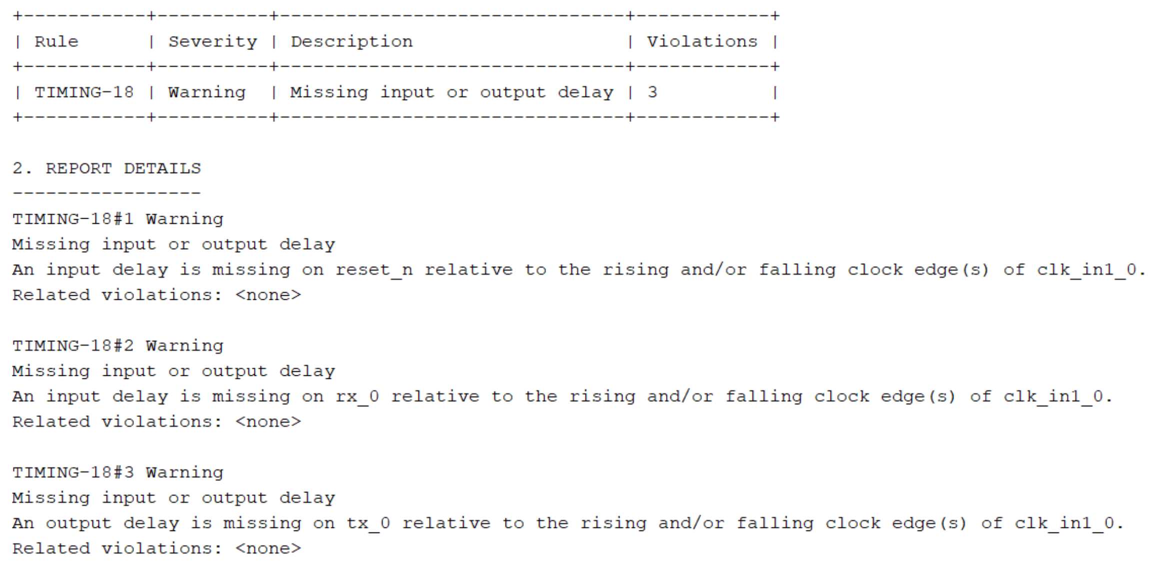
Reporting Methodology

We need to update the design to include a BUFG for the clocking.

I chose to do this simply in IP integrator. Rather than updating the code.

Rerun synthesis to update design and report methodology again



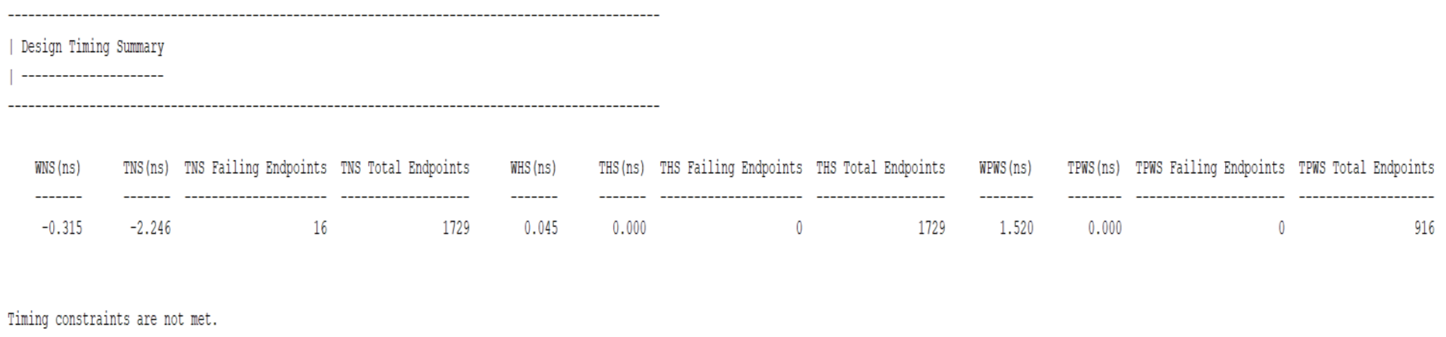

```
+-----------+----------+--------------------------------+------------+
| Rule      | Severity | Description                    | Violations |
+-----------+----------+--------------------------------+------------+
| TIMING-18 | Warning  | Missing input or output delay | 3          |
+-----------+----------+--------------------------------+------------+

2. REPORT DETAILS
-----------------
TIMING-18#1 Warning
Missing input or output delay
An input delay is missing on reset_n relative to the rising and/or falling clock edge(s) of clk_in1_0.
Related violations: <none>

TIMING-18#2 Warning
Missing input or output delay
An input delay is missing on rx_0 relative to the rising and/or falling clock edge(s) of clk_in1_0.
Related violations: <none>

TIMING-18#3 Warning
Missing input or output delay
An output delay is missing on tx_0 relative to the rising and/or falling clock edge(s) of clk_in1_0.
Related violations: <none>
```

Optimize Design

Optimize the design using the opt_design command in tcl console

Once completed run the timing report summary. This will indicate there are likely to be timing issues.

Remember at this stage routing is estimation

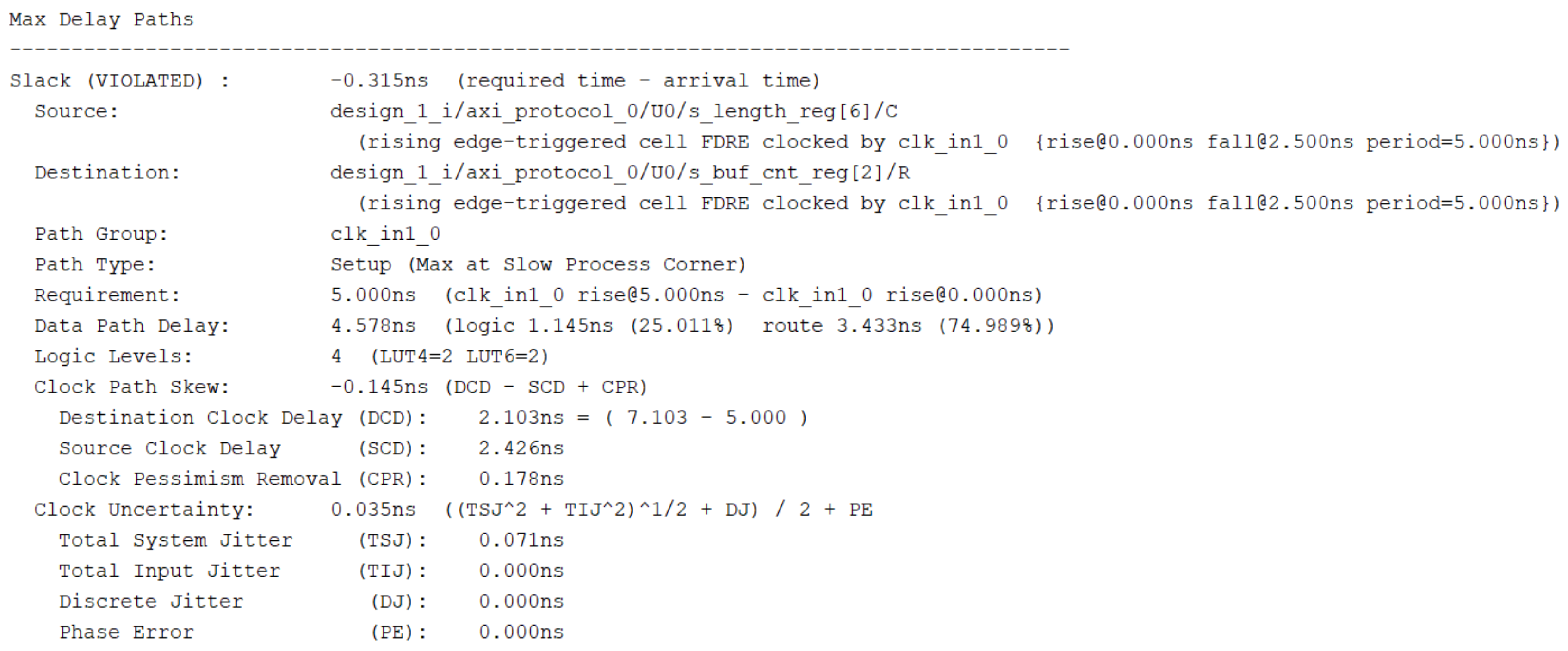


```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)     TPWS(ns)  TPWS Failing Endpoints  TPWS Total Endpoints
    -------      -------  ---------------------  -------------------      -------      -------  ---------------------  -------------------     --------     --------  ----------------------  --------------------
     -0.315       -2.246                     16                 1729        0.045        0.000                      0                 1729        1.520        0.000                       0                   916


Timing constraints are not met.
```

Optimize Design

Run the timing report post optimization this will list several failing nets.

Associated with the Smart Connect and the one of the RTL blocks.



```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (VIOLATED) :        -0.315ns  (required time - arrival time)
  Source:                 design_1_i/axi_protocol_0/U0/s_length_reg[6]/C
                            (rising edge-triggered cell FDRE clocked by clk_in1_0  {rise@0.000ns fall@2.500ns period=5.000ns})
  Destination:            design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[2]/R
                            (rising edge-triggered cell FDRE clocked by clk_in1_0  {rise@0.000ns fall@2.500ns period=5.000ns})
  Path Group:             clk_in1_0
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            5.000ns  (clk_in1_0 rise@5.000ns - clk_in1_0 rise@0.000ns)
  Data Path Delay:        4.578ns  (logic 1.145ns (25.011%)  route 3.433ns (74.989%))
  Logic Levels:           4  (LUT4=2 LUT6=2)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.103ns = ( 7.103 - 5.000 )
    Source Clock Delay      (SCD):    2.426ns
    Clock Pessimism Removal (CPR):    0.178ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

How do we address these?

Estimated net delays are close to the best possible placement for all paths.

- Change the RTL.

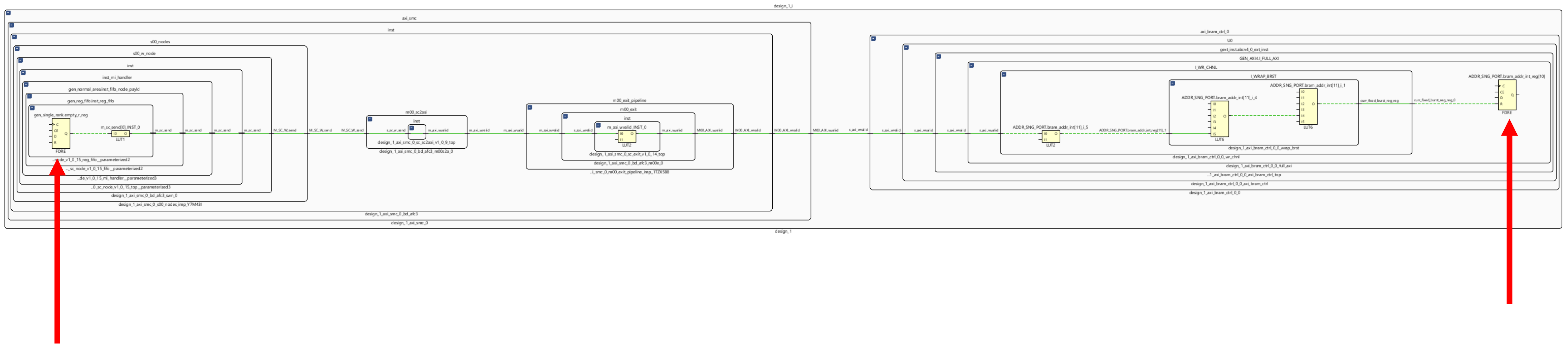
- Use different synthesis options.

- Add timing exceptions such as multicycle paths, if appropriate and safe for the functionality in hardware.

Smart Connect

Selecting a failing path on the smart connect will enable us to see the schematic design - This provides context as to the potential issues.

In this case it can be seen as a long delay and routing between the registers, in two blocks.



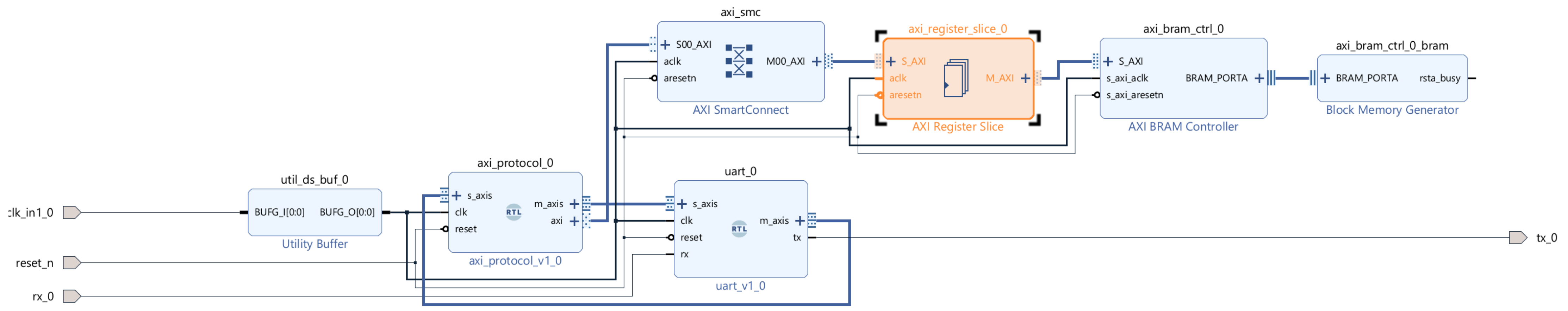
Output Register in Smart Connect

Input Register in AXI Bram Control

Smart Connect

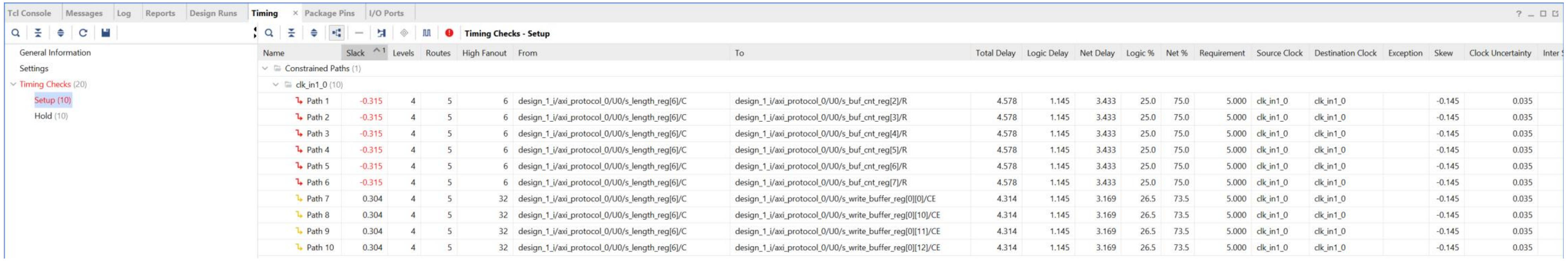
Solution in this case is to insert a register slice between the Smart Connect and AXI Bram Controller.

This inserts an additional register in the combinatorial path.



Smart Connect

Rerunning the synthesis and timing report will show the errors associated with the Smart Connect have been addressed. However the ones with the associated RTL remain.



Tcl Console | Messages | Log | Reports | Design Runs | Timing | Package Pins | I/O Ports

Timing Checks - Setup

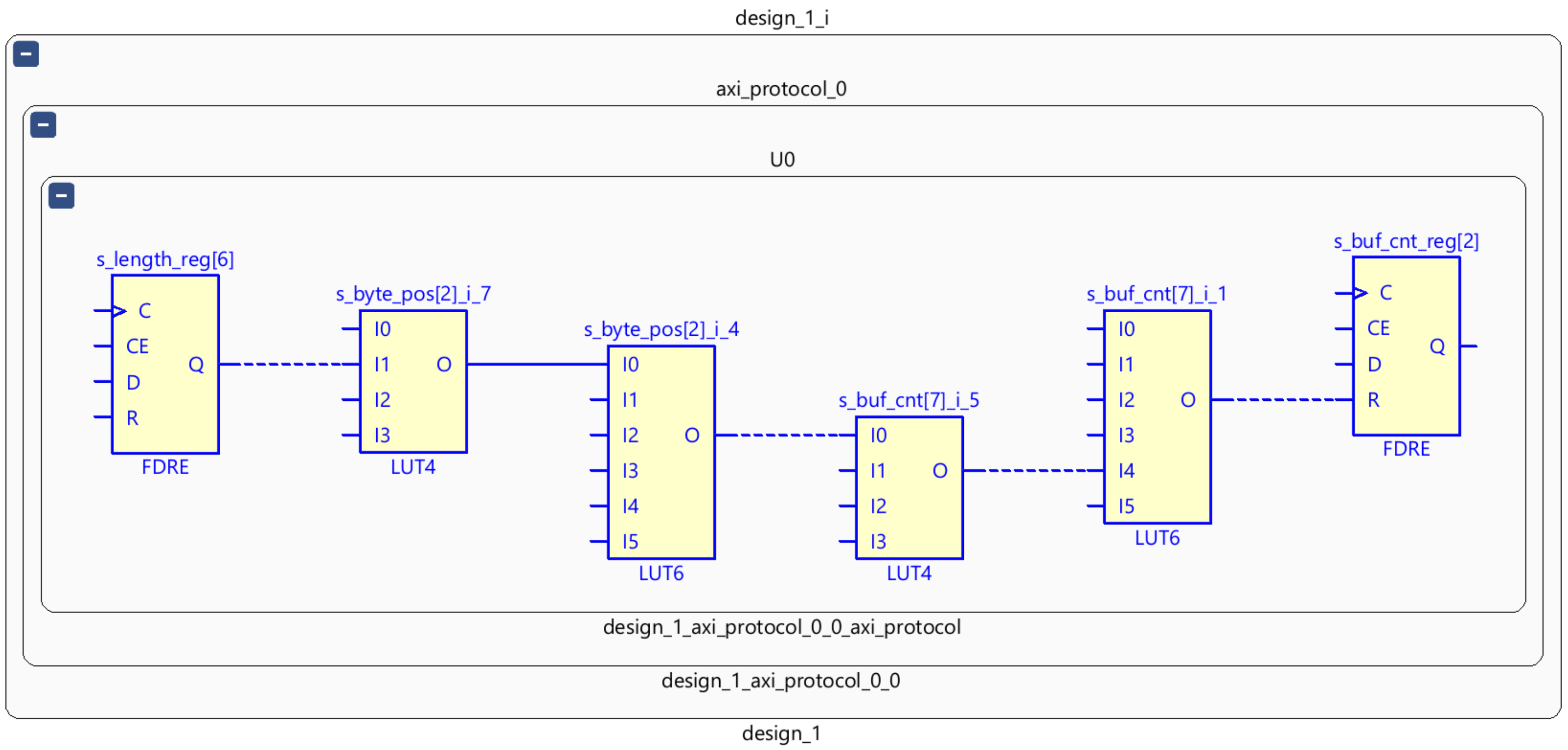
- General Information
- Settings
- Timing Checks (20)
  - Setup (10)
  - Hold (10)

Constrained Paths (1)
clk_in1_0 (10)

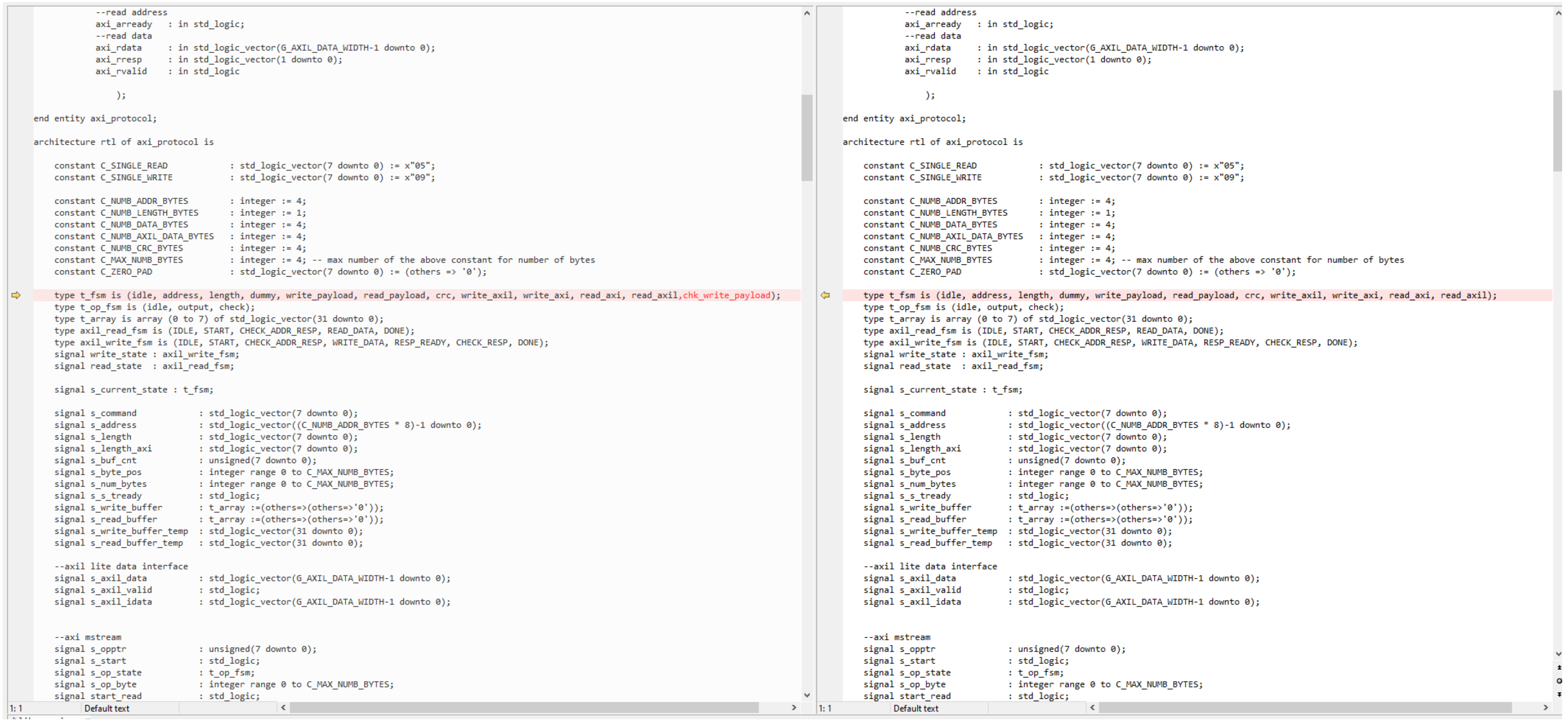
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Destination Clock	Exception	Skew	Clock Uncertainty	Inter
Path 1	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[2]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 2	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[3]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 3	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[4]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 4	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[5]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 5	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[6]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 6	-0.315	4	5	6	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_buf_cnt_reg[7]/R	4.578	1.145	3.433	25.0	75.0	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 7	0.304	4	5	32	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_write_buffer_reg[0][0]/CE	4.314	1.145	3.169	26.5	73.5	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 8	0.304	4	5	32	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_write_buffer_reg[0][10]/CE	4.314	1.145	3.169	26.5	73.5	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 9	0.304	4	5	32	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_write_buffer_reg[0][11]/CE	4.314	1.145	3.169	26.5	73.5	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 10	0.304	4	5	32	design_1_i/axi_protocol_0/U0/s_length_reg[6]/C	design_1_i/axi_protocol_0/U0/s_write_buffer_reg[0][12]/CE	4.314	1.145	3.169	26.5	73.5	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	

Custom RTL Code

Examining the schematic as we did for the Smart Connect we will see the issue is the number of LUTs and the delay between two registers.



Custom RTL Code



```
            --read address
            axi_arready   : in std_logic;
            --read data
            axi_rdata     : in std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);
            axi_rresp     : in std_logic_vector(1 downto 0);
            axi_rvalid    : in std_logic

                );

end entity axi_protocol;

architecture rtl of axi_protocol is

    constant C_SINGLE_READ           : std_logic_vector(7 downto 0) := x"05";
    constant C_SINGLE_WRITE          : std_logic_vector(7 downto 0) := x"09";

    constant C_NUMB_ADDR_BYTES       : integer := 4;
    constant C_NUMB_LENGTH_BYTES     : integer := 1;
    constant C_NUMB_DATA_BYTES       : integer := 4;
    constant C_NUMB_AXIL_DATA_BYTES  : integer := 4;
    constant C_NUMB_CRC_BYTES        : integer := 4;
    constant C_MAX_NUMB_BYTES        : integer := 4; -- max number of the above constant for number of bytes
    constant C_ZERO_PAD              : std_logic_vector(7 downto 0) := (others => '0');

    type t_fsm is (idle, address, length, dummy, write_payload, read_payload, crc, write_axil, write_axi, read_axi, read_axil,chk_write_payload);
    type t_op_fsm is (idle, output, check);
    type t_array is array (0 to 7) of std_logic_vector(31 downto 0);
    type axil_read_fsm is (IDLE, START, CHECK_ADDR_RESP, READ_DATA, DONE);
    type axil_write_fsm is (IDLE, START, CHECK_ADDR_RESP, WRITE_DATA, RESP_READY, CHECK_RESP, DONE);
    signal write_state : axil_write_fsm;
    signal read_state  : axil_read_fsm;

    signal s_current_state : t_fsm;

    signal s_command            : std_logic_vector(7 downto 0);
    signal s_address            : std_logic_vector((C_NUMB_ADDR_BYTES * 8)-1 downto 0);
    signal s_length             : std_logic_vector(7 downto 0);
    signal s_length_axi         : std_logic_vector(7 downto 0);
    signal s_buf_cnt            : unsigned(7 downto 0);
    signal s_byte_pos           : integer range 0 to C_MAX_NUMB_BYTES;
    signal s_num_bytes          : integer range 0 to C_MAX_NUMB_BYTES;
    signal s_s_tready           : std_logic;
    signal s_write_buffer       : t_array :=(others=>(others=>'0'));
    signal s_read_buffer        : t_array :=(others=>(others=>'0'));
    signal s_write_buffer_temp  : std_logic_vector(31 downto 0);
    signal s_read_buffer_temp   : std_logic_vector(31 downto 0);

    --axil lite data interface
    signal s_axil_data          : std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);
    signal s_axil_valid         : std_logic;
    signal s_axil_idata         : std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);


    --axi mstream
    signal s_opptr              : unsigned(7 downto 0);
    signal s_start              : std_logic;
    signal s_op_state           : t_op_fsm;
    signal s_op_byte            : integer range 0 to C_MAX_NUMB_BYTES;
    signal start_read           : std_logic;
```

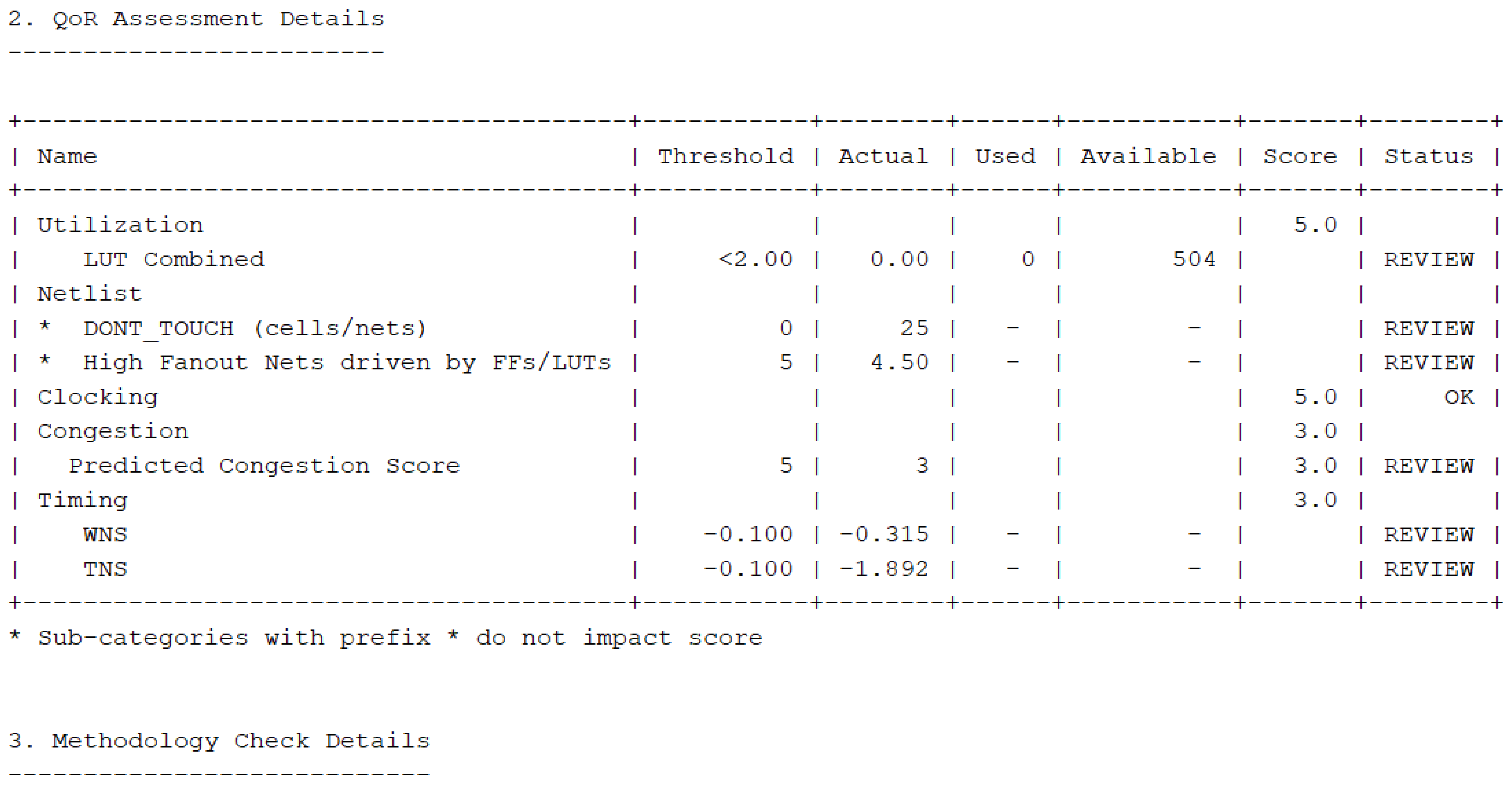
1: 1 Default text

```
            --read address
            axi_arready   : in std_logic;
            --read data
            axi_rdata     : in std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);
            axi_rresp     : in std_logic_vector(1 downto 0);
            axi_rvalid    : in std_logic

                );

end entity axi_protocol;

architecture rtl of axi_protocol is

    constant C_SINGLE_READ           : std_logic_vector(7 downto 0) := x"05";
    constant C_SINGLE_WRITE          : std_logic_vector(7 downto 0) := x"09";

    constant C_NUMB_ADDR_BYTES       : integer := 4;
    constant C_NUMB_LENGTH_BYTES     : integer := 1;
    constant C_NUMB_DATA_BYTES       : integer := 4;
    constant C_NUMB_AXIL_DATA_BYTES  : integer := 4;
    constant C_NUMB_CRC_BYTES        : integer := 4;
    constant C_MAX_NUMB_BYTES        : integer := 4; -- max number of the above constant for number of bytes
    constant C_ZERO_PAD              : std_logic_vector(7 downto 0) := (others => '0');

    type t_fsm is (idle, address, length, dummy, write_payload, read_payload, crc, write_axil, write_axi, read_axi, read_axil);
    type t_op_fsm is (idle, output, check);
    type t_array is array (0 to 7) of std_logic_vector(31 downto 0);
    type axil_read_fsm is (IDLE, START, CHECK_ADDR_RESP, READ_DATA, DONE);
    type axil_write_fsm is (IDLE, START, CHECK_ADDR_RESP, WRITE_DATA, RESP_READY, CHECK_RESP, DONE);
    signal write_state : axil_write_fsm;
    signal read_state  : axil_read_fsm;

    signal s_current_state : t_fsm;

    signal s_command            : std_logic_vector(7 downto 0);
    signal s_address            : std_logic_vector((C_NUMB_ADDR_BYTES * 8)-1 downto 0);
    signal s_length             : std_logic_vector(7 downto 0);
    signal s_length_axi         : std_logic_vector(7 downto 0);
    signal s_buf_cnt            : unsigned(7 downto 0);
    signal s_byte_pos           : integer range 0 to C_MAX_NUMB_BYTES;
    signal s_num_bytes          : integer range 0 to C_MAX_NUMB_BYTES;
    signal s_s_tready           : std_logic;
    signal s_write_buffer       : t_array :=(others=>(others=>'0'));
    signal s_read_buffer        : t_array :=(others=>(others=>'0'));
    signal s_write_buffer_temp  : std_logic_vector(31 downto 0);
    signal s_read_buffer_temp   : std_logic_vector(31 downto 0);

    --axil lite data interface
    signal s_axil_data          : std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);
    signal s_axil_valid         : std_logic;
    signal s_axil_idata         : std_logic_vector(G_AXIL_DATA_WIDTH-1 downto 0);


    --axi mstream
    signal s_opptr              : unsigned(7 downto 0);
    signal s_start              : std_logic;
    signal s_op_state           : t_op_fsm;
    signal s_op_byte            : integer range 0 to C_MAX_NUMB_BYTES;
    signal start_read           : std_logic;
```

1: 1 Default text

Custom RTL

Running the QOR Assessment on the design indicates we might have some challenges with the RTL design as it stands



```
2. QoR Assessment Details
-------------------------

+-----------------------------------------+-----------+--------+------+-----------+-------+--------+
| Name                                    | Threshold | Actual | Used | Available | Score | Status |
+-----------------------------------------+-----------+--------+------+-----------+-------+--------+
| Utilization                             |           |        |      |           |   5.0 |        |
|    LUT Combined                         |     <2.00 |   0.00 |    0 |       504 |       | REVIEW |
| Netlist                                 |           |        |      |           |       |        |
| *  DONT_TOUCH (cells/nets)              |         0 |     25 |  -   |       -   |       | REVIEW |
| *  High Fanout Nets driven by FFs/LUTs  |         5 |   4.50 |  -   |       -   |       | REVIEW |
| Clocking                                |           |        |      |           |   5.0 |     OK |
| Congestion                              |           |        |      |           |   3.0 |
|   Predicted Congestion Score            |         5 |      3 |      |           |   3.0 | REVIEW |
| Timing                                  |           |        |      |           |   3.0 |        |
|    WNS                                  |    -0.100 | -0.315 |  -   |       -   |       | REVIEW |
|    TNS                                  |    -0.100 | -1.892 |  -   |       -   |       | REVIEW |
+-----------------------------------------+-----------+--------+------+-----------+-------+--------+
* Sub-categories with prefix * do not impact score


3. Methodology Check Details
----------------------------
```

Custom RTL

Issue is between registers s_length and s_buff_cnt

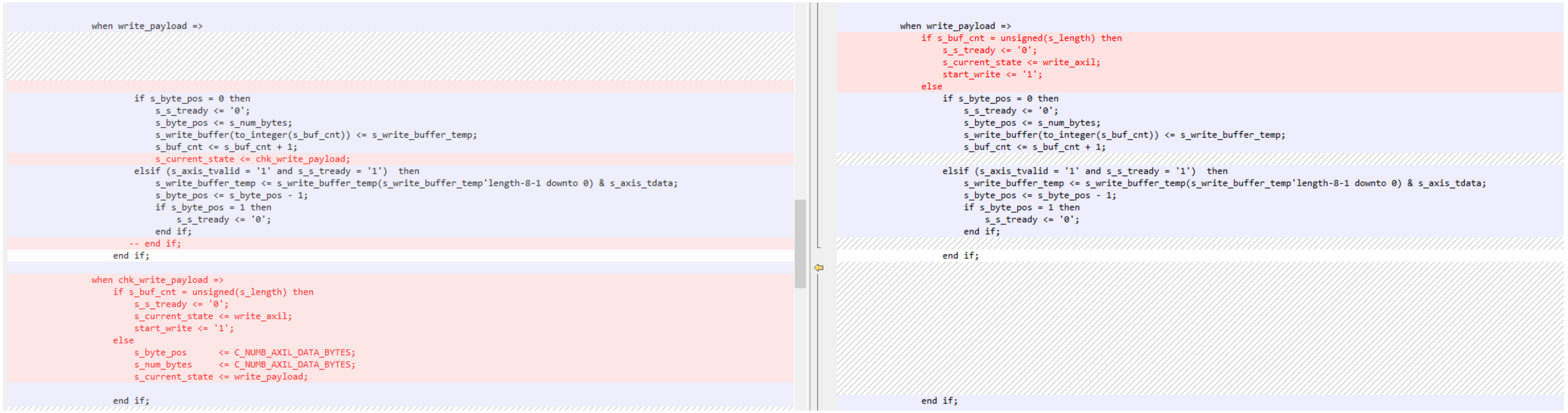
Bytes are received one at a time over the AXIS (UART) 4 bytes needed to do an AXIL 32 write.

As this was modified from our code which supported AXI – the check was included to ensure we can support multiple words.

Three options

1. Remove code – Valid approach but not good for this demo
2. Modify state machine – add another state – Option I chose
3. Define multi cycle path constraint.

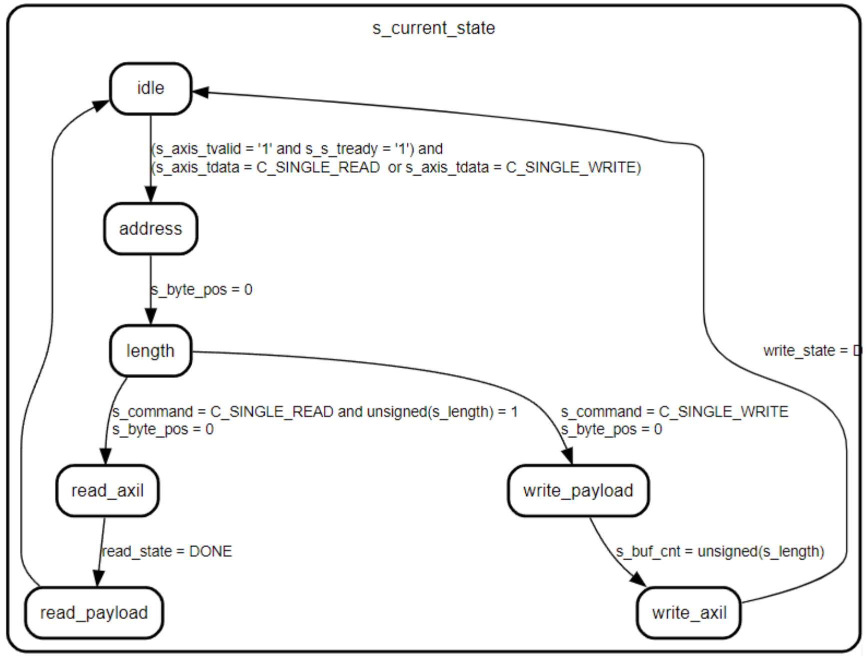
Custom RTL Code



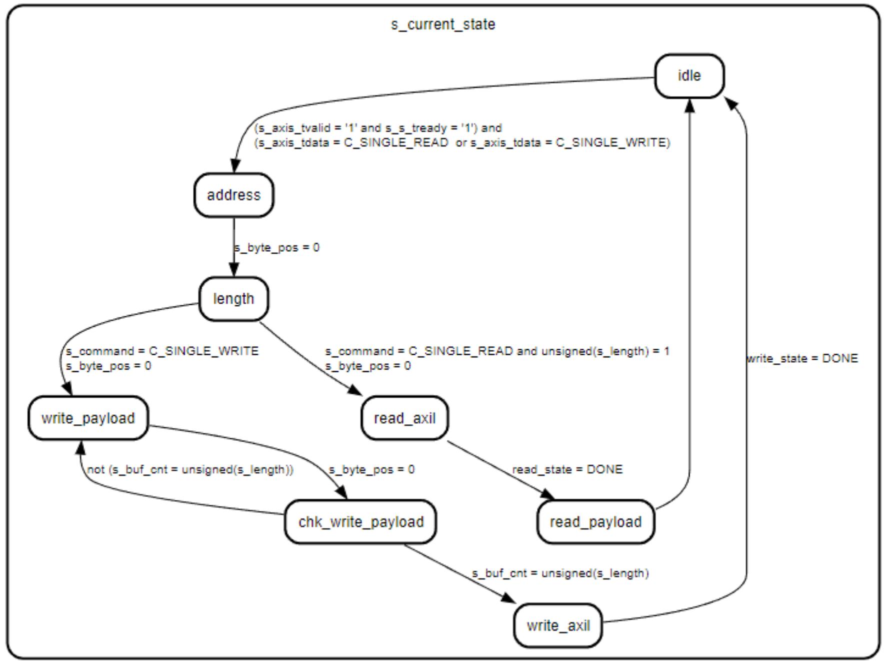
```
        when write_payload =>



                if s_byte_pos = 0 then
                    s_s_tready <= '0';
                    s_byte_pos <= s_num_bytes;
                    s_write_buffer(to_integer(s_buf_cnt)) <= s_write_buffer_temp;
                    s_buf_cnt <= s_buf_cnt + 1;
                    s_current_state <= chk_write_payload;
                elsif (s_axis_tvalid = '1' and s_s_tready = '1')  then
                    s_write_buffer_temp <= s_write_buffer_temp(s_write_buffer_temp'length-8-1 downto 0) & s_axis_tdata;
                    s_byte_pos <= s_byte_pos - 1;
                    if s_byte_pos = 1 then
                        s_s_tready <= '0';
                    end if;
               -- end if;
            end if;

        when chk_write_payload =>
            if s_buf_cnt = unsigned(s_length) then
                s_s_tready <= '0';
                s_current_state <= write_axil;
                start_write <= '1';
            else
                s_byte_pos      <= C_NUMB_AXIL_DATA_BYTES;
                s_num_bytes     <= C_NUMB_AXIL_DATA_BYTES;
                s_current_state <= write_payload;

            end if;
```

```
        when write_payload =>
            if s_buf_cnt = unsigned(s_length) then
                s_s_tready <= '0';
                s_current_state <= write_axil;
                start_write <= '1';
            else
                if s_byte_pos = 0 then
                    s_s_tready <= '0';
                    s_byte_pos <= s_num_bytes;
                    s_write_buffer(to_integer(s_buf_cnt)) <= s_write_buffer_temp;
                    s_buf_cnt <= s_buf_cnt + 1;

                elsif (s_axis_tvalid = '1' and s_s_tready = '1')  then
                    s_write_buffer_temp <= s_write_buffer_temp(s_write_buffer_temp'length-8-1 downto 0) & s_axis_tdata;
                    s_byte_pos <= s_byte_pos - 1;
                    if s_byte_pos = 1 then
                        s_s_tready <= '0';
                    end if;

                end if;

            end if;
```

Custom RTL Code



s_current_state
idle
(s_axis_tvalid = '1' and s_s_tready = '1') and
(s_axis_tdata = C_SINGLE_READ or s_axis_tdata = C_SINGLE_WRITE)
address
s_byte_pos = 0
length
s_command = C_SINGLE_READ and unsigned(s_length) = 1
s_byte_pos = 0
s_command = C_SINGLE_WRITE
s_byte_pos = 0
write_state = D
read_axil
write_payload
read_state = DONE
s_buf_cnt = unsigned(s_length)
read_payload
write_axil

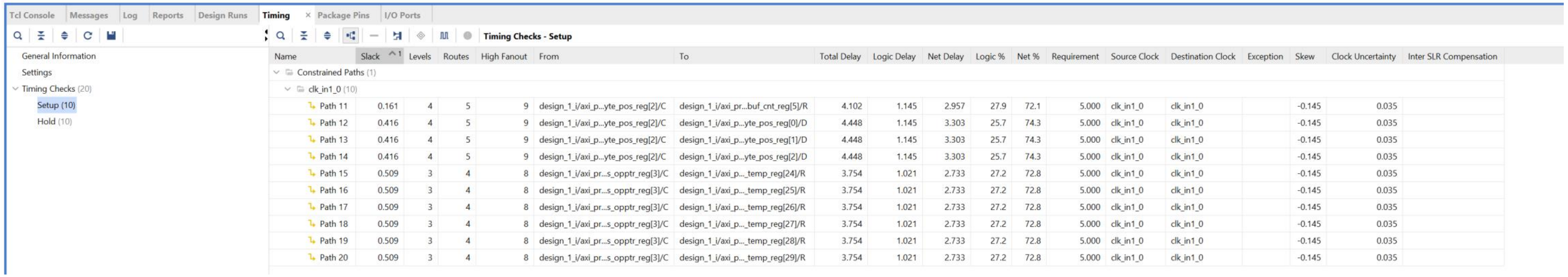


s_current_state
idle
(s_axis_tvalid = '1' and s_s_tready = '1') and
(s_axis_tdata = C_SINGLE_READ or s_axis_tdata = C_SINGLE_WRITE)
address
s_byte_pos = 0
length
s_command = C_SINGLE_WRITE
s_byte_pos = 0
s_command = C_SINGLE_READ and unsigned(s_length) = 1
s_byte_pos = 0
write_state = DONE
write_payload
read_axil
not (s_buf_cnt = unsigned(s_length))
s_byte_pos = 0
read_state = DONE
chk_write_payload
read_payload
s_buf_cnt = unsigned(s_length)
write_axil

Custom RTL

Rerun synthesis, optimization of the design and the timing report and we will see the issues resolved

We can now progress to placing the design



Tcl Console | Messages | Log | Reports | Design Runs | Timing | Package Pins | I/O Ports

Timing Checks - Setup

- General Information
- Settings
- Timing Checks (20)
  - Setup (10)
  - Hold (10)

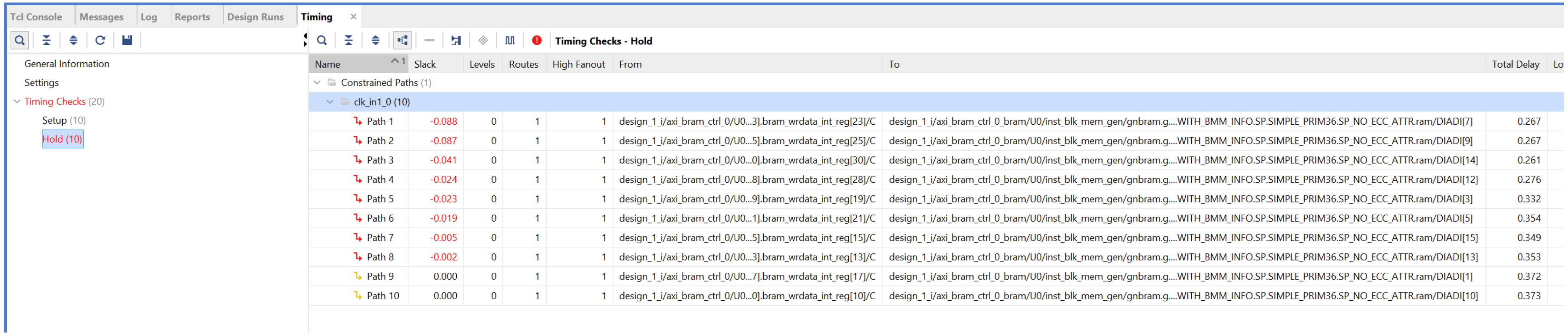
Constrained Paths (1)
clk_in1_0 (10)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Destination Clock	Exception	Skew	Clock Uncertainty	Inter SLR Compensation
Path 11	0.161	4	5	9	design_1_i/axi_p...yte_pos_reg[2]/C	design_1_i/axi_pr...buf_cnt_reg[5]/R	4.102	1.145	2.957	27.9	72.1	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 12	0.416	4	5	9	design_1_i/axi_p...yte_pos_reg[2]/C	design_1_i/axi_p...yte_pos_reg[0]/D	4.448	1.145	3.303	25.7	74.3	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 13	0.416	4	5	9	design_1_i/axi_p...yte_pos_reg[2]/C	design_1_i/axi_p...yte_pos_reg[1]/D	4.448	1.145	3.303	25.7	74.3	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 14	0.416	4	5	9	design_1_i/axi_p...yte_pos_reg[2]/C	design_1_i/axi_p...yte_pos_reg[2]/D	4.448	1.145	3.303	25.7	74.3	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 15	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[24]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 16	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[25]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 17	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[26]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 18	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[27]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 19	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[28]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	
Path 20	0.509	3	4	8	design_1_i/axi_pr...s_opptr_reg[3]/C	design_1_i/axi_p..._temp_reg[29]/R	3.754	1.021	2.733	27.2	72.8	5.000	clk_in1_0	clk_in1_0		-0.145	0.035	

Place Design

Running the place design and reporting the timing, we can see with the placement completed there are minor hold violations

However, as these hold violations are low we can see if the router can address them



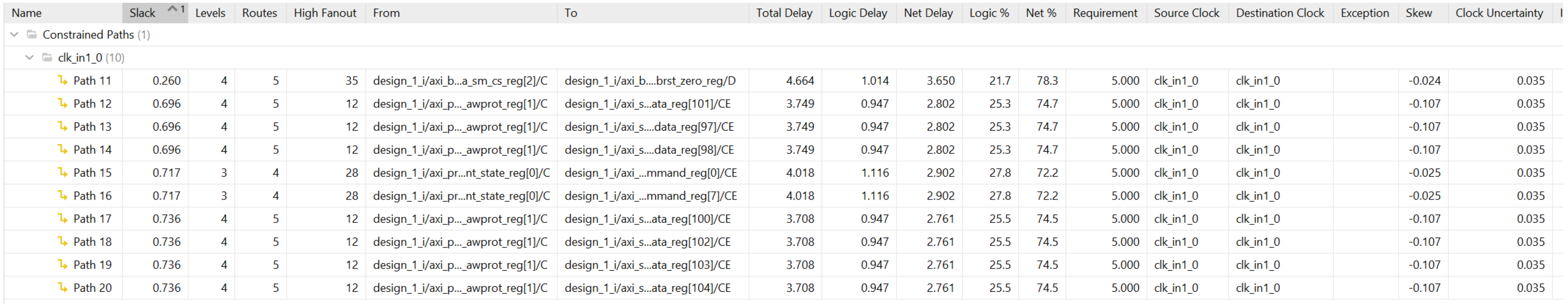
Timing Checks - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Lo
Constrained Paths (1)								
clk_in1_0 (10)								
Path 1	-0.088	0	1	1	design_1_i/axi_bram_ctrl_0/U0...3].bram_wrdata_int_reg[23]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[7]	0.267	
Path 2	-0.087	0	1	1	design_1_i/axi_bram_ctrl_0/U0...5].bram_wrdata_int_reg[25]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[9]	0.267	
Path 3	-0.041	0	1	1	design_1_i/axi_bram_ctrl_0/U0...0].bram_wrdata_int_reg[30]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[14]	0.261	
Path 4	-0.024	0	1	1	design_1_i/axi_bram_ctrl_0/U0...8].bram_wrdata_int_reg[28]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[12]	0.276	
Path 5	-0.023	0	1	1	design_1_i/axi_bram_ctrl_0/U0...9].bram_wrdata_int_reg[19]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[3]	0.332	
Path 6	-0.019	0	1	1	design_1_i/axi_bram_ctrl_0/U0...1].bram_wrdata_int_reg[21]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[5]	0.354	
Path 7	-0.005	0	1	1	design_1_i/axi_bram_ctrl_0/U0...5].bram_wrdata_int_reg[15]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[15]	0.349	
Path 8	-0.002	0	1	1	design_1_i/axi_bram_ctrl_0/U0...3].bram_wrdata_int_reg[13]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[13]	0.353	
Path 9	0.000	0	1	1	design_1_i/axi_bram_ctrl_0/U0...7].bram_wrdata_int_reg[17]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[1]	0.372	
Path 10	0.000	0	1	1	design_1_i/axi_bram_ctrl_0/U0...0].bram_wrdata_int_reg[10]/C	design_1_i/axi_bram_ctrl_0_bram/U0/inst_blk_mem_gen/gnbram.g....WITH_BMM_INFO.SP.SIMPLE_PRIM36.SP_NO_ECC_ATTR.ram/DIADI[10]	0.373	

Route Design

Hold violation addressed, by the router we have a design which now has a baseline.

Using this baseline we can go on to develop the IO constraints are correctly defined.



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement	Source Clock	Destination Clock	Exception	Skew	Clock Uncertainty
Constrained Paths (1)																	
clk_in1_0 (10)																	
Path 11	0.260	4	5	35	design_1_i/axi_b...a_sm_cs_reg[2]/C	design_1_i/axi_b....brst_zero_reg/D	4.664	1.014	3.650	21.7	78.3	5.000	clk_in1_0	clk_in1_0		-0.024	0.035
Path 12	0.696	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s...ata_reg[101]/CE	3.749	0.947	2.802	25.3	74.7	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 13	0.696	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s....data_reg[97]/CE	3.749	0.947	2.802	25.3	74.7	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 14	0.696	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s....data_reg[98]/CE	3.749	0.947	2.802	25.3	74.7	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 15	0.717	3	4	28	design_1_i/axi_pr...nt_state_reg[0]/C	design_1_i/axi_...mmand_reg[0]/CE	4.018	1.116	2.902	27.8	72.2	5.000	clk_in1_0	clk_in1_0		-0.025	0.035
Path 16	0.717	3	4	28	design_1_i/axi_pr...nt_state_reg[0]/C	design_1_i/axi_...mmand_reg[7]/CE	4.018	1.116	2.902	27.8	72.2	5.000	clk_in1_0	clk_in1_0		-0.025	0.035
Path 17	0.736	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s...ata_reg[100]/CE	3.708	0.947	2.761	25.5	74.5	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 18	0.736	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s...ata_reg[102]/CE	3.708	0.947	2.761	25.5	74.5	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 19	0.736	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s...ata_reg[103]/CE	3.708	0.947	2.761	25.5	74.5	5.000	clk_in1_0	clk_in1_0		-0.107	0.035
Path 20	0.736	4	5	12	design_1_i/axi_p..._awprot_reg[1]/C	design_1_i/axi_s...ata_reg[104]/CE	3.708	0.947	2.761	25.5	74.5	5.000	clk_in1_0	clk_in1_0		-0.107	0.035

Summary of Example

To achieve a baseline, we have performed the following

1. Used Constraint Wizard to define the initial timing constraints
2. Corrected methodology violation – Insertion of BUFG on clock
3. Addressed a timing violation on Smart Interconnect by insertion of register stage.
4. Addressed a timing violation on Custom RTL by modifying the RTL
5. Allowed the router to address a small hold violation

Key take away most of these timing issues where addressed before we ran the Place and Route elements of the design which are the ones which take considerable time.

Advanced approaches to consider

There are several advanced approaches we can use for timing closure.

- Block Level Synthesis – Implement different synthesis options on modules in the design
- Optimize the number of logic levels
- Reduce control sets - Clock Enables and resets should be reduced whenever possible
- Optimise High fan out - Enable replication and promoting high fan out to global networks

User Guide 949 is a great starting point for advice

Conclusion

1. Think about timing from day one of the design
    a) Plan the clocking and clock networks

2. Follow guides such as Ultrafast Design Techniques to get the best possible design to start.

3. Establish the baseline timing closure following synthesis

4. Leverage the reports

Questions

ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com

info@adiuvoengineering.com