CDC Techniques Without Synchronizers: Holistic Approach for High Quality Low Power Design

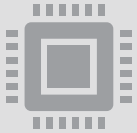
Sangeeta Raste (PMTS, AMD)
Preethi Venkateshan (MTS, AMD)
Grant Simmons (MTS, AMD)

AMD
together we advance_

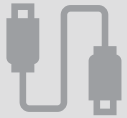
Agenda

- Introduction
- Background
- CDC approaches
  - Traditional approach
  - Alternative approach
- Example 1 – Single Bit Transfer
- Example 2 – Logic Rearrangement
- Conclusion

Introduction



Robust low power design with effective Clock Domain Crossing (CDC) management is crucial in ASIC development



Traditional multi-flop synchronizers increase power, performance, and area (PPA) overhead in low power designs

Innovative yet robust alternatives are needed to maintain PPA targets of low power design

Background

About CDC (Clock Domain Crossing)

- CDC refers to the transfer of data between different clock domains in a digital system
- Reliable CDC is crucial for ensuring data integrity and system reliability
- Proper CDC design and checking is essential to prevent data corruption and metastability
- Insufficient CDC design can lead to system failures

- Challenges in CDC
  - Metastability
  - Setup hold violations
  - Data loss
  - Glitches

CDC methodology

- **CDC traditional methods**
  - Multi flop synchronizer
  - Async FIFO
  - Pulse synchronizer
  - Handshaking protocols for complex systems
  - Gray code encoding

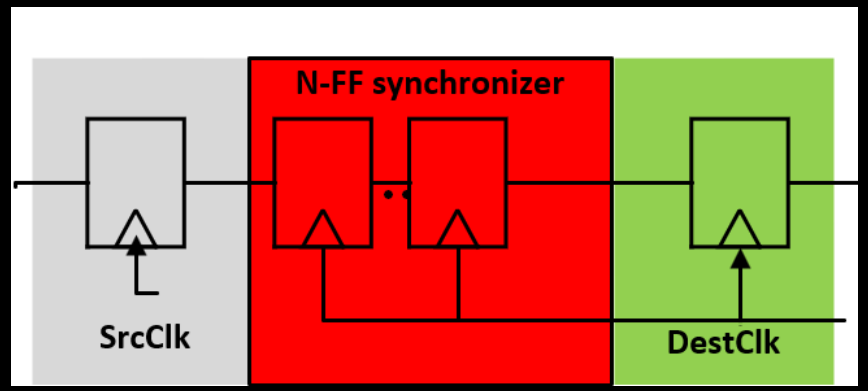
- **Factors affecting CDC methods**
  - Design architecture
  - Relationship between 2 clock domains
  - Timing relationships

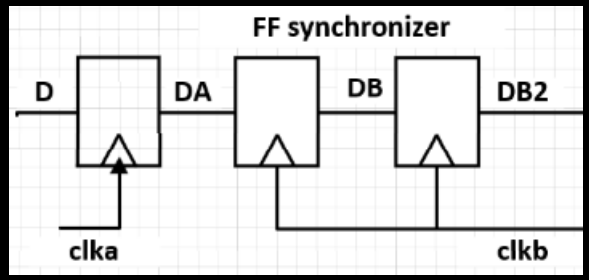
- **Tools**
  - Industry standards tools are used for CDC analysis and verification

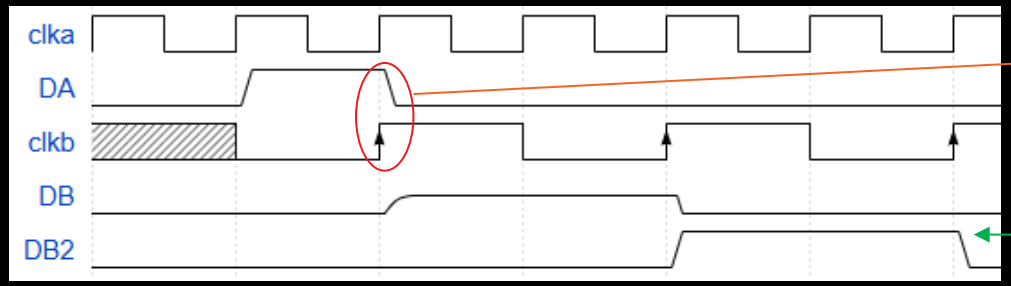
CDC Approaches for Single Bit Data Transfer

Traditional approach

- Flip-flop in one clock domain captures a signal that is generated in another clock domain
- If signal changes close to the clock edge of the receiving flip-flop, can lead to metastability
  - Flip-flop may not reach stable '0' or '1' before the next clock edge, resulting in an indeterminate state
- With synchronizer
  - The first flip-flop captures the potentially metastable signal
  - Subsequent flip-flops provide additional gain/time for the signal to stabilize





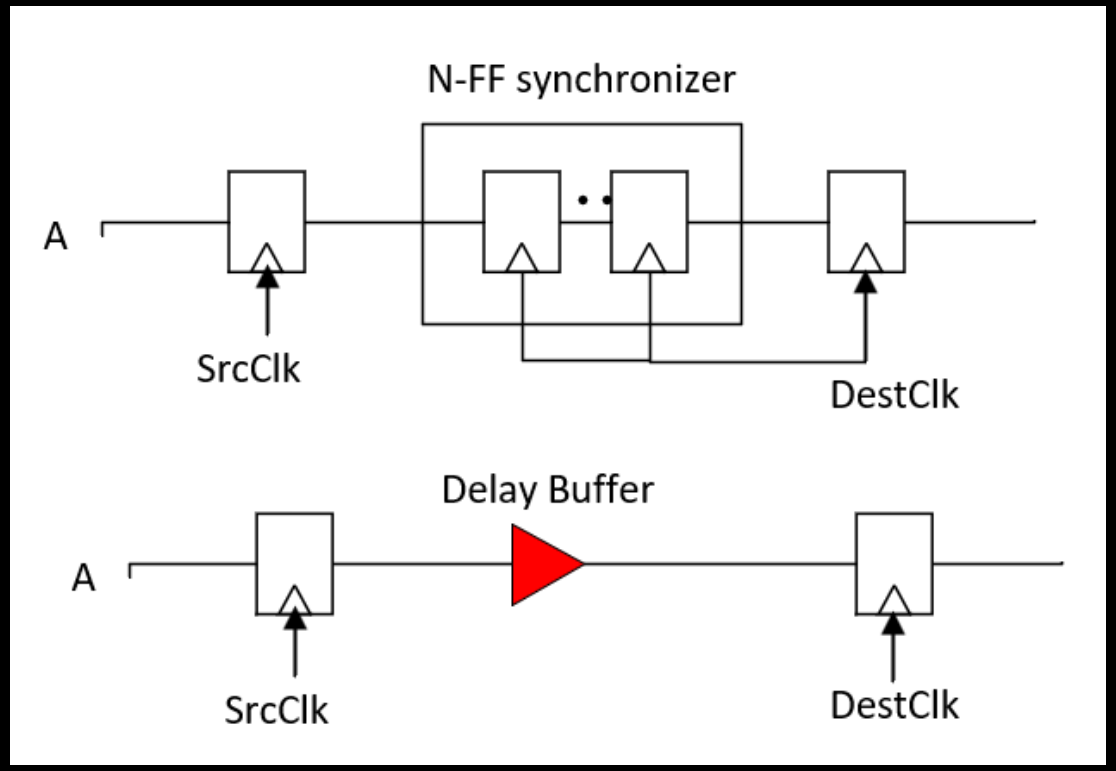


Traditional flop synchronizer and PPA impact

- Increased Area
  - Multi-flop synchronizers need extra flip-flops to maintain signal stability across different clock domains
  - Results in greater silicon area cost
- Power Consumption
  - More flip-flops leads to higher power consumption
- Performance Impact
  - The additional synchronization stages introduced by multi-flops can cause increased latency

Impact on PPA – Power, performance, and area are critical metrics for SoC efficiency
An alternative approach is necessary for low-power design applications

Alternative approach – delay buffers



- What is the purpose of delay buffers?
- Delay buffers serve as reliable alternatives when clocks are architecturally related, especially when the data signal operates at a slower pace
- Each delay buffer cell includes a max delay parameter
  - Delay based on the relationship between the source and destination clocks
  - Ensures safe signal transfer between SrcClk & DestClk
- Maximum delay value determined in collaboration with physical design teams
- Reduces the number of logic elements and registers compared to traditional synchronizers

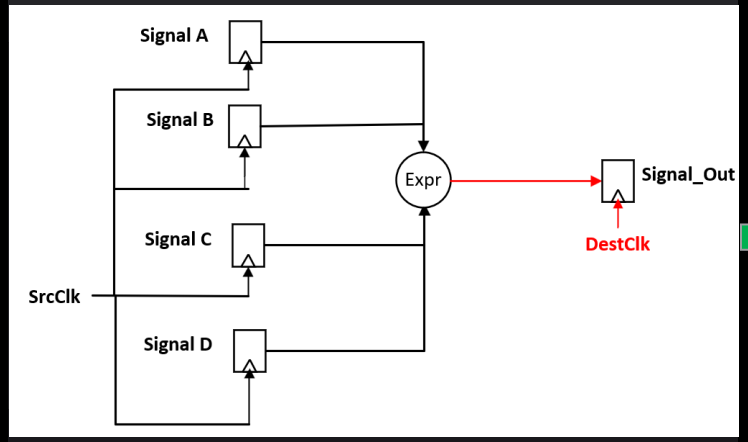
Delay buffers approach – where and how to use?

- Delay buffer cells address single-bit CDC violations without synchronizers
- Buffers safely transfer signals between clock domains within set delay limits
- CDC tools may need customization to correctly identify delay buffers
- To ensure robustness need to use assertion methodology and RTL checkers

Example 1 – Single Bit Transfer

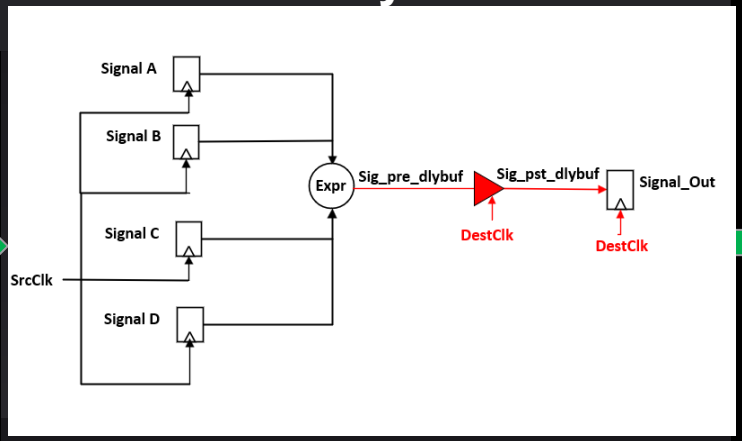
Example 1

Signal transfer



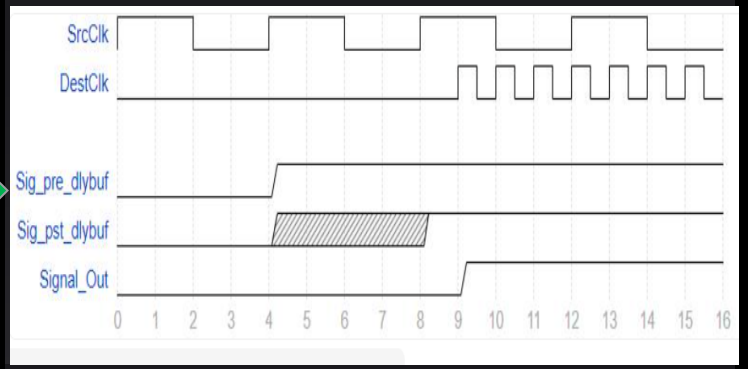
- Multiple signals from the SrcClk domain combine into a combinational logic expression, output is stored in a register in the DestClk domain
- CDC violation: missing synchronizer

Add delay buffer



- Insert a delay buffer at the output of the combination logic
- In the CDC tool, specify the delay buffer as a synchronizer
- This will resolve the CDC violation

Make it more robust by verification

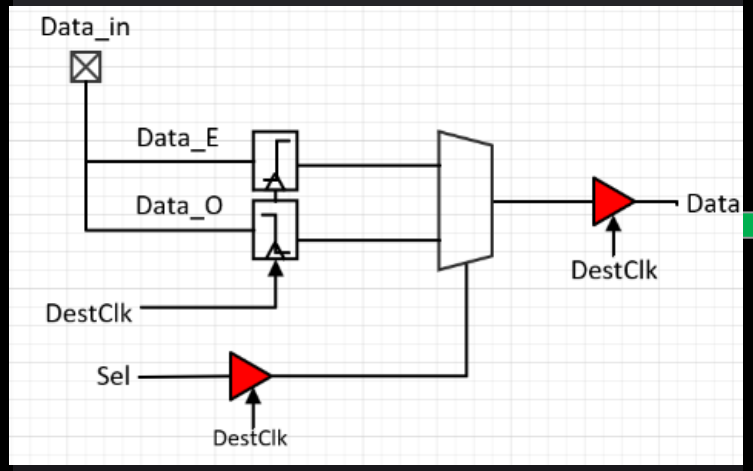


- Perform simulation
- Use assertion-based verification
- Verify that the clock on the target flop is correctly gated throughout the X injection period

Example 2 – Logic Rearrangement, Constraints/Assertions Robust Verification

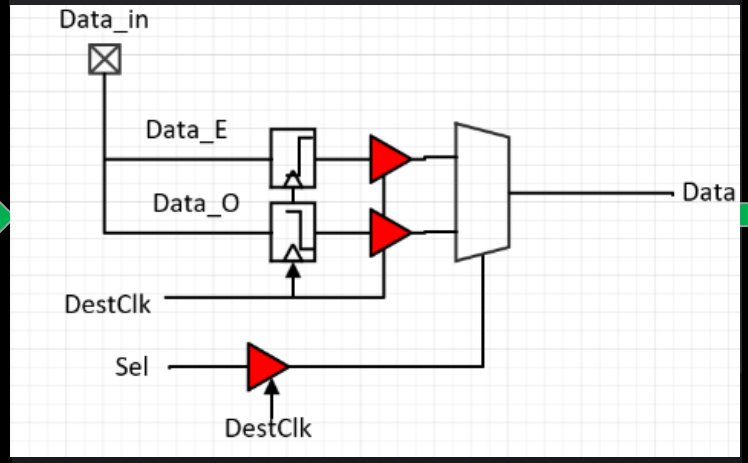
Example 2

Design



- Add delay buffers both at mux output and on select line
- Use CDC tool to apply delay buffers as synchronizers
- CDC violation: combinational logic before synchronizer

Taking care of combo logic before synch violation



- Insert delay buffers at the Mux input
- CDC violation: reconvergence
- Address reconvergence violations using constraints and assertions

Taking care of reconvergence violation

Add constraints to make sure mux inputs are mutually exclusive

Add Assertion to enforce select line stability

```
cdc_signal -mutually_exclusive
{Data_E} {Data_O} -module
{data_sampler}
```

```
cdc_signal -stable {Sel} -module
{data_sampler}

property mutually_excl_check;
   @(posedge DestClk)
        $stable(Data_E) | $stable(Data_O);
endproperty

Property stable_check;
    @(posedge DestClk)
        $stable(Sel);
endproperty
```

Conclusion

Conclusion

- In low power design it is critical to analyze the power and area cost of each logic addition
- With careful design and thoughtful application, the approaches discussed offer substantial benefits in achieving reliable and efficient CDC analysis, tools may need customization for delay buffers
- By minimizing additional logic, leveraging delay buffer cells, and enforcing rigorous timing and verification strategies, these techniques achieve high quality CDC signoff without compromising PPA (power, performance, or area)
- Approach establishes a scalable framework for high-quality, low power design, ensuring both functional correctness and design efficiency
- Some designs, especially with strict timing, may not suit these methods

Copyright and disclaimer


AMD