Improvements Needed

1. **Testing Serial Handshaking**
   - Currently, the **`serial_ready_in`** signal remains 1 throughout the **Testbench** verification process.
   - Proper validation requires testing transitions from **0 → 1 → 0** while receiving data.
   - This will ensure correct handshaking behavior and confirm that the **Serializer** properly reacts to ready transitions.
2. **Hold and Setup Timing Challenges concerning** the **`serial_ready_in Input port.`**
   - A significant amount of time was spent on resolving hold and setup time violations after synthesis and implementation.
   - The primary input port **`serial_ready_in`** was affected, even though there was no combinational logic in the path **(`serial_ready_in → serial_ready_in_t_reg`).**
   - Despite several modifications, Vivado's synthesis tool seemed to over-optimize this path, repeatedly introducing either hold or setup violations.
   - Different strategies were attempted:
     - Adding redundant combinational delay (e.g., two inverters, AND gate with a fixed 1)
     - Applying **`dont_touch`** to prevent optimization
     - Increasing state encoding bit width to force additional routing delay
   - However, these changes caused setup violations when hold was fixed and vice versa. Possible cause, the synthesis tool over-optimizes paths without logic, introducing aggressive routing optimizations. A more refined timing constraint strategy or a directive to disable over-optimization might be required.
   - **This prevented me from addressing point #1, Testing Serial hand shaking in Testbench with limited time available**
3. **Potential Solutions for Timing Closure**
   - One way to reduce timing pressure could be modifying the Serializer design:
     - Instead of sending one bit per cycle at 200 MHz, make the Serializer send one bit every two cycles at 200 MHz.
     - This would require **applying multi-cycle path constraints** for correct timing analysis.
   - Alternatively, reduce the Serializer clock speed by running it at 100 MHz instead of 200 MHz:
     - This would require **converting the Sync FIFO into an Async FIFO** with a CDC from 200 MHz to 100 MHz.
4. **Verifying Serializer Clock Speed Requirements**
   - The original specification in **"Gateware Engineer Assignment.pdf"** requires only the Sobel and Pooling layers to operate at 200 MHz.
   - There is no constraint requiring the Serializer to run at 200 MHz.
   - Therefore, the Serializer could be modified to run at 100 MHz, simplifying timing closure and reducing the risk of hold/setup violations.
   - The proposed pipeline:

     `Pooling Output (200 MHz) → Async FIFO (200 MHz to 100 MHz) → Serializer (100 MHz)`

   - This modification could significantly improve timing margin while still meeting design requirements.
   - Though I still believe Serializer could be run at 200 Mhz. It's a single Primary Input port to Register path that is giving me Hold violations. All other paths work at 200Mhz without issues inside the Serializer.
5. **Making all registers in design so Synchronous Reset instead of current Async Reset**
   - Due to my inexperience in designing circuits using Xilinx and Vivado, I started out with an Asynchronous reset strategy

- After running Synthesis and Implementation, I saw all the warnings about Async reset and there were numerous.
- I read some documentation and figured out, Xilinx FPGA designs prefer Sync resets to prevent contamination of memory elements like BRAM.
- So an easy fix to remove all Async resets for all registers can be implemented. Also remove the Async reset synchronizer module
- Async Reset to Sync Reset Conversion:

```
Async reset    :
  always @(posedge clk_200mhz or negedge reset_n) begin
    if (!reset_n) begin
      {stg1_Gxsum1, stg1_Gxsum2, stg1_Gxsum3} <= 0;
    end else begin
      stg1_Gxsum1 <= $signed(window[0][2]) - $signed(window[0][0]);

Sync Reset    :
  always @(posedge clk_200mhz ) begin
    if (!reset_n) begin
      {stg1_Gxsum1, stg1_Gxsum2, stg1_Gxsum3} <= 0;
    end else begin
```

-
- Also the Async FIFO module itself will need to use Async Reset, since Sync reset option doesn't seem to be available in Customize IP.