Key Concepts and Logic Fixes for FPGA Tetris Implementation

This document summarizes the essential concepts and logic improvements used to fix issues in the FPGA Tetris implementation on the Basys 3 board (Artix-7). These lessons are useful for future FPGA projects and technical interviews, where understanding timing, synchronization, and game logic is crucial.

Key Concepts Learned:

- **Clock Division and Timing Domains**: The Basys 3 board provides a 100 MHz clock by default. Game logic like Tetris does not need such a high speed. A clock divider was implemented to create a slower clock (25 MHz) for the Tetris logic. This avoided timing violations (negative WNS/TNS) while still ensuring smooth gameplay.
- **Gravity Logic for Automatic Falling**: Blocks originally only fell when the down button was pressed. A gravity counter driven by the divided clock was added. The counter generates a periodic 'gravity enable' signal (grav_ce), which triggers automatic falling at a human-playable speed.
- **Clock Domain Synchronization**: All logic that interacts (Tetris state machine, gravity, button inputs) must share the same clock domain to prevent glitches. Running debounce at 100 MHz while game logic was at 25 MHz caused missed button presses. Fix: run debounce modules at 25 MHz as well.
- **Debouncing Button Inputs**: Mechanical switches on the FPGA board produce noisy signals (bouncing). Debouncing ensures stable input detection by requiring the signal to remain steady for a set period (e.g., 10 ms). At 25 MHz, this means counting ~250,000 cycles before confirming a stable press.
- **Resource Usage and BRAM**: For large grid storage (like the 20x10 Tetris field), Block RAM (BRAM) is preferred over distributed flip-flops. This improves timing closure and avoids scrambling issues after line clears.
- **Placement and Routing Considerations**: When line clearing scrambled the grid, the issue was caused by memory not being synchronized with the display logic. Ensuring proper reset, sequential write operations, and avoiding simultaneous multiple drivers on the same memory cell were key fixes.
- **Wall and Rotation Logic**: Some blocks could not rotate fully near the wall due to boundary checks inside the 4x4 block rotation grid. The solution is to adjust collision detection so that the block shifts left/right when a rotation would otherwise overlap the wall.

By dividing the clock, synchronizing all modules to the same domain, adding gravity logic, and ensuring stable debouncing, the Tetris engine now runs smoothly on the FPGA. These design choices illustrate best practices in FPGA development and can be applied to future projects and interview discussions.