**SNES Controller Interface**

Our system interfaces with Super Nintendo Entertainment System (SNES) game controllers to handle user input. We chose the SNES controller because it features a relatively high number of button inputs and is fairly simple to interface with. Our design supports two controllers, although adding support for more controllers requires only one additional pin per controller and minimal changes to the controller logic.

The controllers are accessed via a two-step memory mapped IO operation. In the first step, the core instructs the controller interface to latch the buttons states of the attached controllers. This begins a multi-cycle operation in which the button states for each controller are shifted in serially. Since this is a high latency operation (~50 system clock cycles), the program will either have to spin until the button states are available to be read, or the programmer can opt to perform a separate task before returning to read the button data. Once the button states have been latched, the states for each individual controller can be read by the CPU using a memory-mapped load instruction.

**SNES Interface Protocol**

The SNES controller features a simple protocol for reading the state of the controller. There are three signal wires between the ASIC and the SNES controller: latch, pulse, and data. When the latch signal goes high, the SNES controller latches the button states into a 12-bit shift register. On the falling edge of the latch signal, the state of the first button is available on the data line. The ASIC then sends an addition 11 clock pulses on the pulse line, which shifts out the remaining 11 button states on the falling edge of the clock pulses.

In our design, the latch and pulse lines are shared between all attached controllers, and there is one data line returning from each controller. This allows us to read all controllers in parallel in order to share the high-latency latching operation among all controllers. We also have a more hardware intensive design which asynchronously latches the buttons states, which reduces software overhead. We might opt to use this design in our final fabricated chip if there is sufficient room within the frame.