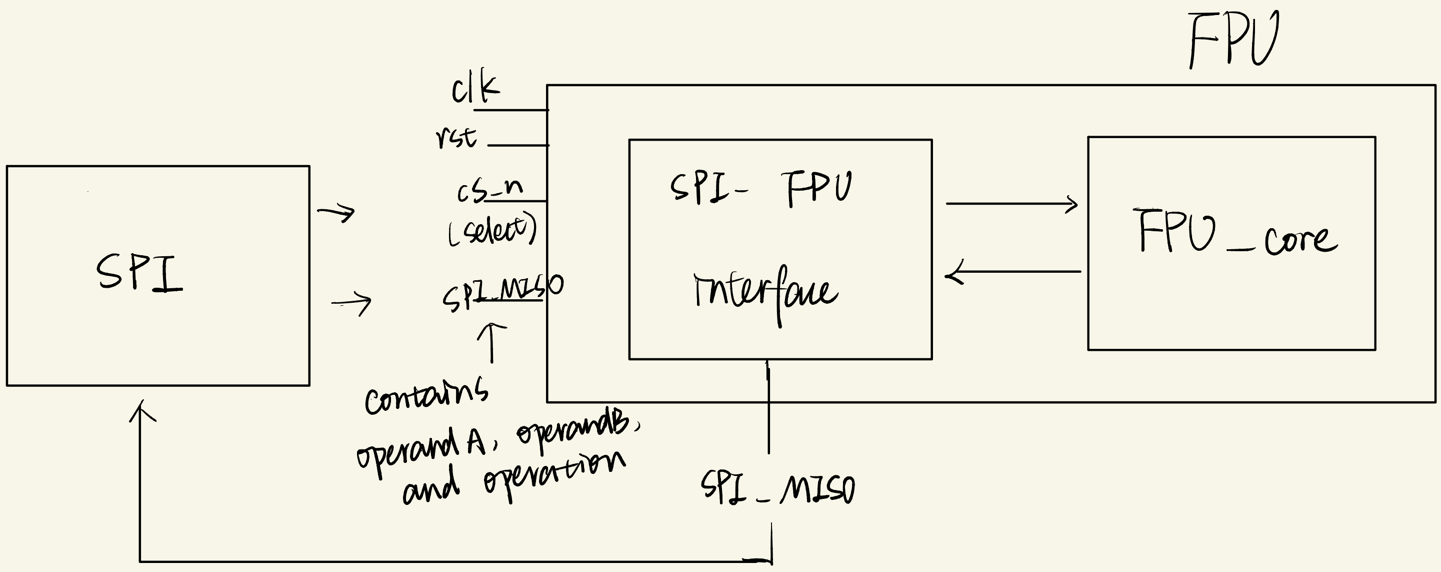
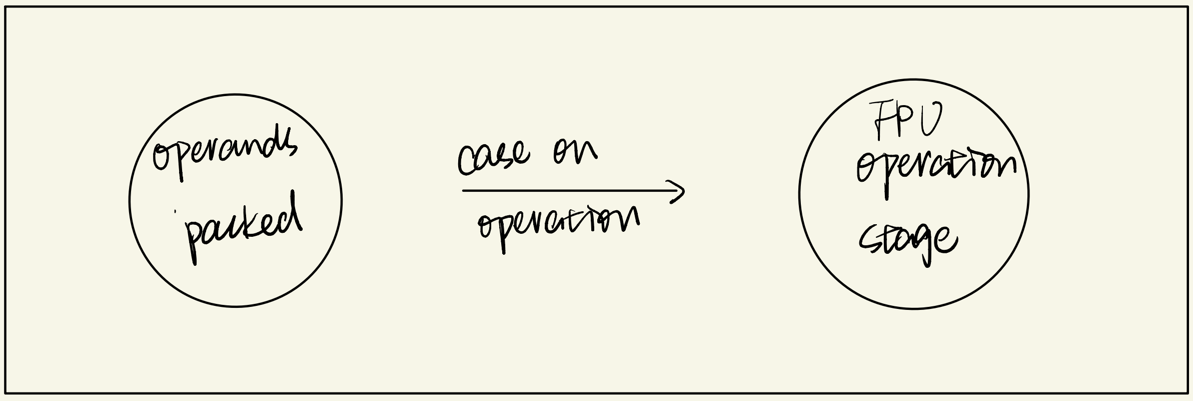
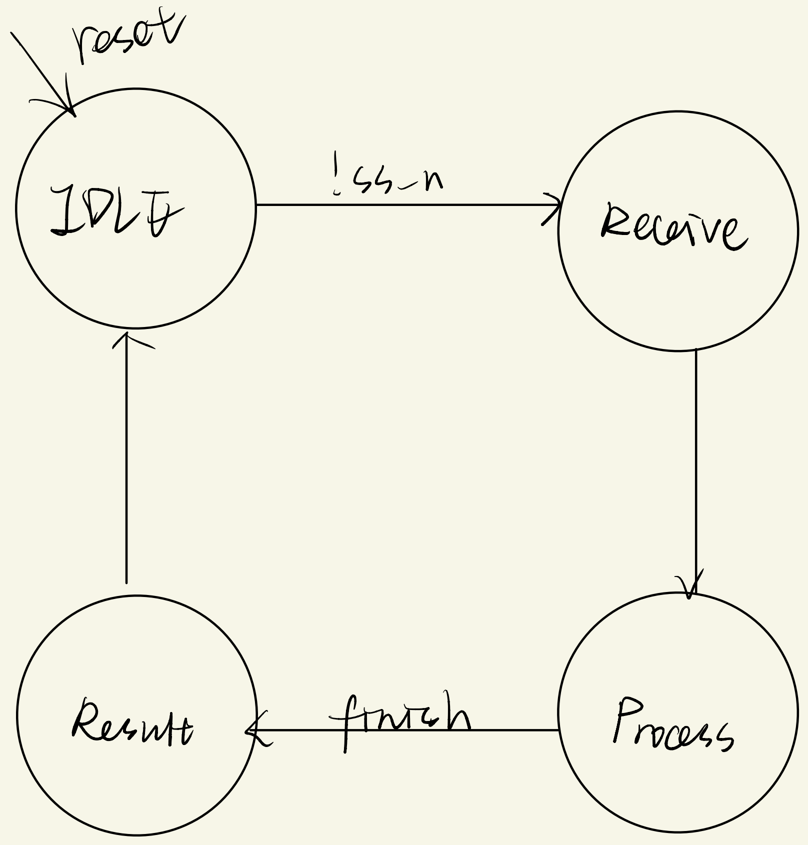
overview



FPU_core



SPI- FPU interface

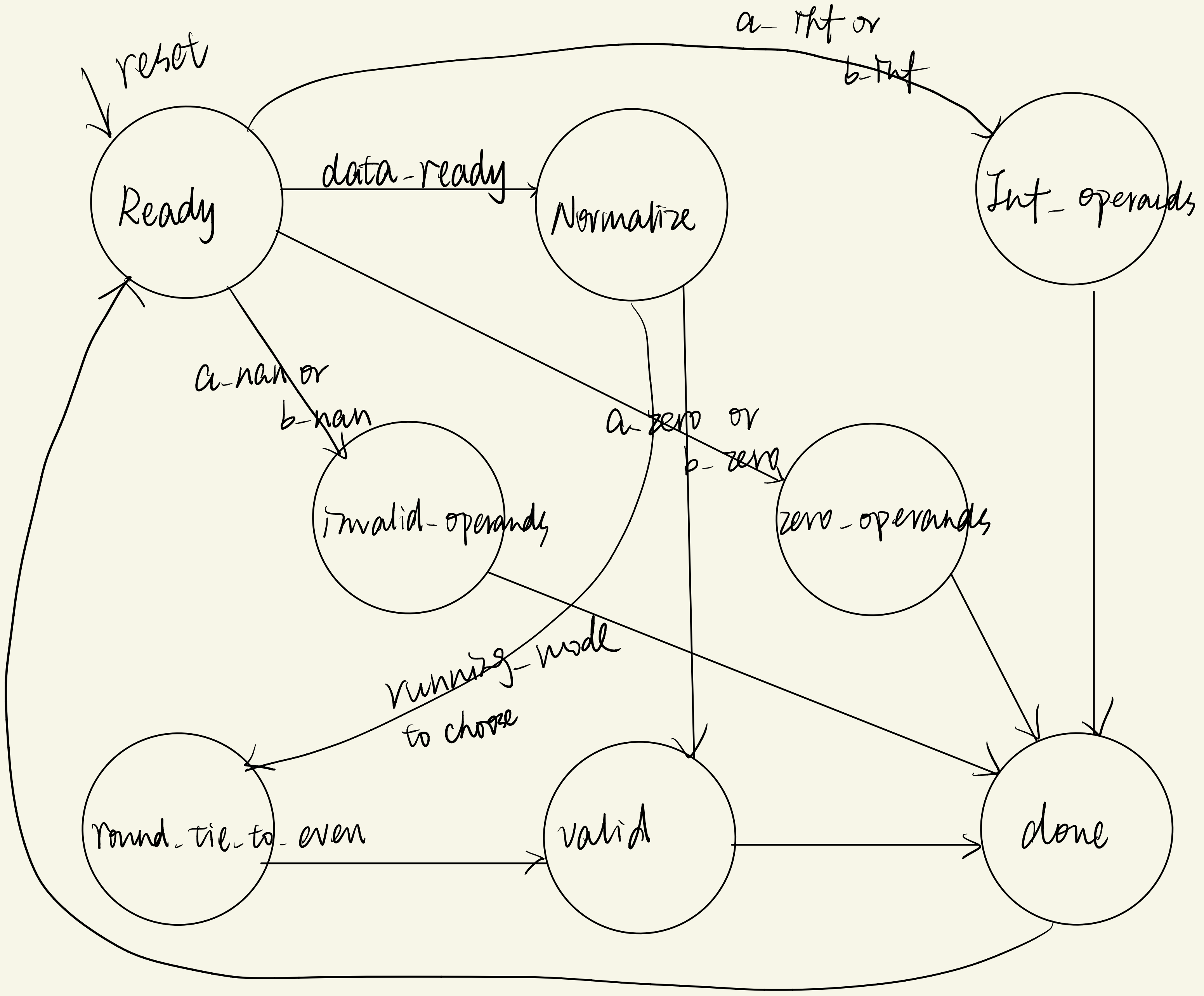


(have counter instead, keep track of full receive for both operands and operation mode)

(wait for results for FPU_core)

operation design for ADD:

using half precision



similar design for other operations

1. SPI Communication Testing

- **SystemVerilog Unit Test:** Verify SPI slave module receives and transmits data correctly.
- **CocoTB Integration Test:** Simulate SPI transactions and check correct data exchange between the microcontroller and the FPU.
- Test different SPI modes and clock speeds to confirm robustness.
- Inject random bit-flips to test error handling.

2. Floating-Point Arithmetic Operations Testing

- **SystemVerilog Unit Test:** Verify each arithmetic module independently (adder, multiplier, etc.).
- **CocoTB Integration Test:** Run full FPU operations and compare results to expected IEEE 754 outputs.
- Edge case testing:
    - **Zero (0.0 + 0.0)**
    - **Negative numbers (-3.5 + 2.1)**
    - **Large and small numbers (1e-38 + 1e38)**
- Cross-check hardware results with a Python or C-based floating-point reference model.

3. Handling Special Floating-Point Cases

- **Unit Test:** Inject NaN, Infinity, subnormal numbers, and verify proper propagation.
- **Integration Test:** Check system behavior when encountering invalid operations (e.g., Inf - Inf).
- Ensure division by zero triggers an error flag.

4. Latency and Throughput Testing

- **Unit Test:** Count cycles for each arithmetic module separately.
- **Integration Test:** Measure total system response time from SPI command to result output.
- Identify and optimize any performance bottlenecks.

5. Error Handling & Fault Injection

- **Unit Test:** Inject corrupted data and check if error flags are set.
- **Integration Test:** Introduce invalid SPI commands and verify system response.
- Confirm error LED activation for invalid operations.

6. Hardware-in-the-Loop (HIL) Testing on FPGA

Validate real-world performance on the iCE40 FPGA.

- Use a microcontroller or PC to send SPI commands and receive FPU results.
- Observe SPI signals with a logic analyzer.
- Automate test cases with CocoTB to stress-test the design.