Nick Ward (npward)
18-624 - Intro to Open-Source FPGA and ASIC Chip Design
3/27/25

EX9: Scanning Chains

Task #1: Building a Chain Interface

Task 1 - Part E:
See repo inside ScanChain_starter.py

Task 1 - Part F:
See repo inside ScanChain_starter.py

Task 1 - Part G:
See repo inside ScanChain_starter.py

Task 1 - Part H:
See repo inside ScanChain_starter.py

Task #2: Testing the Adder

I unfortunately did not have enough time to finish this. I was able to set up the test in the testbench, but I was not able to get anything of the scan chain values to appear on x_out, and I didn't have enough time to debug. Feel free to look inside of ScanChain_starter.py to see where I got to.

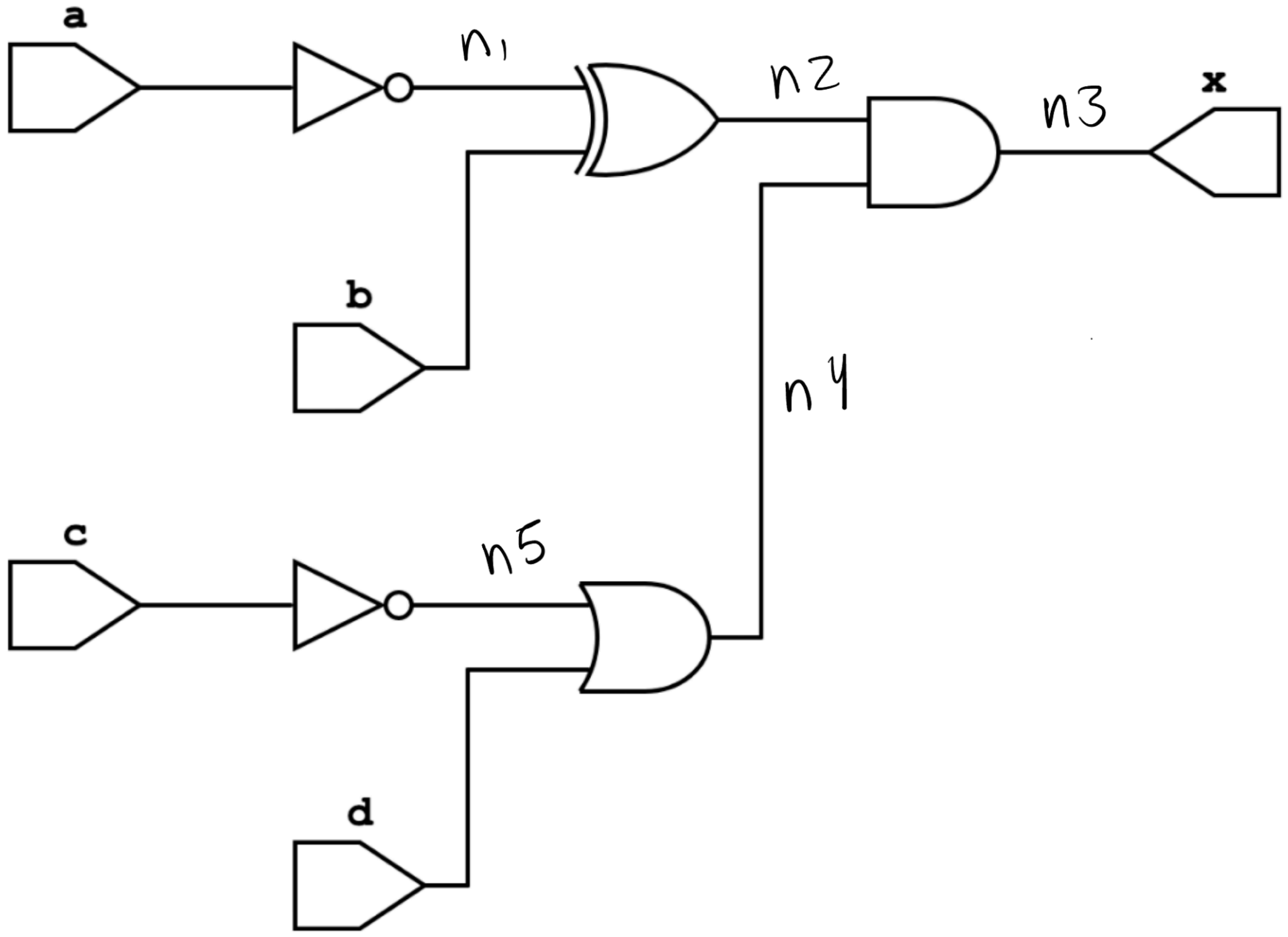
Task #3: FSM Testing

Due to getting blocked on the scan chain issue before, I unfortunately did not have enough time to finish this.

Task #4: Fault Models

Key:

- A,0 = a stuck at 0
- A,1 = a stuck at 1
- {1,0,0,1} = {a,b,c,d}



Test Vectors for each stuck at fault:

A,0 = {1,0,1,1}, {1,0,0,1}, {1,0,0,1}, {1,0,0,0}
A,1 = {0,0,1,1}, {0,0,0,1}, {0,0,0,1}, {0,0,0,0}
B,0 = {0,1,0,1}, {0,1,1,1}, {0,1,0,0}, {0,1,0,1}
B,1 = {1,1,0,1}, {1,1,1,1}, {1,1,0,0}, {1,1,0,1}
C,0 = {0,0,1,0}, {1,1,1,0}
C,1 = {0,0,0,0}, {1,1,0,0}
D,0 = {0,0,1,1}, {1,1,1,1}
D,1 = {0,0,1,0}, {1,1,1,0}
n1,0 = {1,0,1,1}, {1,0,0,1}, {1,0,0,1}, {1,0,0,0}
n1,1 = {0,0,1,1}, {0,0,0,1}, {0,0,0,1}, {0,0,0,0}
n2,0 = {1,1,1,1}, {0,0,0,1}, {1,1,1,0}, {0,0,1,0}, {0,0,1,1}
n2,1 = {1,0,1,1}, {1,0,0,1}, {0,1,1,0}, {0,1,1,0}, {0,1,1,1}

n3,0 = {1,1,1,1}, {0,0,0,1}, {1,1,1,0}, {0,0,1,0}, {0,0,1,1}, {1,1,0,1}, {1,1,0,0}, {0,0,1,1}, {0,0,0,0}
n3,1 = {1,0,1,1}, {1,0,0,1}, {0,1,1,0}, {0,1,1,1}, {1,1,1,0}, {0,0,1,0}
n4,0 = {1,1,0,1}, {1,1,1,1}, {1,1,0,0}, {0,0,0,1}, {0,0,1,1}, {0,0,0,0}
n4,1 = {1,1,1,0}, {0,0,1,0}
n5,0 = {0,0,0,0}, {1,1,0,0}
n5,1 = {0,0,1,0}, {1,1,1,0}

I didn't make a test bench for this one, and this is how far I got.

Task #5: Reflection

1. Think about at least two other faults that could occur during manufacturing (other than stuck-at). How would you detect them?

Another fault could be within the transition from a signal going from 0 -> 1 or 1 -> 0 or vice versa. I believe this is called a slow-to-rise/fall fault, and could be detected by rapidly switching inputs and verifying that the outputs are what you expect.
Another could be that part of the circuit is disconnected or severed from the rest of the circuit. You could detect this by testing inputs and neither 1 nor 0 inputs change the output of the circuit.

2. What are some of the trade-offs involved in having a scan chain (both benefits and downsides)?

Scan chains allow you to test post-silicon and allow you to easily activate and propagate test signals to specific parts of the circuit that may not be close or easily testable from the chip outputs. However, this requires another large part of your chip to be verified pre-silicon and will take up area on your chip which may be a problem if area may be a constraint.

3. Let's say you have some part of a circuit in which you don't want to include a scan chain or external testing points (for example, a cryptographic processor that you want to isolate from external access for security purposes). How can you ensure that you're not shipping potentially-broken silicon to an end-user?

You could have a BIST engine on board that tests that specific chip and has no other entry points to verify if that block is working properly. On top of that, you need to make sure that you are extremely thorough during pre-silicon verification.

4. The designs used in this assignment had the scan-chain inserted before any major optimizations were run. Can you think of any issues this may lead to, and how might you deal with them?

The scan chain might cause errors in the timing model of a certain circuit, so it is good to then verify that you still meet timing requirements with the scan chain and then optimize if there are violations.

Task #6: Project Work

Instead of spinning my wheels trying to finish the rest of this assignment, I decided to spend some quality time on implementation for the project. This week, I worked on the VGA controller module. Based on an incoming entity signal from the game state controller, this will draw a pixel of a certain color on the screen. So for example, if the current x,y is air, the game state controller would return 00, and then the VGA controller would draw r,g,b = {0,0,0} for that pixel. It is the job of the game state controller to keep track of where each entity is on the screen, and the job of the VGA controller is to draw those entities on the screen. The next big bulk of the project will be spent on designing the game state controller, which seems to be the most challenging module to implement (including the dynamic tail!).