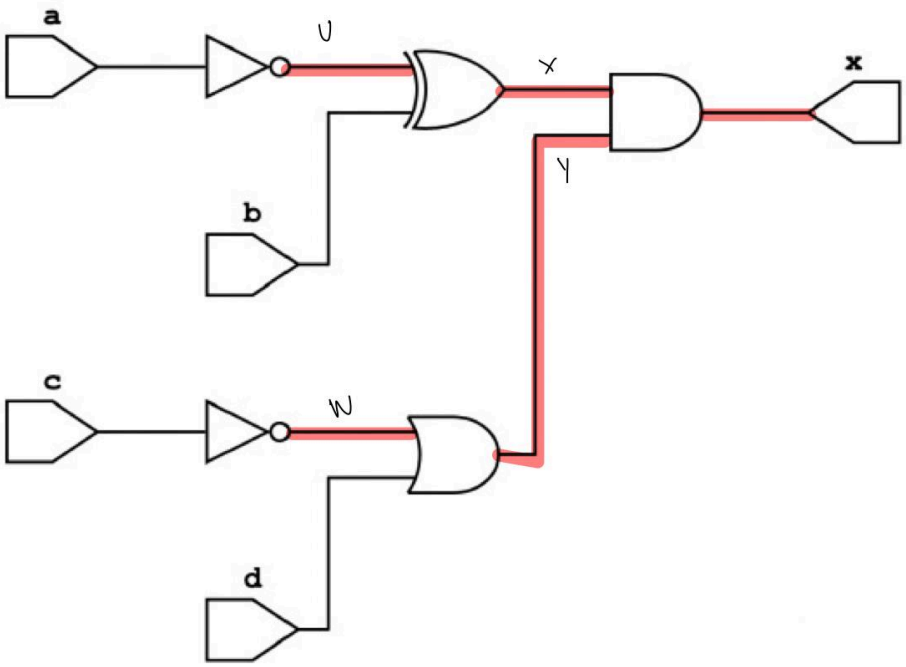
Mary Rose Rubino
EX9

Task 4:



Test Vectors: {a, b, c, d}

Wire U:

- Stuck at 0: {0, 0, 0 or 1, 1} → expected output is 1.
- Stuck at 1: {1, 1, 0 or 1, 1} → expected output is 1.

Wire W:

- Stuck at 0: {0, 0, 0, 0} or {1, 1, 0, 0} → expected output is 1.
- Stuck at 1: {0, 0, 1, 1} or {1, 1, 1, 1} → expected output is 1.

Wire X:

- Stuck at 0: {0, 0, 0 or 1, 1} or {1, 1, 0 or 1, 1} expected output is 1.
- Stuck at 1: {1, 0, 0 or 1, 1} or {0, 1, 0 or 1, 1} expected output is 0.

Wire Y:

- Stuck at 0: {0, 0, 0 or 1, 1} or {1, 1, 0 or 1, 1} expected output is 1.
- Stuck at 1: {0, 0, 1, 0} or {1, 1, 1, 0} expected output is 0.

fault1.sv:

{0, 0, 0, 1} found Wire U stuck at 0, Wire X stuck at 0, and Wire Y stuck at 0.
{0, 0, 0, 0} found Wire W is stuck at 0.

{0, 0, 1, 1} found Wire W is stuck at 1.
{0, 1, 0, 1} found Wire X is stuck at 1.

fault2.sv:

{1, 0, 0, 1} found Wire X is stuck at 1.
{0, 1, 0 1} found Wire X is stuck at 1.

fault3.sv:

Fault detected Wire U is stuck at 0, vector {0, 0, 0, 1}
Fault detected Wire U is stuck at 1, vector {1, 1, 0, 1}
Fault detected Wire W is stuck at 0, vector {0, 0, 0, 0}
Fault detected Wire W is stuck at 0, vector {1, 1, 0, 0}
Fault detected Wire W is stuck at 1, vector {0, 0, 1, 1}
Fault detected Wire W is stuck at 1, vector {1, 1, 1, 1}
Fault detected Wire X is stuck at 0, vector {0, 0, 0, 1}
Fault detected Wire X is stuck at 0, vector {1, 1, 0, 1}
Fault detected Wire Y is stuck at 0, vector {0, 0, 0, 1}
Fault detected Wire Y is stuck at 0, vector {1, 1, 0, 1}

fault4.sv:

Fault detected Wire X is stuck at 1, vector {1, 0, 0, 1}
Fault detected Wire X is stuck at 1, vector {0, 1, 0, 1}
Fault detected Wire Y is stuck at 1, vector {0, 0, 1, 0}
Fault detected Wire Y is stuck at 1, vector {1, 1, 1, 0}

fault5.sv:

No faults detected

I chose each of the vectors by starting with attempting to set the wire to test to the opposite value I thought it was stuck at. Then, I chose other inputs with the goal of making it so that the output of the final and gate was dependent on the result of that wire. In some cases, this meant making one input 1, and attempting to set the other input to 1, and in others it meant making the second input 1, and expected the output to be 0 when the other input is 0.

Task 5:

1. Think about at least two other faults that could occur during manufacturing (other than stuck-at). How would you detect them?
    a. An open circuit (two wires are disconnected), or a short circuit. Both of these could be detected in the same way we have been checking for the stuck at 0, or stuck at 1 errors: with crafted test vectors, and scan chain resisters.
2. What are some of the trade-offs involved in having a scan chain (both benefits and downsides)?
    a. Some benefits of having a scan chain are that it makes it much easier, and less expensive (when compared to other post-silicon testing methods) to test you design after it was manufactured. It can also be integrated with BIST. However, having a scan chain increases a design's complexity, as well as area and power consumption. There is also an added problem with potential manufacturing bugs within the scan chain itself.
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?
    a. This could be checked using BIST, where the results of the test are stored in an internal register that is not visible to the user. Also the chip could contain a secured JTAG interface.
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?
    a. It is possible that if further optimizations are made to a design (such as adding pipelining to meet timing), the scan chain will not be able to include all FFs in the design. In this case I would try to create more test vectors, that could show more information about the state of the new FFs or new wires within the design.

Task 6:

Status update:

I have started working on the RTL for the sub modules involved with my project. I have finished the inverse operations for the Mixcolumns module, the SubRows module, and I created the case statement for my sbox module. This week I plan on working on finishing the Scheduler module, and combining all of my submodules to create a single round module.