1. **How you implemented the BIST hardware**

***General***

My implementation of the BIST hardware utilized a finite state machine, counters, and signature hardware. The state machine takes 5 stages to cycle through a test and 1 stage to wait for initiation. Counters were used to monitor the number of scan chain shifts and the number of patterns applied, an LFSR was used to create the inputs, and an SISR was used to create and store the output signature respectively.

***I/O, Registers, and Hardware***

The BIST has externally 4 inputs, clk, rst, bistmode, and cut\_sdo, and 4 outputs, bistdone, bistpass, cut\_scanmode, and scut\_sdi. Internally, the BIST contains several registers to store the state of the machine—state register to hold the current state, counter to hold the current count for shifting, and pattern register to hold the current test pattern number. The LFSR hardware module takes an input value, reset value, clock, and outputs a 16-bit pseudorandom pattern to be used during scan in. The SISR hardware module takes an input value, reset value, clock, as well as a masking signal to allow shifts, and outputs a 16-bit signature. The LFSR and SISR uses the primitive polynomial *x16 + x5 + x4 + x3 + 1* and the SISR takes an input that is XORed along with the specified taps.

***Control Signals***

The BIST starts when it sees the posedge of rst and bistmode is set to one. This sets the test to be one which begins a full cycle of the state machine. To end a BIST before completion, reset can be set high while bistmode is set to 0, stopping and placing the circuit into its normal mode. The BIST changes values on the negedge of the clock to preload all values before they are latched by the circuit.

***State Machine***

When BIST mode is not activated, the BIST is idling in state 6 waiting for test register to be set to one to begin. State 6 resets the LFSR and SISR and waits to proceed to state 0. When the BIST starts, the BIST transitions to state 0 where the LFSR is started to generate random patterns. It then transitions to state 1 where the first scan chain is loaded with the 228 inputs using the first tap of the LFSR. State 1 adds one to the counter, sets the input of the scan chain to be shifted in, and a counter is used to count 228 times before proceeding to the next state. Once the full scan chain is shifted in, the BIST will transition to state 2 where the counter is reset to 0 and the CUT will be clocked under normal operations to get the output in the scan chain. In the next clock cycle the BIST will transition to stage 3 where the counter begins again and the output is shifted out into the SISR and a new pattern is scanned in at the same time (like stage 1, but with output). When counter counts 228 times and the new input pattern and output is completely scanned out, the BIST transitions to state 4 where the number of patterns is increased by 1 and the CUT is clocked under normal operation mode to get the new output values in the scan chain. At this point, if the number of patterns is less than 2000, then the BIST is placed back into state 3 to repeat states 3, 4, and 5 until 2000 patterns have been completed. When the number of patterns reach 2000, the BIST transitions to 5 where it raises the bistdone flag and checks the SISR for the correct fault-free signature and sets bistpass to 1 if the signature is correct or 0 if incorrect. When the bistdone fag is raised, the test signal is returned to 0 and the next state will be set back to 6 to wait for another test to be activated.

1. **What challenges you faced in getting the BIST hardware to work.**

A challenge faced when implementing the BIST hardware was getting the correct number of shifts of the scan chain as well as getting the correct output shifts and number of patterns. Another issue that was challenging was correctly shifting output of the scan chain into the SISR without corrupting the signature. The most difficult issue was getting the correct functionality for starting and stopping the BIST based on the rst, bistmode, and bistdone signature.

1. **An idea of how much hardware would be required to implement your design.**

The amount of hardware required to implement this BIST:

* LFSR Hardware:
  + 16 DFFs
  + 4 XOR gates
* SISR Hardware:
  + 16 DFFs
  + 5 XOR gates
  + 16 AND gates for clock masking to control next signal input
* State Machine Hardware
  + 24 DFFs
  + ~100 gates for signal generation (8 signals)

Approximately ~56 FFs and ~125 gates.