Combinational circuit:
Create two versions of the same circuit: one unmodified one and one who's gates have been minimized using boolean algebra/kmaps. We'll check if the fault coverages are the same or if the optimized circuit is less faulty than the unoptimized one (or vice versa).

Sequential Circuit:
If you've ever used the vending machines at school, you know they can be iffy. Sometimes the machines will eat your money. Other times, the machines will dispense an extra soda. One time, I got a soda AND my change back! So it appears we have three cases: 1. machine works 2. machine eats your money 3.machine glitches (in your favor).

For our project we'll build an fsm for a simple vending machine. Fault insertion will be used to determine which of the three cases above is more likely (this will be done by reading out the contents of the dffs). The random tvs will be used to represent the 'actions' taken with the machine.

ex.
A 9-bit test vector (e.g 001001000), will have three actions: 001 001 000
Assume soda costs $2
FSM will have 4 states (and thus 4 ffs)= insert money, $1 inserted, Dispense soda, refund money

Potential actions and their transitions:
No action: 000
Insert $1: 001
Insert $2 010
Insert $5 011
Coin release 100

According to our test vector, the actions taken will be:
No action (000), insert $1 (001), insert $1 (001)

Since $2 were inserted into the machine, the FSM will travel through insert money state, $1 inserted state, dispense soda state and then back to insert money state. If everything works well, these FFs (or at least one of them) will have a 1 in them. Inserting faults in the circuit should cause glitches like reaching the dispense soda state without inserting at least $2, or the dispense soda state being skipped when inserting $5. The purpose of all this is to "simulate" the soda machines at school and seeing if they're worth using or not.