**Problem 1**

These 9 vectors were selected using a greedy algorithm of all possible test vectors that got full coverage (vector signal order is Die22,Die21,Die20,Die12,Die11,Die10,StorePoint). Nodes not covered here are ones on the Q output of DFFs. The greedy part of this algorithm was that, using Excel, all possible configurations with exactly one stuck node were enumerated. Then, every possible input configuration was tested. The one that failed the most cases was taken as a fault vector (e.g. the vector with all 0’s input causes 17 such configurations to fail). Then, the next iteration would consider only the nodes that a previously found vector didn’t touch, and the input configuration that failed the most was again taken. This iteration occurred until all stuck nodes caused at least one failure. This assumed that P0, P1, P2 and P3, the Q outputs of DFFs, are all at 0. If they are stuck at 1, this part fails, but so will the vector that checks whether it is stuck at 1. I also goofed somewhere along the line, requiring me to use one more vector than necessary. To test robustness, every possible configuration of stuck/not stuck middle nodes was tested. Because this is O(2^n), only ~2% of all ~200,000,000 were tested, but they showed no failures. These vectors are not listed in the order that they were chosen with the greedy algorithm.

* 0,0,0,0,0,1,0
* 0,1,0,0,0,0,1
* 0,0,0,0,0,0,0
* 0,1,0,1,1,1,0
* 1,1,1,1,1,1,0
* 1,1,1,1,1,0,0
* 0,0,0,1,0,0,0
* 1,1,0,1,1,1,0
* 0,0,1,1,1,1,0
* 0,1,1,0,0,1,0

Then, using these, I did another greedy algorithm that found, when starting with 0’s on the Q outputs, which vectors made the most Q outputs change on StorePoint edge. This found that the following sequence would detect when the P\* nodes were stuck. To accommodate for this, we just move these vectors to the end.

* 0,0,0,0,0,1,0
* 0,0,0,0,0,1,1
* 1,1,1,1,1,1,0
* 1,1,1,1,1,1,1
* 0,0,0,0,0,0,0
* 0,0,0,0,0,0,1

To ensure that we start with all 0’s on the P nodes, which we require to properly test the rest of the system, we first clock the 0 vector, which corresponds to zeroed P nodes.

* 0,0,0,0,0,0,0
* 0,0,0,0,0,0,1

We had to add a vector to ensure that the output DiceEq11 wasn’t stuck at 0 (since all the vectors before now output 0)

* 0,1,1,1,1,0,0

**Problem 2**

**Reset width**. Note that on reset, we only care when the buffer goes from high to low. Then we propagate that signal.

Path: U22 HL -> U3 CLR to Qbar -> U17 HL -> U7 LH -> U18 B LH -> U2 D Setup

Best case time: 9140 ps

**Roll Setup**.

Path: U21 HL -> U10 LH -> U14 HL -> U3 D Setup

Best case time: 5640 ps

**Roll Hold**. Here we need the time for the roll signal change propagation to exceed the DFF hold time.

Path: U4 HL -> U1 D Hold

Best case time: 210 ps

**Clock**.

Path: U1 CLK to Q LH -> U17 HL -> U7 LH -> U18 B LH -> U2 D Setup

Best case time: 8410 ps (188.9 GHz)

**Output Delays**. Look at table.