Project 1

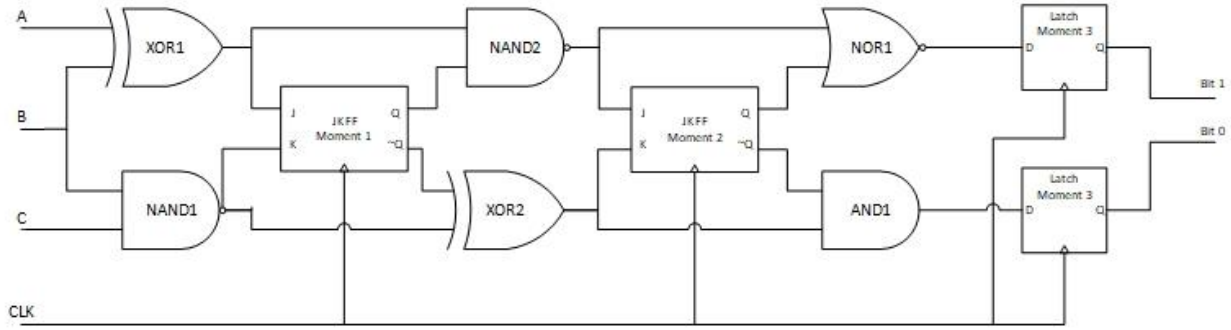
ECE 582
Abram Fouts
Mike Tian
January 30th, 2021

Procedure

Problem 1

Given the design constraints from the project requirements the circuit was design in a balanced fashion. We chose to do 2 gates on both sides of the gates to simplify the design phase. The circuit was modeled in C coding and when the code is showing which bit is high it shows the integer value of that bit position, for example *1 0 1* is *4 0 1*.

Here is the circuit that was designed:



A third moment was added to the circuit to show the circuit values in between each sequential part. An AND gate was added to create an even number of digital logic devices.

Here is the input to output from the circuit design:

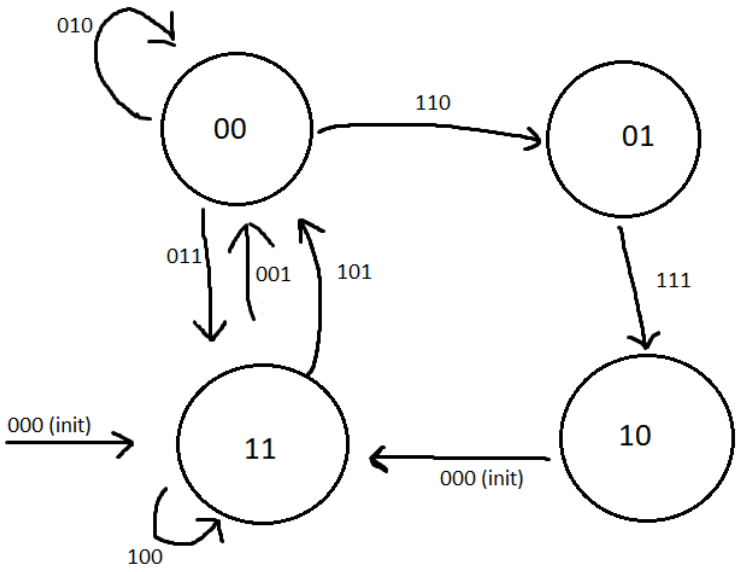
```
Input Bits: 0 0 0 :: S1 Bits: 2 1
Input Bits: 0 0 1 :: S1 Bits: 0 0
Input Bits: 0 2 0 :: S1 Bits: 0 0
Input Bits: 0 2 1 :: S1 Bits: 2 1
Input Bits: 4 0 0 :: S1 Bits: 2 1
Input Bits: 4 0 1 :: S1 Bits: 0 0
Input Bits: 4 2 0 :: S1 Bits: 0 1
Input Bits: 4 2 1 :: S1 Bits: 2 0
```

Note: The JK flipflops are put in reset mode for each input, so the Q = 0 to be safe during a J = 1, K = 1 situation.

Here is the table that corresponds to the inputs and the outputs of S1:

Input	Output
000	11
001	00
010	00
011	11
100	11
101	00
110	01
111	10

Here is the transition diagram:



After reviewing the transition diagram, we believe this circuit would act better as a primary 2 input circuit where bit C his permanently high until needed for specific cases. This generates all unique outputs instead of similar outputs like it does when it is a 3->2 encoder.

Problem 2

A product machine for S1 and the newly created S2 circuit was created. It is defined by generating a specified size array of each output from unique input (0-7) for S1 and S2. It was verified that both circuits are identical. *P* which is defined by the cartesian product *S1 x S2*. The state transition that was defined in Problem 1 is the corresponding transition diagram for S1 and S2 individually. It will be shown how the cartesian product is generated for *S1 x S2*. This is to simulate a state transition diagram fixed at an anchor point on S1-Bit1 and S2-Bit1. This generates a list or grouping of inputs output bit 1 value by every inputs output bit 0 value.

Here is the output of the console from the C program to model this circuit and the initial global state of P where S1 and S2 have the same state (input):

```
Input Bits: 0 0 0 :: S1 Bits: 2 1        Input Bits: 0 0 0 :: S2 Bits: 2 1
Input Bits: 0 0 1 :: S1 Bits: 0 0        Input Bits: 0 0 1 :: S2 Bits: 0 0
Input Bits: 0 2 0 :: S1 Bits: 0 0        Input Bits: 0 2 0 :: S2 Bits: 0 0
Input Bits: 0 2 1 :: S1 Bits: 2 1        Input Bits: 0 2 1 :: S2 Bits: 2 1
Input Bits: 4 0 0 :: S1 Bits: 2 1        Input Bits: 4 0 0 :: S2 Bits: 2 1
Input Bits: 4 0 1 :: S1 Bits: 0 0        Input Bits: 4 0 1 :: S2 Bits: 0 0
Input Bits: 4 2 0 :: S1 Bits: 0 1        Input Bits: 4 2 0 :: S2 Bits: 0 1
Input Bits: 4 2 1 :: S1 Bits: 2 0        Input Bits: 4 2 1 :: S2 Bits: 2 0
```

Here is the raw dump of all the states from P. Since there are only 4 output states a lot are going to be reused, and with a fixed *Bit 1* value the diagram can act as a mealy state machine where it can transition from *00 01 10 11* in any order at any moment.

```
There are 8 * 8 number of sets of the cartesian product of S1 x S2
With a focus on S1(x) as bit 1 S2(y) as bit 0

Grouping 0
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 0

Grouping 1
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 0

Grouping 2
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 0

Grouping 3
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 0

Grouping 4
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 0

Grouping 5
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 0

Grouping 6
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 0
P = S1 x S2: 0 1
P = S1 x S2: 0 0

Grouping 7
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 0
P = S1 x S2: 2 1
P = S1 x S2: 2 0
```

```
There are 8 * 8 number of sets of the cartesian product of S1 x S2
With a focus on S2(x) as bit 1 S1(y) as bit 0

Grouping 0                Grouping 1                Grouping 2
P = S1 x S2: 2 1          P = S1 x S2: 0 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 0 0          P = S1 x S2: 0 0
P = S1 x S2: 2 0          P = S1 x S2: 0 0          P = S1 x S2: 0 0
P = S1 x S2: 2 1          P = S1 x S2: 0 1          P = S1 x S2: 0 1
P = S1 x S2: 2 1          P = S1 x S2: 0 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 0 0          P = S1 x S2: 0 0
P = S1 x S2: 2 1          P = S1 x S2: 0 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 0 0          P = S1 x S2: 0 0

Grouping 3                Grouping 4                Grouping 5
P = S1 x S2: 2 1          P = S1 x S2: 2 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 2 0          P = S1 x S2: 0 0
P = S1 x S2: 2 0          P = S1 x S2: 2 0          P = S1 x S2: 0 0
P = S1 x S2: 2 1          P = S1 x S2: 2 1          P = S1 x S2: 0 1
P = S1 x S2: 2 1          P = S1 x S2: 2 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 2 0          P = S1 x S2: 0 0
P = S1 x S2: 2 1          P = S1 x S2: 2 1          P = S1 x S2: 0 1
P = S1 x S2: 2 0          P = S1 x S2: 2 0          P = S1 x S2: 0 0

Grouping 6                Grouping 7
P = S1 x S2: 0 1          P = S1 x S2: 2 1
P = S1 x S2: 0 0          P = S1 x S2: 2 0
P = S1 x S2: 0 0          P = S1 x S2: 2 0
P = S1 x S2: 0 1          P = S1 x S2: 2 1
P = S1 x S2: 0 1          P = S1 x S2: 2 1
P = S1 x S2: 0 0          P = S1 x S2: 2 0
P = S1 x S2: 0 1          P = S1 x S2: 2 1
P = S1 x S2: 0 0          P = S1 x S2: 2 0
```

If followed closely each grouping has a fixed bit 1 value from either S1 or S2.

Here shows the symbolic simulation being ran where n = 6, and the Boolean algebra for the two output bits is shown by:

*Bit 1: (‘(BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC)) (Q_2) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1)) (‘Q_2)) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1))*

*Bit 0: ‘(‘(BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC)) (Q_2) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1)) (‘Q_2)) + (BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC))*

Note: Q_1 and Q_2 are the initial state for the JK flip flops, and that is set to Q = 0 and ‘Q = 1.

```
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0

Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0

Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
```

```
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0

Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0

Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 0
```

Each block of text is a full input iteration which is represented by simulation _n_.

Here will show a modified gate in S2 which will final AND gate switched to an NOR gate. This only changes the Boolean algebra for one equation, which is for _Bit 0_, so instead of adding ‘Q_2 and XOR2 it will be multiplied then negated.

Bit 1: (‘(BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC)) (Q_2) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1)) (‘Q_2)) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1))

Bit 0: ‘(‘(BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC)) (Q_2) + (A’B + ’AB ‘((A’B + ‘AB)(‘Q_1) + BCQ_1)) (‘Q_2)) * (BC * ‘(A’B + ’AB (‘Q_1) + BCQ_1) + (A’B + ’AB (‘Q_1) + BCQ_1) ‘(BC))

Highlighted in yellow is the change to the Boolean equation.

Here is the output result showing the state space of P = S1 * S2

```
Input Bits: 0 0 0 :: S1 Bits: 2 1
Input Bits: 0 0 1 :: S1 Bits: 0 0
Input Bits: 0 2 0 :: S1 Bits: 0 0
Input Bits: 0 2 1 :: S1 Bits: 2 1
Input Bits: 4 0 0 :: S1 Bits: 2 1
Input Bits: 4 0 1 :: S1 Bits: 0 0
Input Bits: 4 2 0 :: S1 Bits: 0 1
Input Bits: 4 2 1 :: S1 Bits: 2 0
```

```
Input Bits: 0 0 0 :: S2 Bits: 0 1
Input Bits: 0 0 1 :: S2 Bits: 0 1
Input Bits: 0 2 0 :: S2 Bits: 0 1
Input Bits: 0 2 1 :: S2 Bits: 0 1
Input Bits: 4 0 0 :: S2 Bits: 0 1
Input Bits: 4 0 1 :: S2 Bits: 0 1
Input Bits: 4 2 0 :: S2 Bits: 0 1
Input Bits: 4 2 1 :: S2 Bits: 0 1
```

```
There are 8 * 8 number of sets of the cartesian product of S1 x S2
With a focus on S1(x) as bit 1 S2(y) as bit 0
```

```
Grouping 0
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1

Grouping 2
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 4
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1

Grouping 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 3
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1

Grouping 5
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 6
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 7
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
P = S1 x S2: 2 1
```

There are 8 * 8 number of sets of the cartesian product of S1 x S2
With a focus on S2(x) as bit 1 S1(y) as bit 0

```
Grouping 0
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 2
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 4
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 3
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1

Grouping 5
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
```

```
Grouping 6
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
```

```
Grouping 7
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
P = S1 x S2: 0 1
```

The change in S2 caused t he output to get stuck, so this was not a positive change for the circuit.

Here is the equivalent simulation for P = S1 x S2 where the number of input cycle throughs is N = 6.

```
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3

Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3

Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3
```

```
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3

Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3

Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 2
Equivalence Simulation for S1 ^ S2: 1
Equivalence Simulation for S1 ^ S2: 0
Equivalence Simulation for S1 ^ S2: 3
```

Above shows how the circuits are no longer equivalent by the XOR result not being 0. There is a single case where this is the case but out of all 8 inputs there is only one. The integers as the result from the XORing of the circuits is represented in base 10, but it still shows the inequality in base 10 or binary. All in all, the change in the circuit design did not help the circuit.