Anthony Bugatto

EE 421: Digital Electronics

Project 2: Finite State Machine

November 29, 2018

For this project we created a Finite State Machine that recognized the 4 bit patterns: 0011, 1111, and 1001. In order to achieve this, I designed a Moore machine with 4 DFF and encoded 12 states to keep track of the data flow. The states are shown in figure 1, the state table is shown in figure 2, the truth table is shown in figure 3, and the state diagram is shown in figure 4.

State Names	Pattern	Output
A	0	0
B	00	0
C	001	0
D	0011	1
E	1	0
F	10	0
G	100	0
H	1001	1
I	11	0
J	111	0
K	1111	1
RESET	-	RST

Figure 1: States and Encodings

Present State: $y_3y_2y_1y_0$	Next State: $Y_3Y_2Y_1Y_0$		Output:
	w = 0	w = 1	z
RESET	A	E	0
A	B	A	0
B	A	C	0
C	F	D	0
D	F	J	1
E	F	I	0
F	G	E	0
G	B	H	0
H	F	D	1
I	F	J	0
J	F	K	0
K	F	K	1

Figure 2: State Table

Present State: $y_3y_2y_1y_0$	Next State: $Y_3Y_2Y_1Y_0$		Output: z
	w = 0	w = 1	
RESET	0000	0100	0
0000	0001	0000	0
0001	0000	0010	0
0010	0101	0011	0
0011	0101	1001	1
0100	0101	1000	0
0101	0110	0100	0
0110	0001	0111	0
0111	0101	0011	1
1000	0101	1001	0
1001	0101	1010	0
1010	0101	1010	1

Figure 3: Truth Table

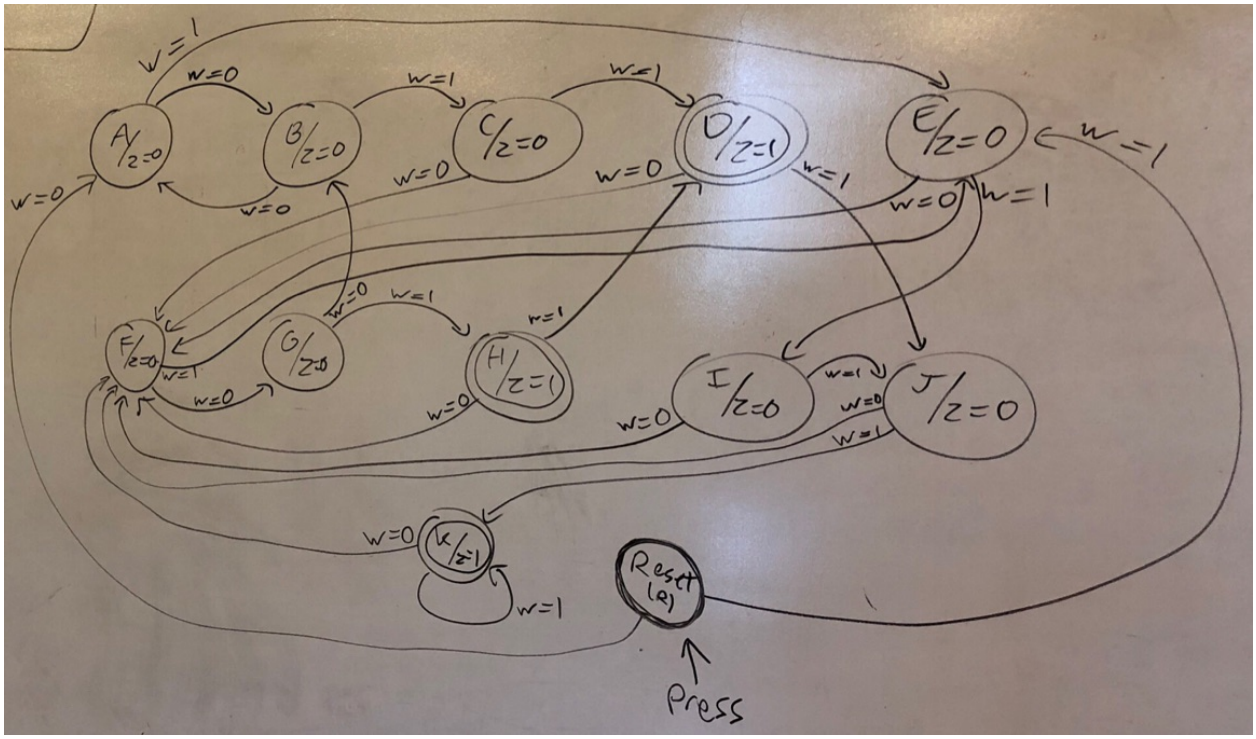

Figure 4: State Diagram

During testing the design was verified by working on the sequence in figure 5. The code is in Appendix A and the simulations are in Appendix B.

w	1	0	0	1	1	0	0	1	1	1	1	0
z	0	0	0	0	1	1	0	0	1	1	0	1

Figure 5: Verification Sequence

The schematic is shown in figure 6.

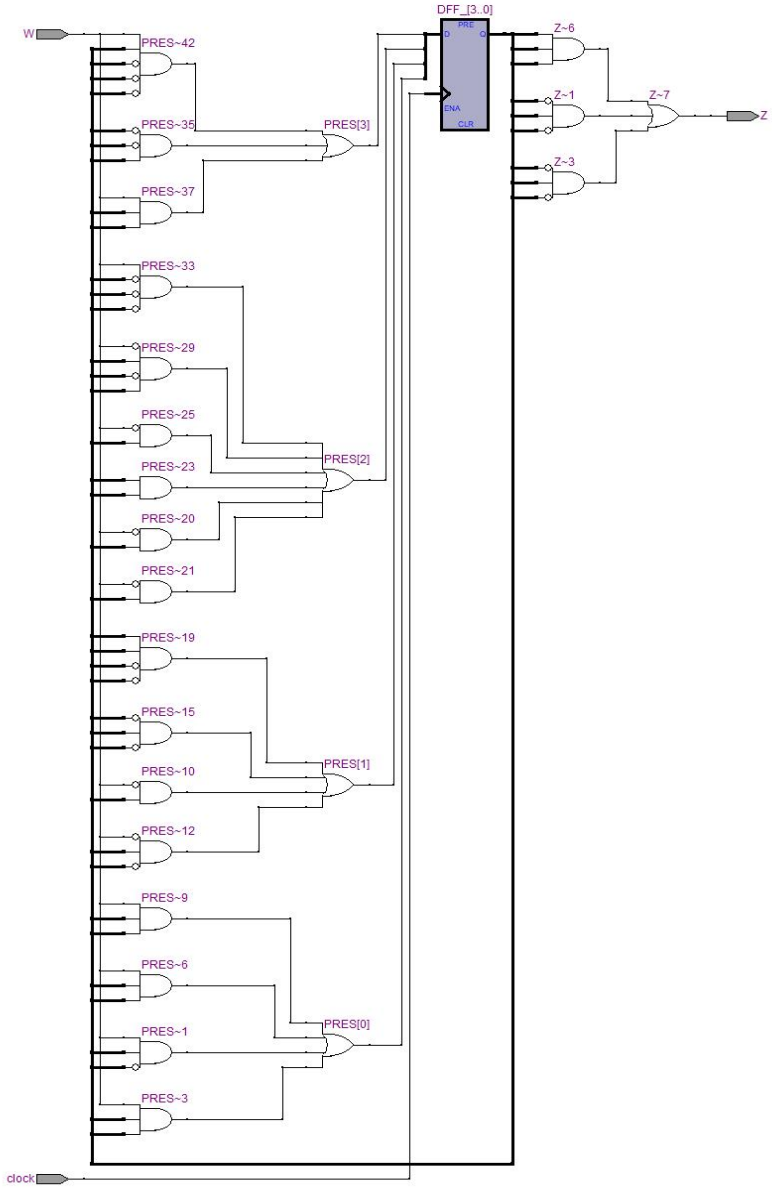

Figure 6: State Machine Schematic

Appendix A: VHDL Code

Using Behavioral Design:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.FF_package.ALL;

ENTITY state_machine IS
	PORT(
			clock, reset, W : IN STD_LOGIC;
			Z : OUT STD_LOGIC
	);
END state_machine;

ARCHITECTURE behavioral OF state_machine IS
	TYPE STATE_TYPE IS (A, B, C, D, E, F, G, H, I, J, K, R);
	SIGNAL state : STATE_TYPE;

	BEGIN
		PROCESS(clock)
			BEGIN
				IF reset = '1' THEN
					state <= R;
				ELSIF clock'EVENT AND clock = '1' THEN --RISING EDGE
					CASE state IS
						WHEN R =>
							IF W = '0' THEN
								state <= A;
							ELSE
								state <= E;
							END IF;
						WHEN A =>
							IF W = '0' THEN
								state <= B;
							ELSE
								state <= A;
							END IF;
						WHEN B =>
							IF W = '0' THEN
								state <= A;
							ELSE
								state <= C;
							END IF;
```

```
WHEN C =>
        IF W = '0' THEN
                state <= F;
        ELSE
                state <= D;
        END IF;
WHEN D =>
        IF W = '0' THEN
                state <= F;
        ELSE
                state <= J;
        END IF;
WHEN E =>
        IF W = '0' THEN
                state <= F;
        ELSE
                state <= I;
        END IF;
WHEN F =>
        IF W = '0' THEN
                state <= G;
        ELSE
                state <= E;
        END IF;
WHEN G =>
        IF W = '0' THEN
                state <= B;
        ELSE
                state <= H;
        END IF;
WHEN H =>
        IF W = '0' THEN
                state <= F;
        ELSE
                state <= D;
        END IF;
WHEN I =>
        IF W = '0' THEN
                state <= F;
        ELSE
                state <= J;
        END IF;
WHEN J =>
        IF W = '0' THEN
```

```
                                                state <= F;
                                        ELSE
                                                state <= K;
                                        END IF;
                                WHEN K =>
                                        IF W = '0' THEN
                                                state <= F;
                                        ELSE
                                                state <= K;
                                        END IF;
                        END CASE;
                END IF;
        END PROCESS;

        Z <= '1' WHEN state = D OR state = H OR state = K ELSE '0';
END behavioral;
```

Using RTL Design:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.FF_package.ALL;

ENTITY state_machine IS
        PORT(
                        clock, W : IN STD_LOGIC;
                        Z : OUT STD_LOGIC
        );
END state_machine;

ARCHITECTURE structural OF state_machine IS
        SIGNAL PRES : STD_LOGIC_VECTOR(3 DOWNTO 0); --Y
        SIGNAL NXT : STD_LOGIC_VECTOR(3 DOWNTO 0); --y

        BEGIN
                -- Y(0) = y(1)y(2)'W + y(1)y(3)W + y(0)y(3)W + y(0)y(2)W
                PRES(0) <= (NXT(1) AND NOT NXT(2) AND W) OR (NXT(1) AND NXT(3) AND W) OR
(NXT(0) AND NXT(3) AND W) OR (NXT(0) AND NXT(2) AND W);

                -- Y(1) = y(0)W' + y(1)y(2)'W' + y(1)y(2)y(3)' + y(0)y(2)'y(3)' + y(0)'y(1)'y(2)y(3)
                PRES(1) <= (NXT(0) AND NOT W) OR (NXT(1) AND NOT NXT(2) AND NOT W) OR
(NXT(0) AND NOT NXT(2) AND NOT NXT(3)) OR (NOT NXT(0) AND NOT NXT(1) AND NXT(2) AND
NXT(3));
```

```
		-- Y(2) = y(1)W' + y(3)W' + y(0)y(3) + y(0)W' + y(1)'y(2)y(3)'W + y(0)'y(2)'y(3)'W
		PRES(2) <= (NXT(1) AND NOT W) OR (NXT(3) AND NOT W) OR (NXT(0) AND
NXT(3)) OR (NXT(0) AND NOT W) OR (NOT NXT(1) AND NXT(2) AND NXT(3) AND NOT W)
					OR (NOT NXT(0) AND NOT NXT(2) AND NOT NXT(3) AND
W);

		-- Y(3) = y(0)'y(2)y(3)' + y(0)y(2)W + y(0)'y(1)'y(3)'W' + y(0)'y(2)'y(3)W
		PRES(3) <= (NOT NXT(0) AND NXT(2) AND NOT NXT(3)) OR (NXT(0) AND NXT(2)
AND W) OR (NOT NXT(0) AND NOT NXT(1) AND NOT NXT(2) AND NXT(3) AND W);

		DFF_0: DFF PORT MAP(clock, PRES(0), NXT(0));
		DFF_1: DFF PORT MAP(clock, PRES(1), NXT(1));
		DFF_2: DFF PORT MAP(clock, PRES(2), NXT(2));
		DFF_3: DFF PORT MAP(clock, PRES(3), NXT(3));

		-- Z = y(1)y(2)'y(3)' + y(0)y(2)'y(3)' + y(0)y(2)y(3)
		Z <= (NXT(1) AND NOT NXT(2) AND NOT NXT(3)) OR (NXT(0) AND NOT NXT(2)
AND NOT NXT(3)) OR (NXT(0) AND NXT(2) AND NXT(3));
END structural;
```

DFF Package:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DFF IS
	PORT(
			CLK, D : IN STD_LOGIC;
			Q : OUT STD_LOGIC
	);
END DFF;

ARCHITECTURE behavior OF DFF IS
	BEGIN
		PROCESS(CLK)
			BEGIN
				IF CLK'EVENT AND CLK = '1' THEN --rising edge
					Q <= D;
				END IF;
		END PROCESS;
END behavior;
```

```
--declare package
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE FF_package IS
	COMPONENT DFF
		PORT(
				CLK, D : IN STD_LOGIC;
				Q : OUT STD_LOGIC
		);
	END COMPONENT;
END FF_package;
```

Appendix B: Simulation