Układy cyfrowe i systemy wbudowane - laboratorium

Karol Kulawiec 241281
Bartosz Rudnikowicz 241382

18.11.2019

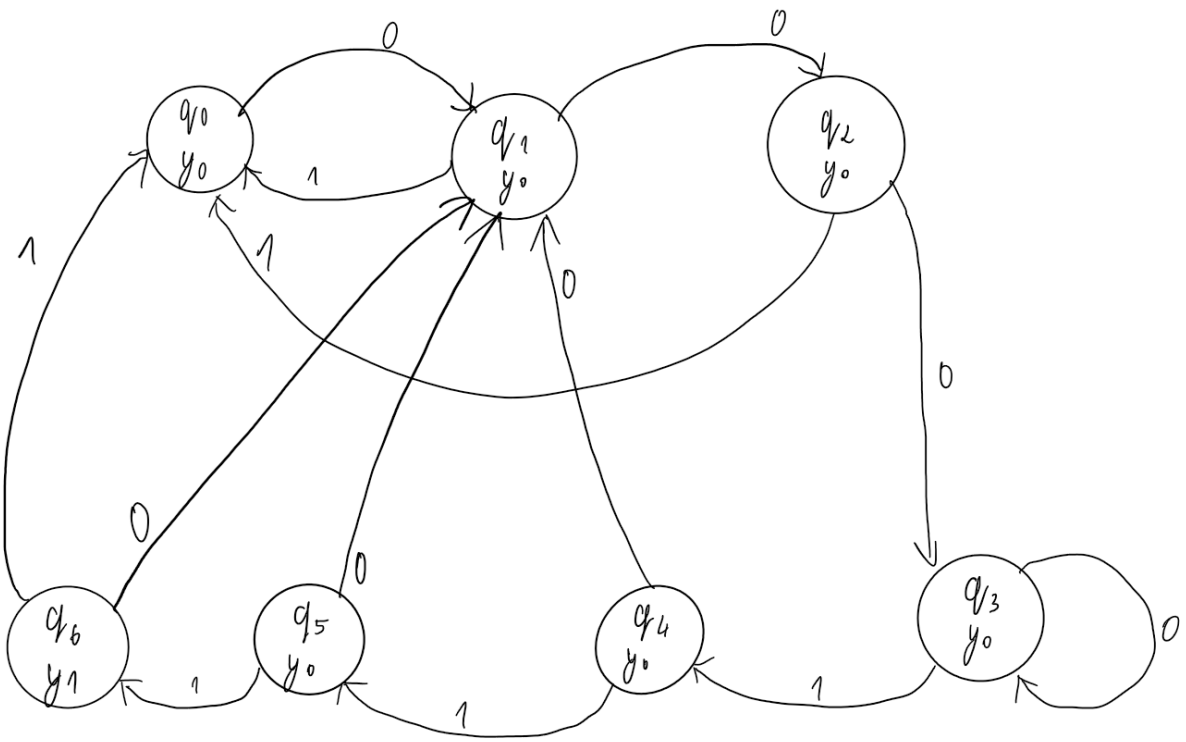
1 Wstęp

Naszym celem było stworzenie detektora sekwencji 6-bitowej, reprezentowanego jako automat Moore'a oraz Mealy. Następnie zasymulowanie go oraz zaimplementowanie sprzętowo.
Sekwencja, którą miał wykrywać automat to: **000111**

2 Przebieg zajęć

2.1 Zadanie 1 - Realizacja automatu korzystająca z przerzutników

Pierwszym zadaniem było wykonanie detektora w języku VHDL, korzystając z trzech przerzutników, zasymulowanie go na 18-bitowej sekwencji oraz sprzętowa implementacja na zestawie laboratoryjnym. Najpierw stworzyliśmy graf automatu Moore'a oraz tabelę przejść.



Rysunek 1: Graf automatu Moore'a wykrywającego sekwencje 000111.

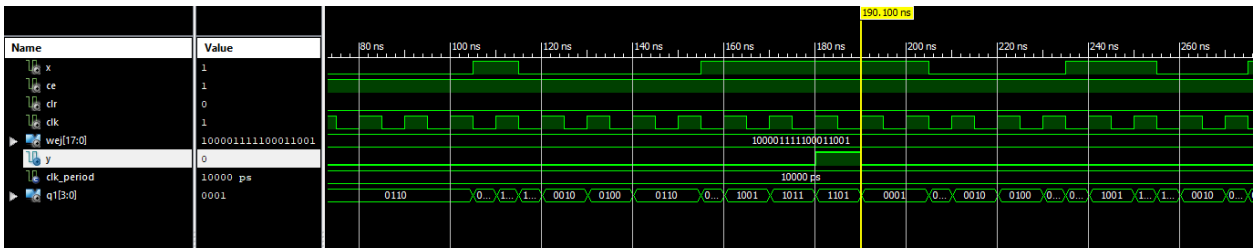
t				t+1			
Q2	Q1	Q0	X	Q2	Q1	Q0	Y
0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	1	1	0
0	1	0	1	0	0	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0
1	0	0	0	0	0	1	0
1	0	0	1	1	0	1	0
1	0	1	0	0	0	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	0
1	1	0	1	0	0	0	0

Tabela 1: Tabela przejść.

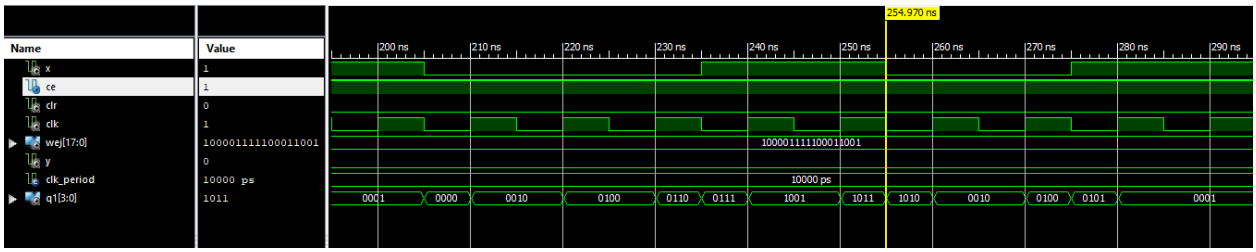
Na tej podstawie stworzyliśmy architekturę widoczną na listingu poniżej. Sygnał wewnętrzny Q1 jest sklejeniem sygnałów Q, kodujących stany jako liczby binarne oraz sygnału X, który jest sygnałem wejściowym.

```
architecture Behavioral of sekwencja is

        signal D : STD_LOGIC_VECTOR (2 downto 0);
        signal Q :  STD_LOGIC_VECTOR (2 downto 0);
        signal Q1 :  STD_LOGIC_VECTOR (3 downto 0);

begin
        FDCE2 : FDCE port map (Q(2), CLK, CE, CLR, D(2));
        FDCE1 : FDCE port map (Q(1), CLK, CE, CLR, D(1));
        FDCE0 : FDCE port map (Q(0), CLK, CE, CLR, D(0));

        Q1 <= Q & X;
        Y <= '1' when Q = O"6" else '0';

        with Q1 select
        D <=    O"1" when X"0",
                O"0" when X"1",
                O"2" when X"2",
                O"0" when X"3",
                O"3" when X"4",
                O"0" when X"5",
                O"3" when X"6",
                O"4" when X"7",
                O"1" when X"8",
                O"5" when X"9",
                O"1" when X"A",
                O"6" when X"B",
                O"1" when X"C",
                O"0" when X"D",
                O"0" when others;

end Behavioral;
```

Następnie dla sekwencji testowej 10**000111**1100011001 przeprowadziliśmy symulację. Na Rysunku 2 widać poprawne wykrycie sekwencji po wprowadzeniu trzeciej jedynki z rzędu, co zostało zaznaczone. Rysunek 3 pokazuje natomiast sekwencje 000110, która różni się tylko ostatnią cyfrą od poprawnej sekwencji. Na zaznaczonym fragmencie widać, że detektor zadziałał poprawnie i wrócił do stanu q_1 odpowiadającemu wprowadzeniu jednego, początkowego zera.



Rysunek 2: Poprawne wykrycie sekwencji 000111.



Rysunek 3: Poprawne uniknięcie pułapki.

Na koniec detektor został zaimplementowany na układzie laboratoryjnym. Wyjście zostało przypisane do diody LED(0), przycisk K0 był wejściem X, a K1 sygnałem CE. Pod przycisk K7 został przypisany reset. Po wgraniu, wszystko działo prawidłowo.

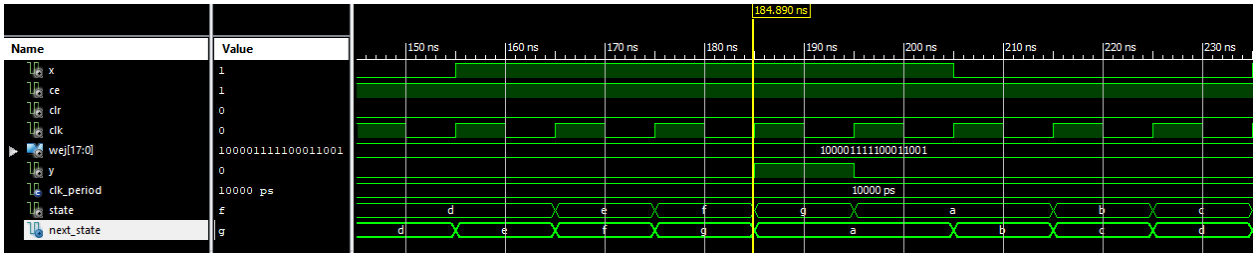
2.2 Zadanie 2 - realizacja automatu z użyciem szablonu

Kolejnym zadaniem było zrealizowanie tego samego automatu, korzystając z szablonu zaprezentowanego na wykładzie, który umożliwia sprzętową implementacje. Kolejne litery alfabetu odpowiadają kolejnym stanom z grafu. Stan q_0 - A, q_1 - B (...) q_6 - G.
Ponieważ process1 jest zawsze niemal identyczny jak ten z szablonu prezentowanego na wykładzie, poniżej znajduje się tylko process2 który odpowiada za zmiany stanów. Pod procesem znajduje się również przypisanie sygnału wyjścia.

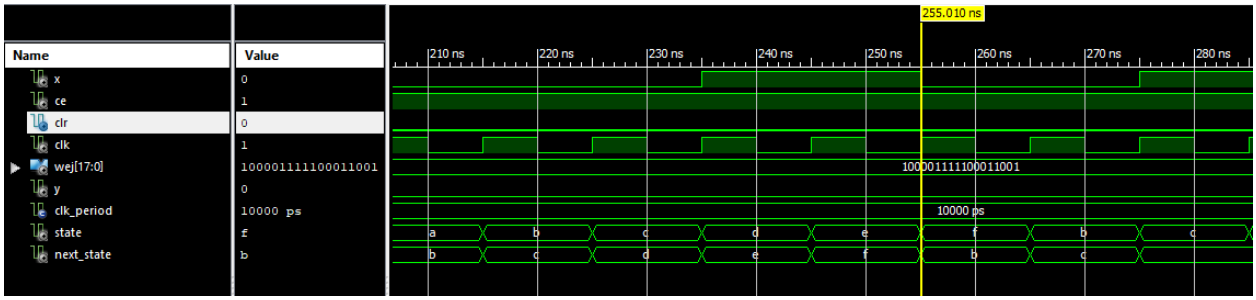
```
process2: process( state, X)
begin
next_state <= state;

case state is

when A =>
        if X = '0' then
                next_state <= B;
        end if;

when B =>
        if X = '0' then
                next_state <= C;
        else
                next_state <= A;
        end if;
```

```
            when C =>
                    if X = '0' then
                            next_state <= D;
                    else
                            next_state <= A;
                    end if;

            when D =>
                    if X = '1' then
                            next_state <= E;
                    end if;

            when E =>
                    if X = '0' then
                            next_state <= B;
                    else
                            next_state <= F;
                    end if;

            when F =>
                    if X = '0' then
                            next_state <= B;
                    else
                            next_state <= G;
                    end if;

            when G =>
                    if X = '0' then
                            next_state <= B;
                    else
                            next_state <= A;
                    end if;

            end case;
    end process process2;

    Y <= '1' when state = G else '0';
```

Następnie, dla tej samej sekwencji testowej co w zadaniu poprzednim, przeprowadziliśmy symulację. Na rysunkach poniżej widać poprawne działanie detektora.



Rysunek 4: Poprawne wykrycie sekwencji 000111.

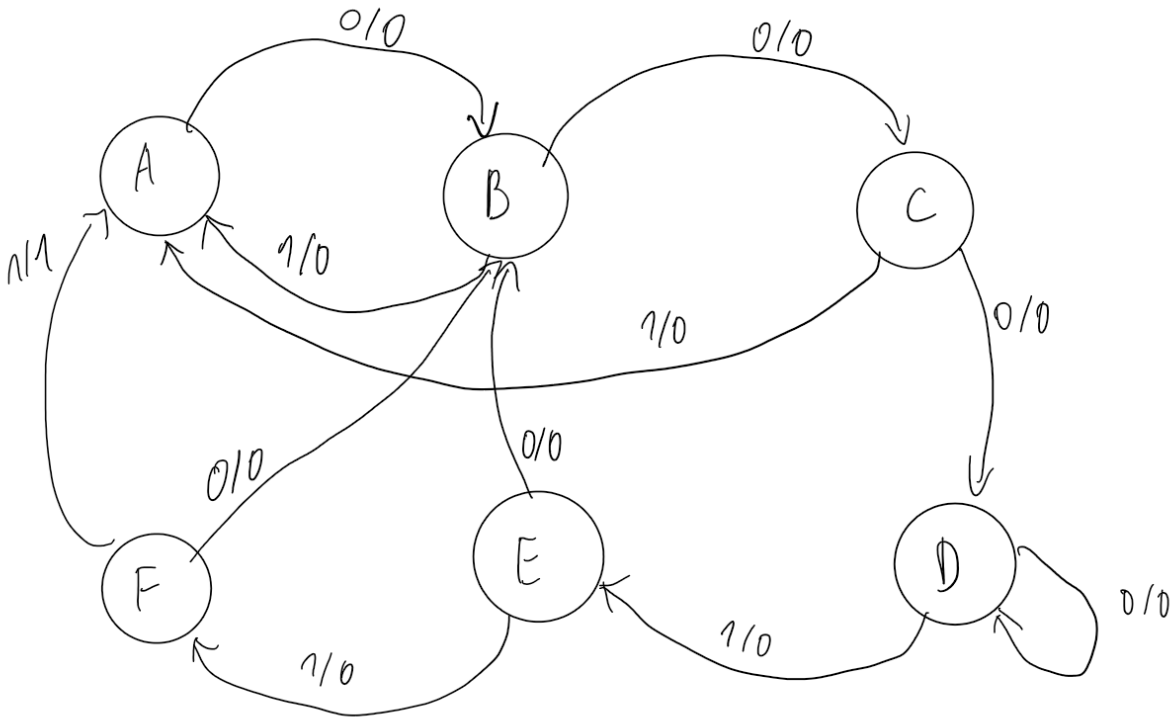


Rysunek 5: Poprawne uniknięcie pułapki.

Na koniec detektor został zaimplementowany na układzie laboratoryjnym tak jak w zadaniu poprzednim. Po wgraniu, wszystko działo prawidłowo.

2.3 Zadanie 3 - automat Mealy

Ostatnim zadaniem było wykonanie tego samego detektora w formie automatu Mealy. Poniżej przedstawiony jest graf automatu. Oznaczenia na krawędziach oznaczają X/Y.



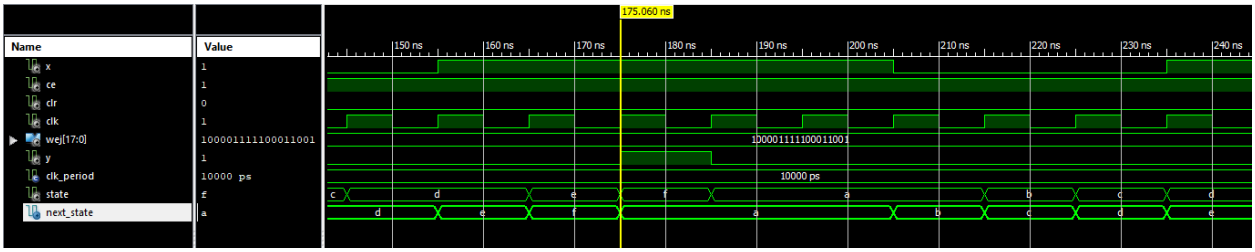
Rysunek 6: Graf automatu Mealy'ego wykrywającego sekwencje 000111.

Process2 w automacie Mealy'ego różni się jedynie końcówką od process2 w automacie Moore'a, dlatego aby nie powielać treści, poniżej zostanie zamieszczona jedynie różnica:

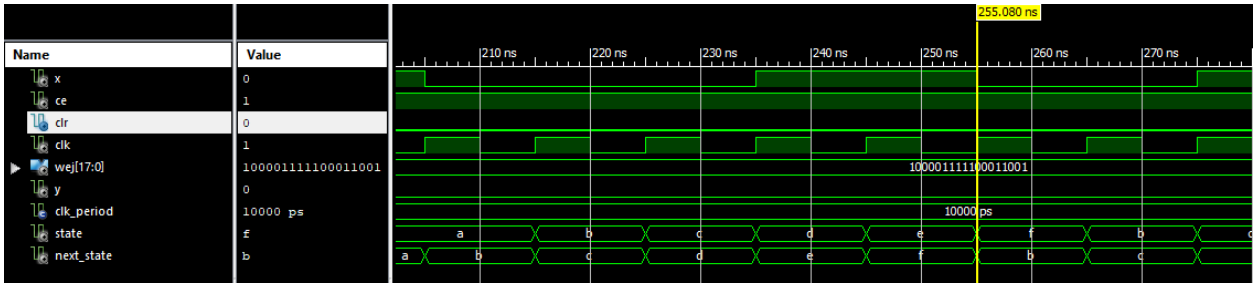
```
when F =>
        if X = '0' then
                next_state <= B;
        else
                next_state <= A;
        end if;
```

```
        end case;
end process process2;

        Y <= '1' when state = F and X = '1'
        else '0';
```

Ponieważ symulacja nie została przeprowadzona podczas zajęć, została przez nas dokończona w domu. Dla tej samej sekwencji testowej co w poprzednich zadaniach, przeprowadziliśmy symulację. Na rysunkach poniżej widać poprawne działanie detektora.



Rysunek 7: Poprawne wykrycie sekwencji 000111.



Rysunek 8: Poprawne uniknięcie pułapki.

Z powodu braku fizycznego układu w domu, implementacja na układzie nie została przeprowadzona.

3 Podsumowanie

Podczas zajęć zaprojektowaliśmy, zasymulowaliśmy i zaimplementowaliśmy na zestawie laboratoryjnym ZL-9572 pierwsze dwa zadania. Z powodu nieskończenia zadania 3 podczas zajęć, symulację automatu Mealy'ego przeprowadziliśmy w domu.