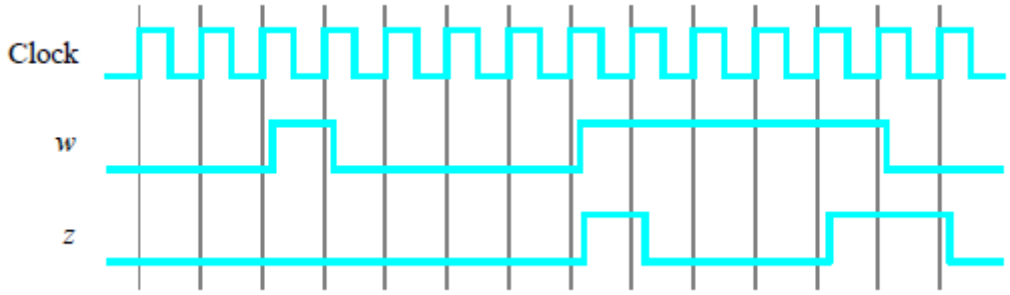
Lab. 8. Automaty skończone

Wymagana wiedza

- **Verilog**: operatory **casex**, **casez**, atrybuty **full_case**, **parallel_case**, operatory proceduralne, parametry lokalne;
- Definicja automatów skończonych, automatów Mealy, Moore (struktura, zasada działania), reprezentacja automatów skończonych;
- bezpośrednia realizacja automatów skończonych w przypadku jednoargumentowego kodowania, zerowy początkowy kod stanu;
- style opisu automatów skończonych (3 style);
- kodowanie stanów wewnętrznych automatów skończonych, kodowanie domyślne **CAD Quartus**.

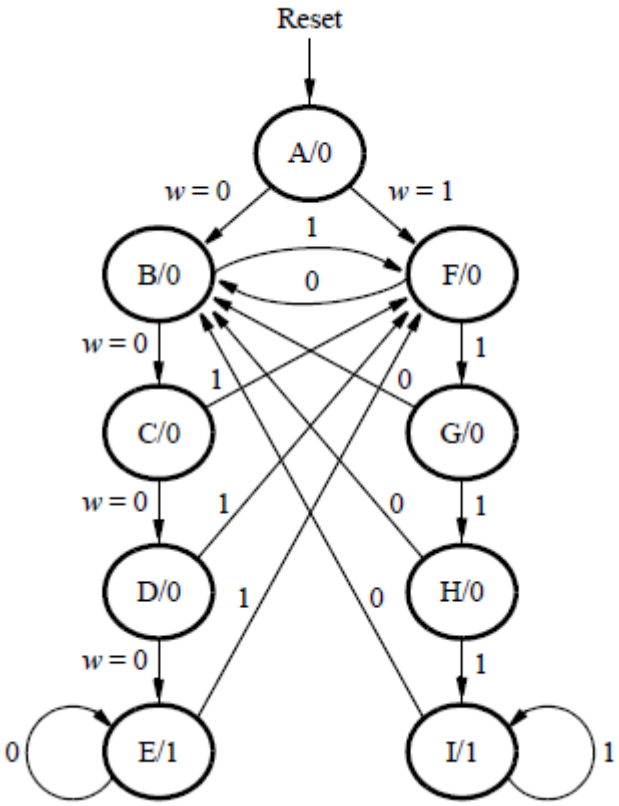
Wykonanie

1. Zaimplementuj skończoną maszynę stanów, która na wejściu **w** rozpoznaje sekwencje czterech jedynek lub czterech zer. Za każdym razem, gdy taka sekwencja zostanie znaleziona, wynikiem jest **z = 1**, w przeciwnym przypadku **z = 0**. Dozwolone są sekwencje nakładające się, czyli jeśli **z = 1** po czterech jedynkach (zerach), to **z** będzie 1 i po pięciu, sześciu, itd. jedynkach (zerach). Przykład relacji między w i z pokazano na rys. 1.



Rys. 1. Wymagana synchronizacja dla wyjścia z

Diagram stanu (graf automatu) pokazano na rys. 2.



Rys. 2. Diagram stanów dla FSM

Utwórz projekt FSM_one_hot skończonej maszyny stanów w przypadku kodowania unarnego (one-hot).

Tabela 1. Kody one-hot dla FSM

Name	State Code $y_8y_7y_6y_5y_4y_3y_2y_1y_0$
A	000000001
B	000000010
C	000000100
D	000001000
E	000010000
F	000100000
G	001000000
H	010000000
I	100000000

Używaj tylko prostych operatorów przypisania do określania wartości na wejściach przerzutników (nie możesz używać instrukcji case i if). Przetestuj projekt za pomocą następującego przypisania pinów:

Wyprowadzenia	**Piny**
w	**SW1**
clk	**KEY0**
aclr	**SW0**
z	**LEDR9**

y[8:0]	LEDR[8:0]

gdzie **y[8:0]** – Wyjścia przerzutników definiujące kody stanu automatu skończonego.

2. Wykonaj skończoną maszynę stanów z rys. 2 (projekt FSM_user_coding) używając instrukcji case i if (jeśli to konieczne). Przypisz następujące kody do stanów wewnętrznych:

Tabela 3. Kody binarne dla FSM

Name	State Code $y_3y_2y_1y_0$
A	0000
B	0001
C	0010
D	0011
E	0100
F	0101
G	0110
H	0111
I	1000

W **Quartus** ustaw parametr **Assignment** > **Settings** > **Compiler Settings** > **Advanced Settings** (**Synthesis**) > **State Machine Processing** > **User Encoded**, aby **Quartus** użył kodów stanu określonych przez użytkownika w kodzie projektu.

3. Porównaj dwa projekty **FSM_one_hot** i **FSM_user_coding** (zad. 1 i 2) względem kosztu realizacji i szybkości.

4. Utwórz 3 implementacje automatu skończonego z zad. 2: z trzema procesami, dwoma procesami i jednym procesem. Przeanalizuj wszystkie trzy implementacje. Porównaj je względem kosztu realizacji i szybkości. Porównaj także wyniki syntezy automatów skończonych: **Tools** > **Netlist Viewers** > **State Machine Viewer**.

5. Utwórz projekt detector_with_shift_reg, który implementuje detektor sekwencji na rejestrach przesuwnych. W tym celu należy utworzyć dwa 4-bitowe rejestry przesuwne, jeden do rozpoznawania sekwencji jedynek, a drugi do zer. Dołącz niezbędne wyrażenia logiczne do projektu, aby utworzyć wyjście **z**. Porównaj projekt z poprzednimi implementacjami.

6. Utwórz projekt FSM_Morse_code, który wysyła kody alfabetu Morse'a do diody LEDR0 dla pierwszych 8 liter alfabetu łacińskiego.

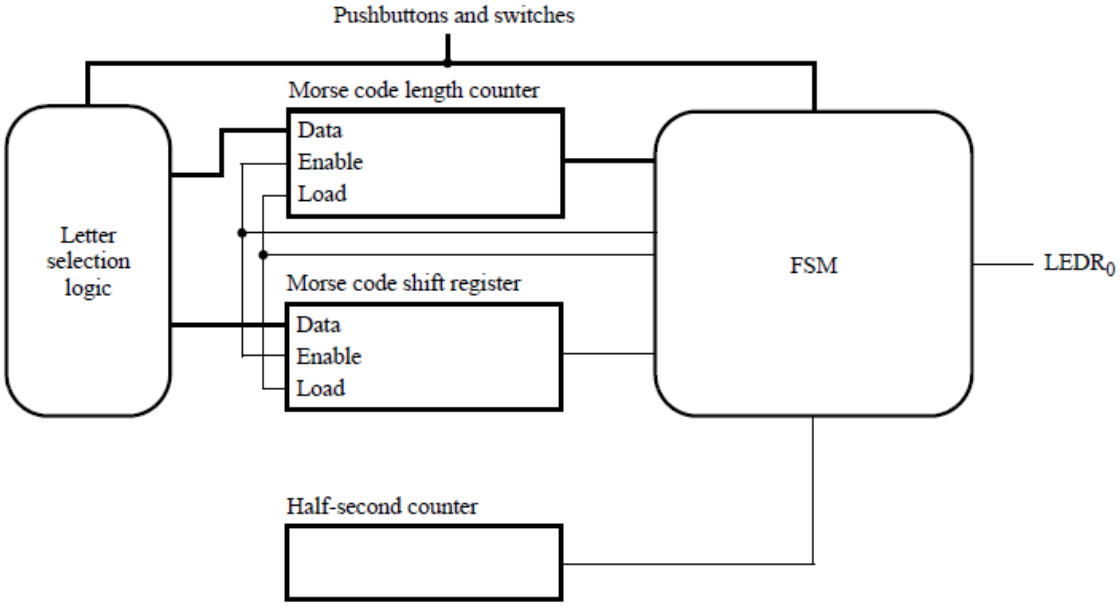
Kod litery określa przełącznik SW2-0.

SW 210	**Litery**	**Kod Morse’a**
000	**A**	. –
001	**B**	–…
010	**C**	–.–.
011	**D**	–..
100	**E**	.

101	**F**	.. –.
110	**G**	– –.
111	**H**	….

gdzie. - odstęp 0,5 s; - - przerwa 1,5 s; 0,5 s - odstępy między kropkami (.) i kreskami (-). Kod litery zaczyna pojawiać się na diodzie LED0 po naciśnięciu klawisza KEY1. Przycisk KEY0 służy jako reset asynchroniczny.

Wskazówka: Użyj liczników do generowania impulsów 0,5 i 1,5 sekundy. Ewentualny schemat strukturalny projektu pokazano na rys. 3.



Rys. 3. Schemat blokowy projektu **Morse_code**