Wydział	Imię i nazwisko		Rok	Grupa	Zespół
	1. Michał Rogowski				
WFiIS	2. Ihnatsi Yermakovich		II	10	02
PRACOWNIA ELEKTRONICZNA WFiIS AGH	Temat				Nr ćwiczenia
	Maszyny stanów skończonych				08
Data wykonania	Data oddania	Zwrot do poprawy	Data oddania	Data zaliczenia	OCENA
09.06.2022	19.06.2022				

Maszyny stanów skończonych

Ćwiczenie nr 08

Michał Rogowski

Ihnatsi Yermakovich

1 Cel ćwiczenia

Celem ćwiczenia było zaprojektowanie maszyn stanów skończonym (FSM) realizujących określone funkcjonalności.

2 Wyniki

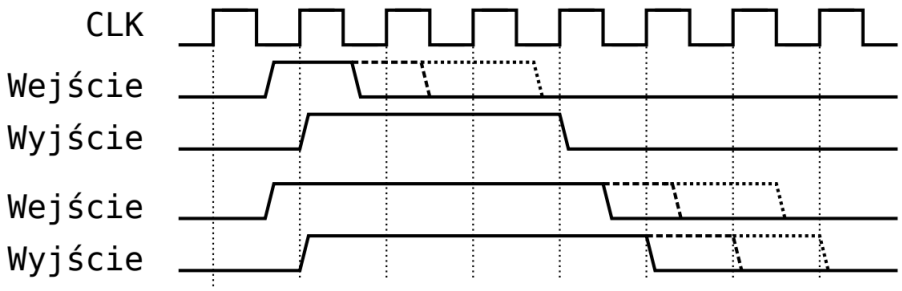
Wszystkie układy zostały zrealizowane za pomocą symulatora PLD dostępnego na serwerze Taurus.

2.1 Maszyna przedłużająca impuls

Ćwiczenie rozpoczęliśmy od zaprojektowania maszyny stanów Moore'a przedłużającej impuls wejściowy. W zależności od długości trwania poziomu wysokiego na wejściu, maszyna:

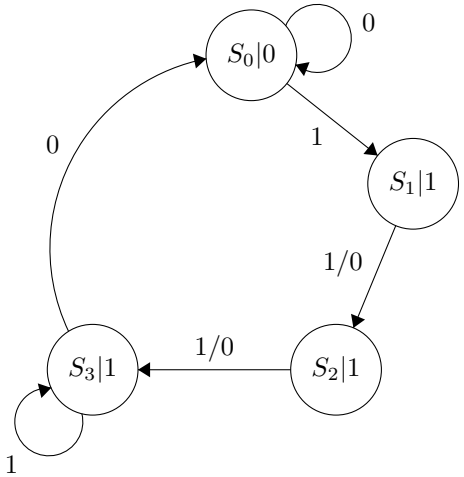
- dla impulsów wejściowych o długości do trzech cykli zegara generuje impuls wyjściowy o długości dokładnie trzech cykli.
- dla impulsów o długości powyżej trzech cykli zegara generuje impuls wyjściowy o takiej długości, jak impuls wejściowy.

Przykładowe przebiegi na wejściu i wyjściu maszyny przedstawiono na rysunku poniżej:



Rysunek 1: Przykładowe przebiegi na wejściu i wyjściu

Projektowanie zacznijmy od poniższego diagramu stanów (stany zapiszemy na bitach O_2 i O_1, (wejście maszyny będzie odpowiadało bitowi A, a wyście bitowi O_3):



Następnie sporządziliśmy tabelę stanów:

Tabela 1: Tabela stanów dla maszyny przedłużającej impuls

Stan	**A**	$\mathbf{O_2}$	$\mathbf{O_1}$	$\mathbf{O_2}'$	$\mathbf{O_1}'$
S_0	0	0	0	0	0
	1	0	0	0	1
S_1	0	0	1	1	0
	1	0	1	1	0
S_2	0	1	0	1	1
	1	1	0	1	1
S_3	0	1	1	0	0
	1	1	1	1	1

Następnie dokonamy minimalizacji funkcji logicznych dla O_2, O_1:

A \ O_2O_1	00	01	11	10
0	0	1	0	1
1	0	1	1	1

$\mathbf{O_2}$

A \ O_2O_1	00	01	11	10
0	0	0	0	1
1	1	0	1	1

$\mathbf{O_1}$

W związku z tym, że maszyna musi zostać zaprojektowana jako maszyna stanów Moore'a uzależnimy wyjście O_3 tylko od O_2, O_1:

Tabela 2: Tabela stanów dla wyjścia O_3

Stan	$\mathbf{O_2}$	$\mathbf{O_1}$	$\mathbf{O_3}'$
S_0	0	0	0
S_1	0	1	1
S_2	1	0	1
S_3	1	1	1

Następnie podobnie dokonaliśmy minimalizacji:

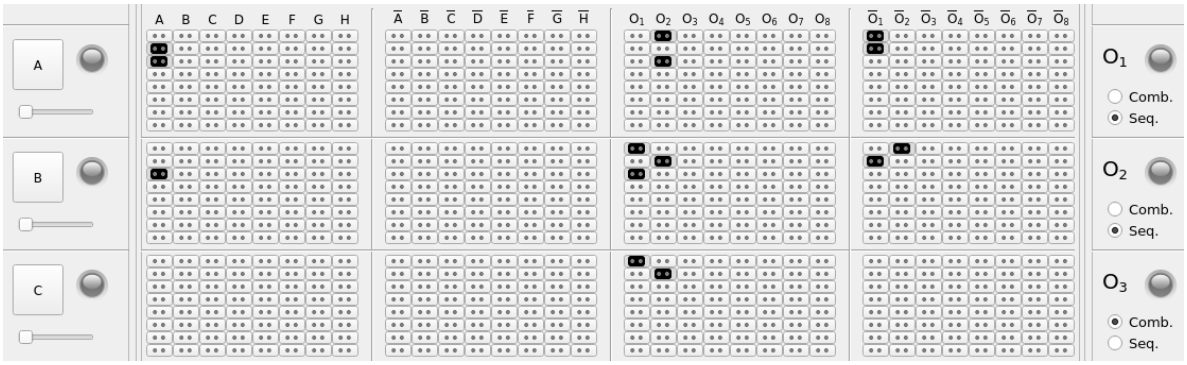
O_2 \ O_1	0	1
0	0	1
1	1	1

$\mathbf{O_3}$

Za pomocą powyższych map jesteśmy w stanie zapisać optymalne funkcje logiczne dla każdego z wyjść:

Wyjście	**Funkcja**
$\mathbf{O_3'}$	$O_2 + O_1$
$\mathbf{O_2'}$	$\overline{O_2}O_1 + O_2\overline{O_1} + AO_1$
$\mathbf{O_1'}$	$O_2\overline{O_1} + A\overline{O_1} + AO_2$

Ostatecznie zaimplementowaliśmy maszynę i poniżej umieszczamy wynik:

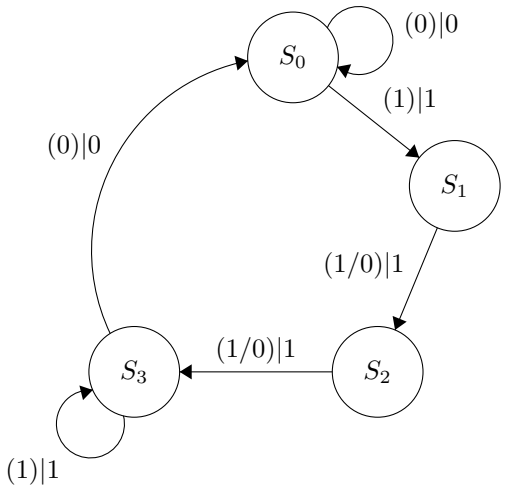


Rysunek 5: Konfiguracja matrycy logicznej dla maszyny przedłużającej impuls

2.2 Latencja, maszyna Meally-go

Zadanie zacznijmy od uwagi, że zgodnie z definicją latencji podanej w opisie ćwiczenia oraz z faktem, że wykorzystujemy symulator, a nie układ rzeczywisty - latencja wyniosła 0, jak dla wyjścia O_3, tak i dla wyjścia O_8, co sugeruje, że i dla kolejnych podłączonych przerzutników latencja też wyniesie 0. Ale odróżniając architekturę Moore'a od architektury Meally-go rozumiemy, że w konfiguracji Meally-go maszyna zareaguje szybciej, bo jej wyjścia będą wprost uzależnione od wejść, czyli układ będzie działał asynchronicznie.

W tym punkcie ćwiczenia zaprojektujemy maszynę z punktu 2.1 jako maszynę Meally-go. Projektowanie zacznijmy od poniższego diagramu stanów:



Następnie sporządziliśmy tabelę stanów:

Tabela 3: Tabela stanów dla maszyny przedłużającej impuls

Stan	**A**	$\mathbf{O_2}$	$\mathbf{O_1}$	$\mathbf{O_3}'$	$\mathbf{O_2}'$	$\mathbf{O_1}'$
S_0	0	0	0	0	0	0
	1	0	0	1	0	1
S_1	0	0	1	1	1	0
	1	0	1	1	1	0
S_2	0	1	0	1	1	1
	1	1	0	1	1	1
S_3	0	1	1	0	0	0
	1	1	1	1	1	1

Następnie podobnie dokonaliśmy minimalizacji:

A \ O_2O_1	00	01	11	10
0	0	1	0	1
1	1	1	1	1

O_3

A \ O_2O_1	00	01	11	10
0	0	1	0	1
1	0	1	1	1

O_2

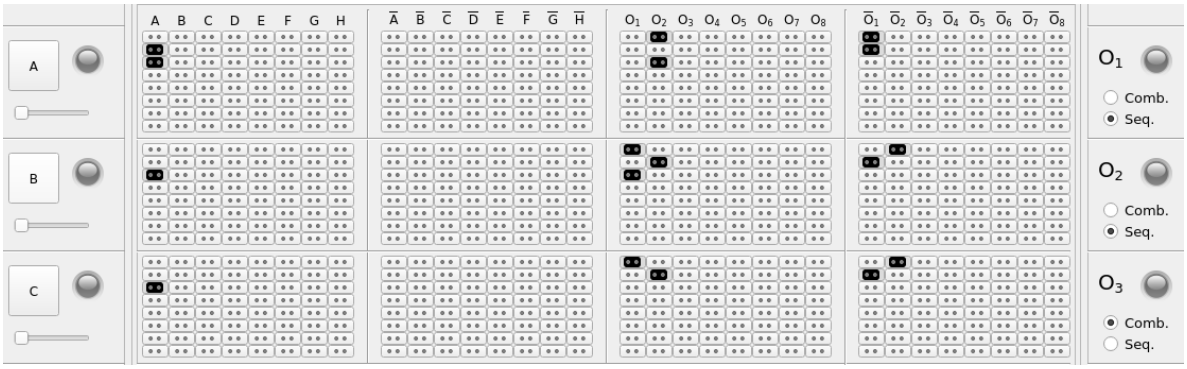
A \ O_2O_1	00	01	11	10
0	0	0	0	1
1	1	0	1	1

O_1

Za pomocą powyższych map jesteśmy w stanie zapisać optymalne funkcje logiczne dla wszystkich wyjść:

Wyjście	**Funkcja**
$\mathbf{O_3'}$	$\overline{O_2}O_1 + O_2\overline{O_1} + A$
$\mathbf{O_2'}$	$\overline{O_2}O_1 + O_2\overline{O_1} + AO_1$
$\mathbf{O_1'}$	$O_2\overline{O_1} + A\overline{O_1} + AO_2$

Ostatecznie zaimplementowaliśmy maszynę i poniżej umieszczamy wynik:



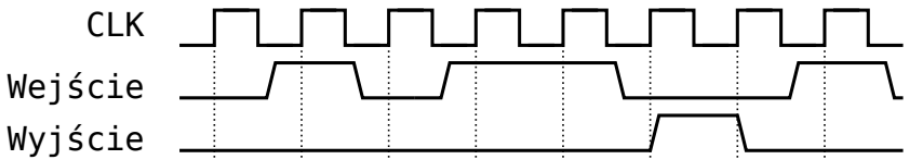
Rysunek 9: Konfiguracja matrycy logicznej dla maszyny przedłużającej impuls

Oczywistym minusem jest fakt, że nasz układ jest podatny na chwilowe szumy w sygnale. Jeżeli pomiędzy wystąpieniem sygnału, a jego zniknięciem nie wystąpi cykl zegara, to wyjście O_3 będzie w stanie wysokim tylko przez czas, póki przychodzi sygnał i nie zostanie wydłużony.

2.3 Maszyna wyszukująca wzorzec (011) o długości trzech bitów

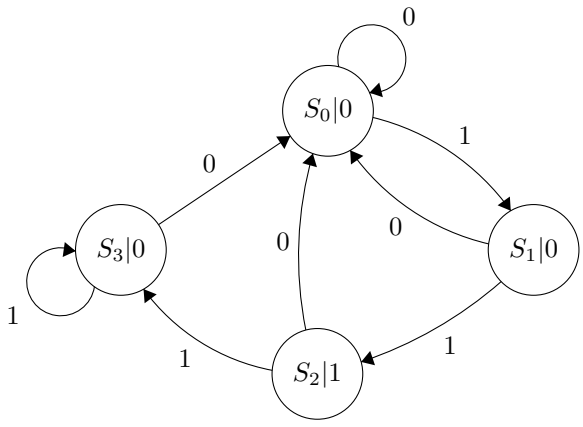
Teraz zaprojektujemy maszynę stanów w architekturze Moore'a, wyszukującą zadany wzorzec długości trzech bitów w ciągu bitów podawanych na jej wejście. W przypadku wykrycia wzorca maszyna ustawia swoje wyjście w stan wysoki na jeden cykl zegara.

Przykładowe przebiegi na wejściu i wyjściu maszyny przedstawiono na rysunku poniżej:



Rysunek 10: Przykładowe przebiegi na wejściu i wyjściu

Projektowanie zacznijmy od poniższego diagramu stanów (stany zapiszemy na bitach O_4 i O_3, wejście maszyny będzie odpowiadało bitowi A, a wyjście bitowi O_3):



Następnie sporządzimy tabelę stanów:

Tabela 4: Tabela stanów dla maszyny wyszukującej wzorzec o długości trzech bitów

Stan	B	O_4	O_3	O_4'	O_3'
S_0	0	0	0	0	0
	1	0	0	0	1
S_1	0	0	1	0	0
	1	0	1	1	0
S_2	0	1	0	0	0
	1	1	0	1	1
S_3	0	1	1	0	0
	1	1	1	1	1

Następnie dokonamy minimalizacji funkcji logicznych dla obu bitów O_4, O_3:

B \ O_4O_3	00	01	11	10
0	0	0	0	0
1	0	1	1	1

O_4

B \ O_4O_3	00	01	11	10
0	0	0	0	0
1	1	0	1	1

O_3

W związku z tym, że maszyna musi zostać zaprojektowana jako maszyna stanów Moore'a uzależniamy wyjście O_5 tylko od bitów O_4, O_3:

Tabela 5: Tabela stanów dla wyjścia O_5

Stan	O_4	O_3	O_5'
S_0	0	0	0
S_1	0	1	0
S_2	1	0	1
S_3	1	1	0

Następnie podobnie dokonaliśmy minimalizacji:

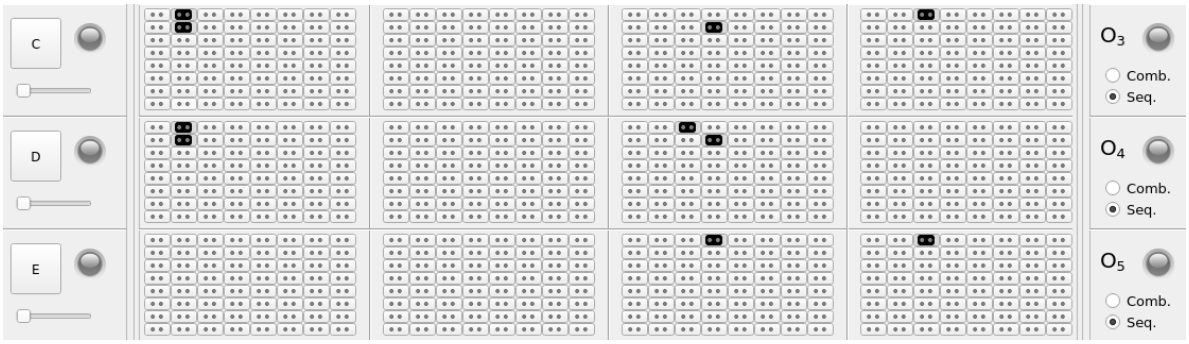
O_4 \ O_3	0	1
0	0	0
1	1	0

O_5

Za pomocą powyższych map jesteśmy w stanie zapisać optymalne funkcje logiczne dla wszystkich wyjść:

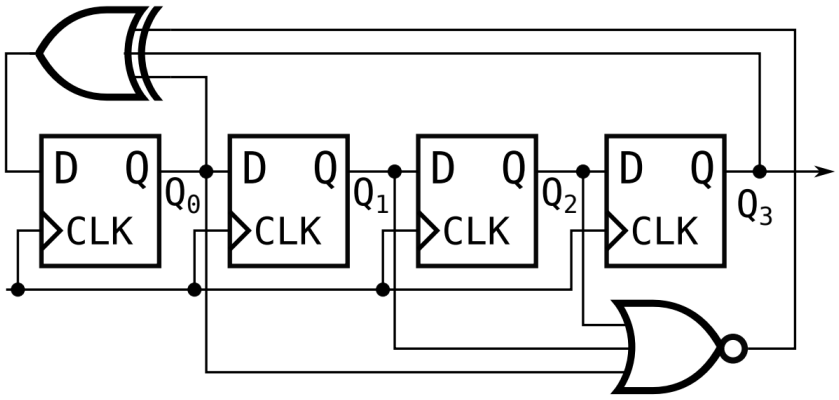
Wyjście	Funkcja
$\mathbf{O'_5}$	$O_4\overline{O_3}$
$\mathbf{O'_4}$	$BO_3 + BO_4$
$\mathbf{O'_3}$	$B\overline{O_3} + BO_4$

Ostatecznie zaimplementowaliśmy maszynę i poniżej umieszczamy wynik:



Rysunek 14: Konfiguracja matrycy logicznej dla maszyny wyszukującej wzorzec o długości trzech bitów

2.4 Generator pseudolosowy



Rysunek 15: Schemat 4-bitowego generatora pseudolosowego LFSR bez stanu zabronionego

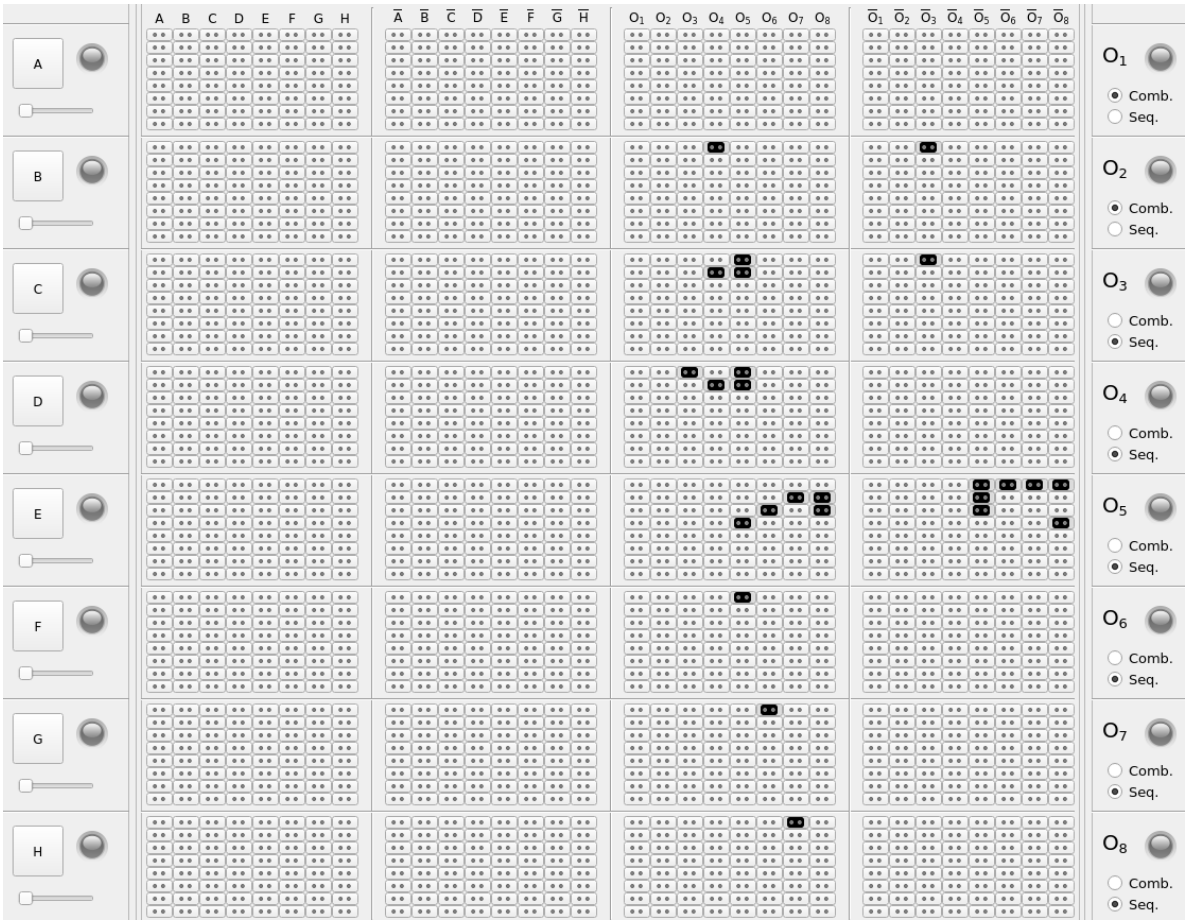
Jako ostatnie zadanie zrealizujemy na bitach O_5 - O_8 czterobitowy generator pseudolosowy LFSR z powyższego schematu, jako weryfikację projektu maszyny wyszukującej wzorzec z poprzedniego punktu. W tym celu wyznaczyliśmy funkcję logiczną trójwejściowej bramki XOR według następującego schematu:

$$A \oplus B \oplus C = \overline{A}\left(\overline{B}C + B\overline{C}\right) + A\left(BC + \overline{B}\overline{C}\right) = \overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C} \tag{1}$$

Teraz jesteśmy w stanie zapisać funkcje logiczne dla wszystkich wyjść (O_5 - O_8) (Funkcja O'_5 opisuje wejście przerzutnika O_5, a pozostałe bity realizują rejestr przesuwny):

Wyjście	**Funkcja**
$\mathbf{O'_8}$	O_7
$\mathbf{O'_7}$	O_6
$\mathbf{O'_6}$	O_5
$\mathbf{O'_5}$	$\overline{O_8 O_7 O_6 O_5} + O_8 O_7 \overline{O_5} + O_8 O_6 \overline{O_5} + \overline{O_8} O_5$

Zwrócimy uwagę, że na bitach O_4 - O_3 są przechowywane stany maszyny wyszukującej wzorzec (011) o długości trzech bitów. Zamiast wejścia B podłączyliśmy bit O_5. Dodatkowo w celu ułatwienia weryfikacji wyjście maszyny rozpoznającej wzorzec przenieśliśmy na bit O_2. Ostatecznie schemat zaimplementowanej maszyny umieszczamy poniżej:



Rysunek 16: Realizacja generatora pseudolosowego LFSR na programowalnej matrycy logicznej

Obserwując stany maszyny potwierdzamy, że przechodzi ona w stan znalezienia wzorca dokładnie 2 razy w czasie trwania sekwencji pseudolosowej z LFSR.