

Projekt 3: Animacja Tekstu

Jan Czechowski, Kinga Konieczna, Paweł Plewa, Tymon Zadara

6 czerwca 2025

Spis treści

1	Cel projektu	2
2	Opis zadania	2
2.1	aut	2
2.2	NS	3
2.3	d_H	4
2.3.1	Wyświetlanie	4
2.3.2	Konfiguracja danych wejściowych dla wyświetlaczy	4
3	Realizacja	5
3.1	aut	5
3.2	NS	6
3.3	d_H	6
3.3.1	Wyświetlacz pierwszy - d_H3	7
3.3.2	Wyświetlacz drugi - d_H2	9
3.3.3	Wyświetlacz trzeci - d_H1	11
3.3.4	Wyświetlacz czwarty - d_H0	13
4	Testowanie	15
5	Wnioski	22

1 Cel projektu

Celem projektu jest skonstruowanie systemu animującego tekst przy użyciu wyświetlaczy 7 - segmentowych. W celu stworzenia działającego układu animacji tekstu należało zmodyfikować poniższe podukłady w podanym programie *Logisim*:

- aut - układ wprowadzający dane do podukładów
- NS - next state - układ zmieniający wartość stanu
- dH0
- dH1
- dH2
- dH3

W ramach realizacji zadania zaimplementowaliśmy animację tekstu czteroliterowego **PEAS**.

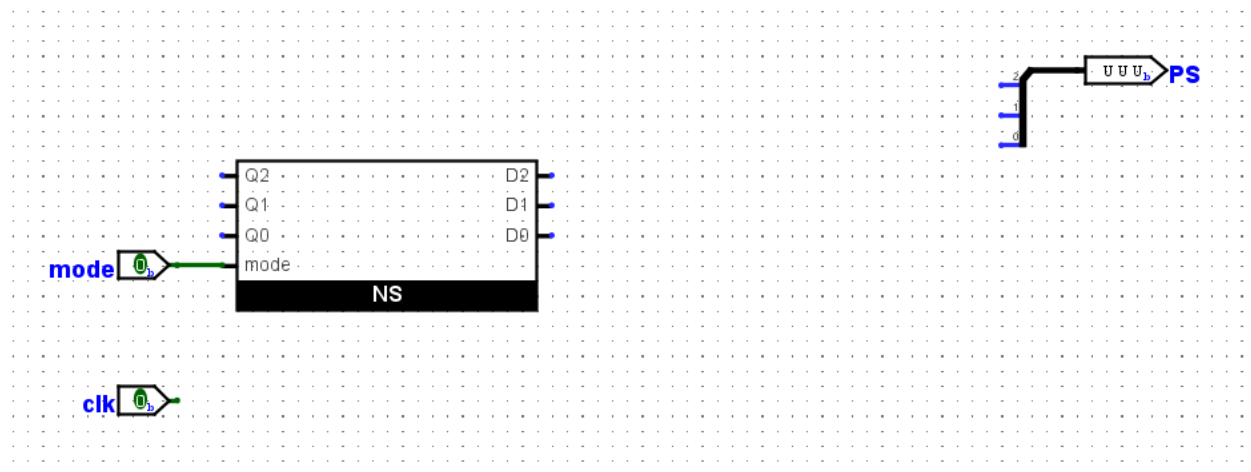
2 Opis zadania

Aby zrealizować zadanie, należało zmodyfikować podukłady podane w sekcji *Cel projektu*, aby skonstruować poprawnie działający animator tekstu.

Każdy z podukładów pełni swoją konkretną funkcję i ma swoje wymagania i założenia które musi spełnić. Te informacje zawarte są w kolejnych podpunktach dotyczących podukładów.

2.1 aut

Pierwszym podukładem jest moduł *aut* przedstawiony na **Rysunku 1**. Jest to moduł zajmujący się wprowadzaniem danych obecnego stanu wyświetlacza do modułu *NS* (w celu jego zwiększenia) oraz wprowadzaniem obecnych danych (już tych zwiększonych) do *maina* gdzie zależnie od wartości wyjściowej tego modułu, wyświetli się odpowiednia litera (lub jej brak) na wyświetlaczach.



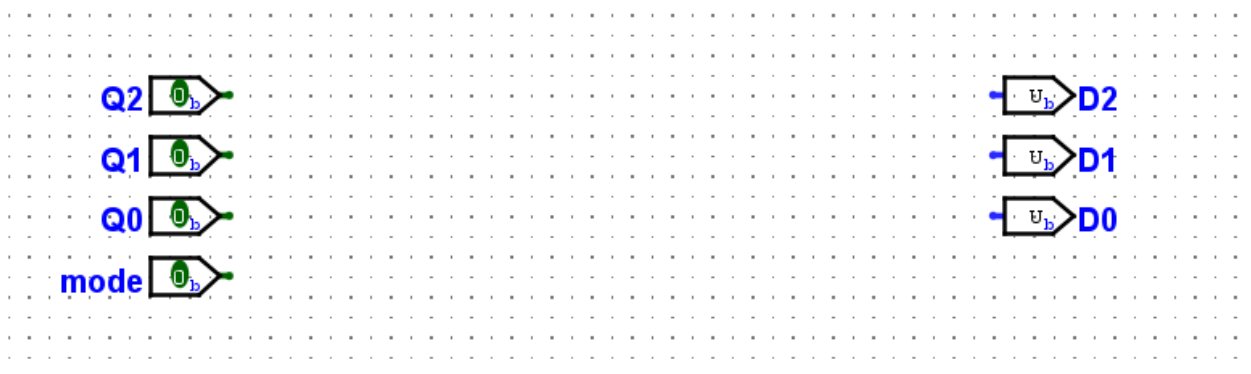
Rysunek 1: Moduł aut przed modyfikacjami

Działanie tego modułu polega na tym, że przy każdej iteracji zegara (włącz i wyłącz) powinna nastąpić zmiana stanu z poprzednio zapisanego na następny, wprowadzenie tego stanu do *maina* oraz zapis zmienionego stanu do pamięci *NS* w celu wykorzystania tego do następnej iteracji.

W ten sposób z każdym krokiem powinniśmy dostawać kolejne stany.

2.2 NS

Drugim podukładem jest moduł *NS* widoczny na **Rysunku 2**. Jest to moduł zajmujący się inkrementacją numeru stanu między wartościami od 0 do 7 (8 wartości dla 8 stanów). Dodatkową funkcją, którą należy zaimplementować w tym module jest przełącznik *mode*, który służy jako włącznik animatora.



Rysunek 2: Moduł NS przed modyfikacjami

2.3 d_H

Ostatnimi czterema podukładami są d_H3 , d_H2 , d_H1 , d_H0 . Są to moduły odpowiedzialne za wyświetlanie danych na wyświetlaczach w zależności od wartości na wejściu modułu.

2.3.1 Wyświetlanie

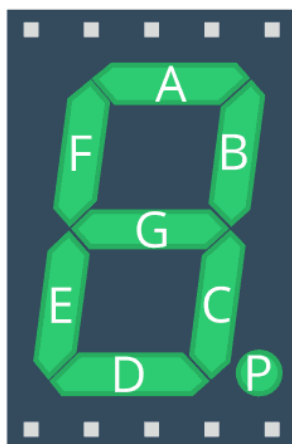
Aby stworzyć te podukłady należy najpierw przeanalizować dane jakie muszą być wyświetlane dla każdego stanu dla wszystkich wyświetlaczy (patrz tabela poniżej 1).

Tabela 1: Wyświetlane dane w zależności od stanów wejściowych dla słowa *PEAS*. ("*" oznacza pusty wyświetlacz.)

Licznik Stanu	d_H3	d_H2	d_H1	d_H0
000	*	*	*	P
001	*	*	P	E
010	*	P	E	A
011	P	E	A	S
100	E	A	S	*
101	A	S	*	*
110	S	*	*	*
111	*	*	*	*

2.3.2 Konfiguracja danych wejściowych dla wyświetlaczy

Kolejnym krokiem w celu skonstruowania działającego animatora tekstu jest zdefiniowanie wartości wejściowych do wyświetlczy dla poszczególnych literek naszego wyrazu **PEAS**. Poniżej zaprezentowany jest schemat wyświetlania danych na wyświetlacz 7-segmentowym [Rysunek 3.]:



Rysunek 3: Schemat wyświetlacza 7-segmentowego [źródło: https://cdn.forbot.pl/blog/wp-content/uploads/2016/11/kursTC_7_1-wyswietlacz_7_SEG.png]

Korzystając ze schematu wyświetlacza 7-segmentowego (3) skonstruowaliśmy tabelę wartości wejściowych (2), jakie trzeba wprowadzić do wyświetlacza, aby uzyskać wymagane przez nas literki słowa *PEAS*.

Tabela 2: Dane wejściowe do wyświetlacza w zależności od wyświetlanej literki)

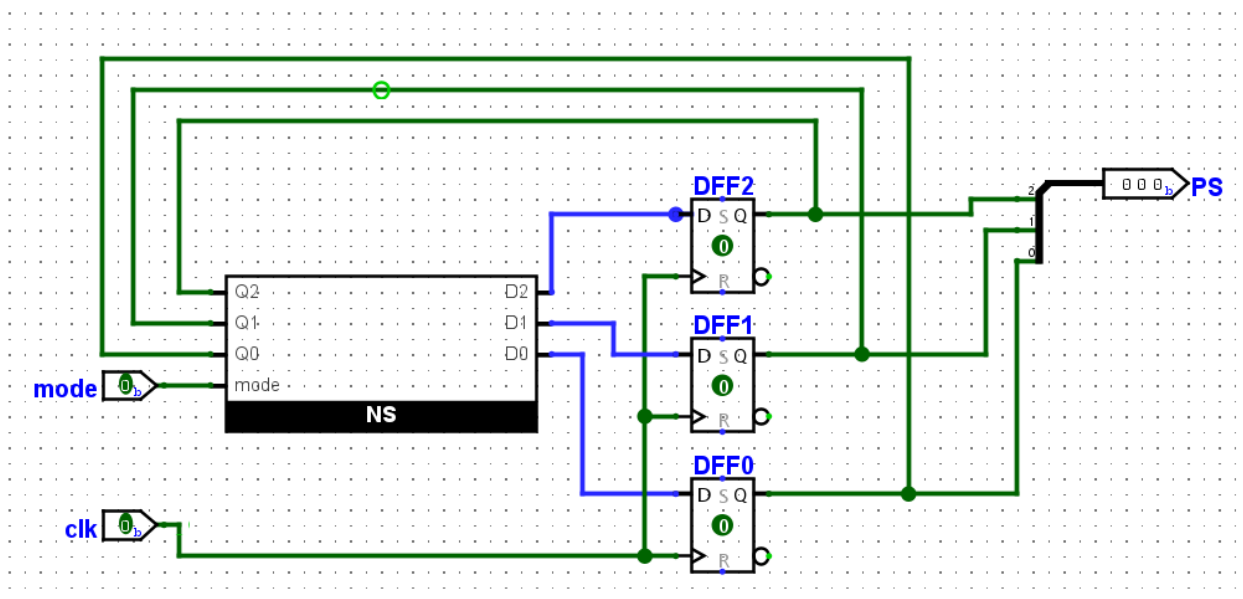
Litera	Segmenty do zapalenia
P	A,B,E,F,G
E	A,D,E,F,G
A	A,B,C,E,F,G
S	A,C,D,F,G

3 Realizacja

W tej sekcji opisane są wszystkie modyfikacje wprowadzone do każdego układu oraz ich działanie.

3.1 aut

W celu uzupełnienia modułu *aut* zastosowano przerzutniki typu D (*D - Flip Flop*) co jest widoczne na **Rysunku 4**. Wartość przerzutnika zmienia się po wprowadzeniu do niego na wejściu innej wartości niż ta w nim zapisana. Weryfikacja wartości następuje przy każdej iteracji zegara. Zastosowane zostały 3 przerzutniki odpowiednio dla 3 bitów przy użyciu których reprezentowana jest liczba stanów animatora (8). Wartości wyjściowe z przerzutników idą do wyjścia modułu *aut* oraz do wejścia modułu *NS* w celu wykorzystania ich do wywołania następnego stanu.



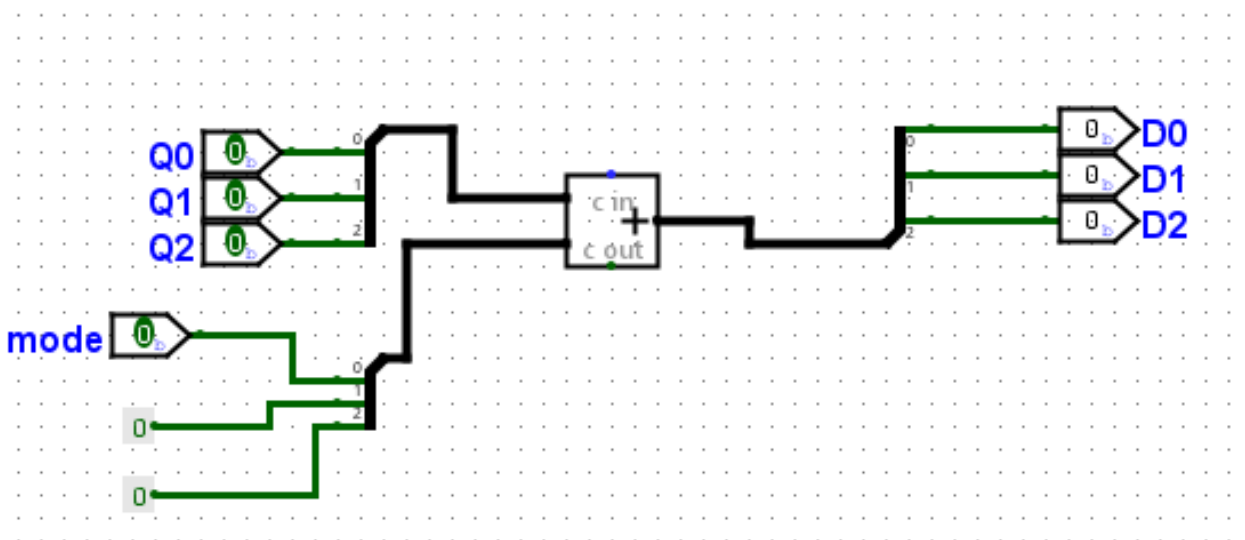
Rysunek 4: Moduł *aut* po modyfikacjach.

3.2 NS

W celu uzupełnienia modułu *NS* zastosowano następujące elementy:

- Splitter IN - połączenie 3 pojedynczych bitów na wejściu w jedną wartość 3-bitową, którą można wtedy podłączyć do addera
- Splitter OUT - rozłączenie 3-bitowej wartości wyjściowej z addera na 3 pojedyncze bity, które można podać na wyjście układu *NS*
- Adder - dodawanie do wartości wejściowej 1 albo 0 (w zależności od włączenia układu). Wykorzystany został adder 3-bitowy w celu ograniczenia wartości na wejściu oraz wyjściu addera do 0, 1, 2, 3, 4, 5, 6, 7.

Moduł *NS* po modyfikacjach widoczny jest na **Rysunku 5**.



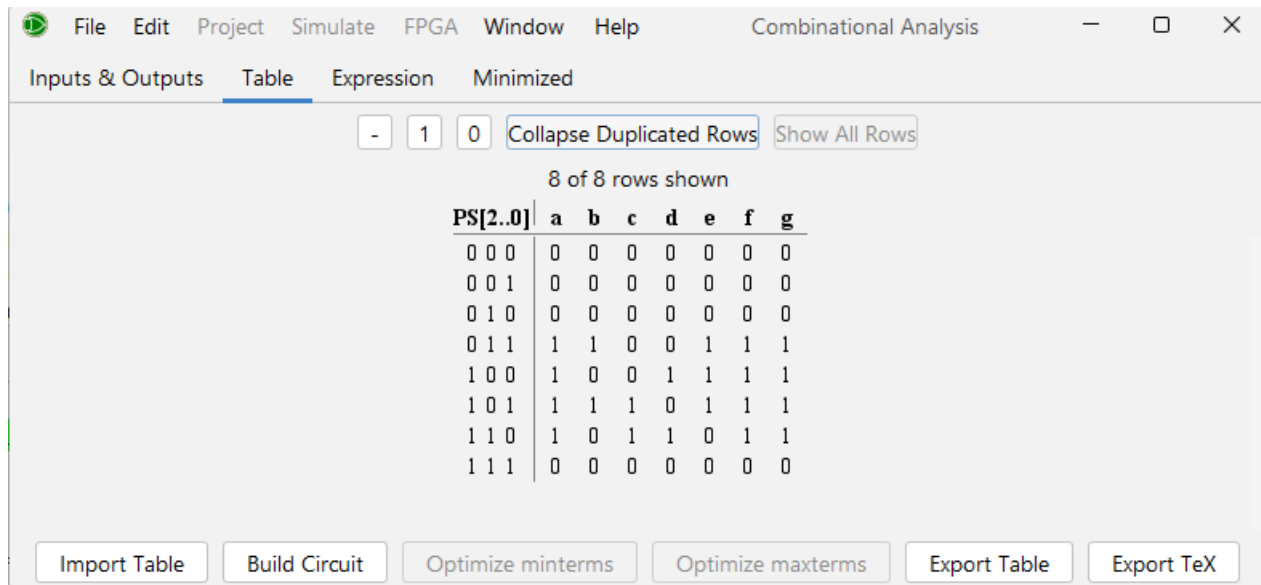
Rysunek 5: Moduł *NS* po modyfikacjach.

3.3 d_H

Proces konfiguracji wyświetlaczy został podzielony na 4 sekcje w zależności od ustawianego wyświetlacza. Konfiguracja każdego z wyświetlaczy polegać będzie na wprowadzeniu danych opisanych w punkcie 2.3 do sekcji *Project* → *Analyze Circuit* → *Table* w Logisim. Funkcja tam utworzy nam gotowy układ bramek w Logisim spełniający założenia naszego układu.

3.3.1 Wyświetlacz pierwszy - d_H3

Tabela prawdy dla wyświetlacza H3 przedstawiona jest poniżej na **Rysunku 6**.

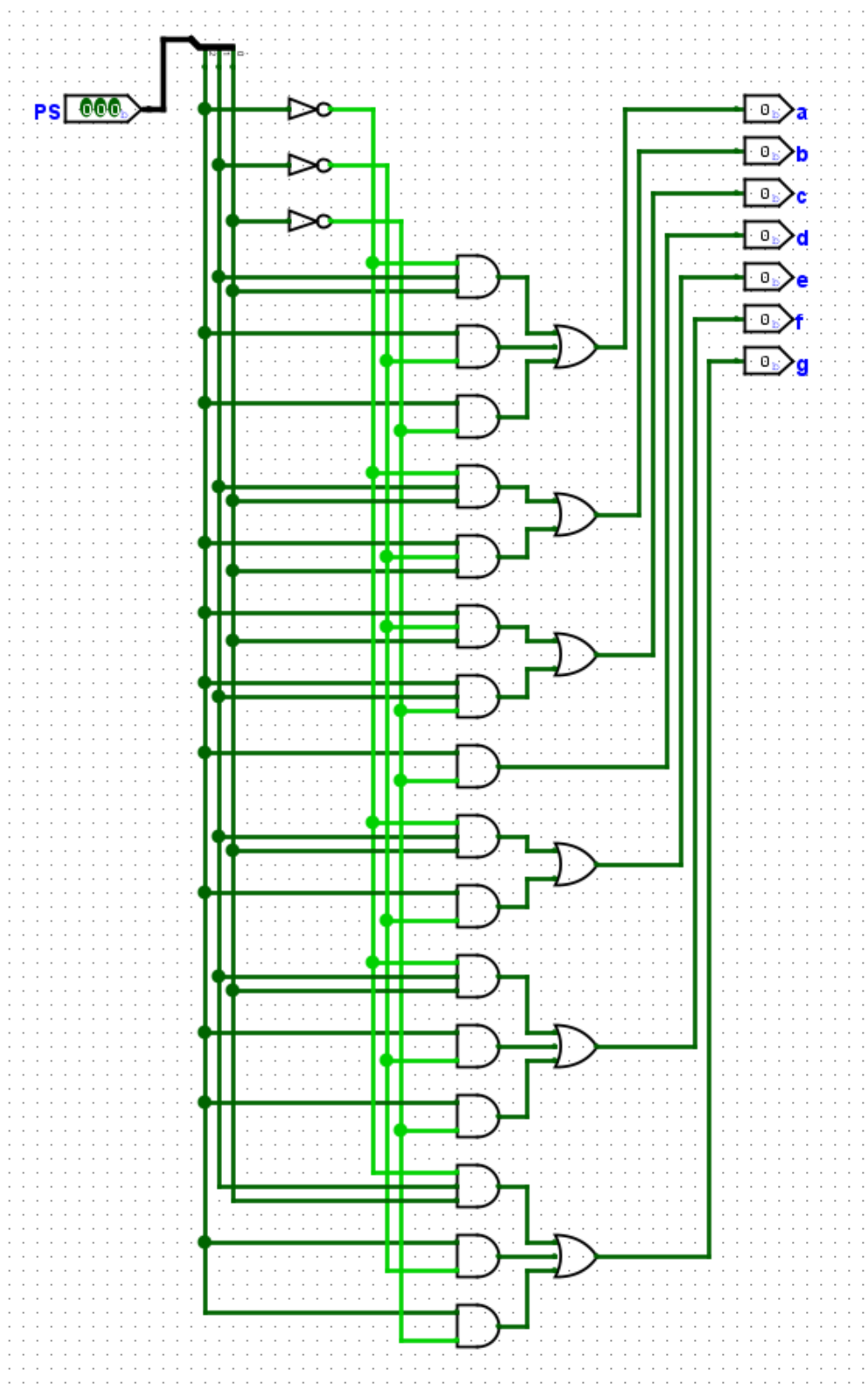


8 of 8 rows shown

PS[2..0]	a	b	c	d	e	f	g
0 0 0	0	0	0	0	0	0	0
0 0 1	0	0	0	0	0	0	0
0 1 0	0	0	0	0	0	0	0
0 1 1	1	1	0	0	1	1	1
1 0 0	1	0	0	1	1	1	1
1 0 1	1	1	1	0	1	1	1
1 1 0	1	0	1	1	0	1	1
1 1 1	0	0	0	0	0	0	0

Rysunek 6: Tabela prawdy dla wyświetlacza H3.

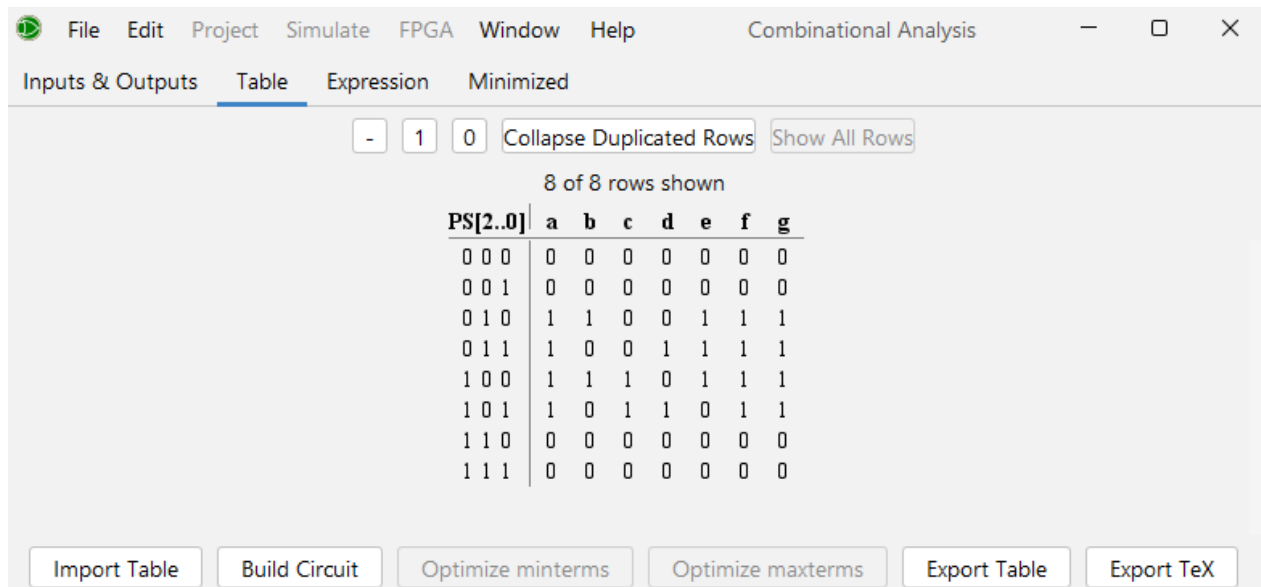
Na **Rysunku 7.** poniżej znajduje się układ bramkowy dla wyświetlacza H3.



Rysunek 7: Układ bramkowy dla wyświetlacza H3.

3.3.2 Wyświetlacz drugi - d_H2

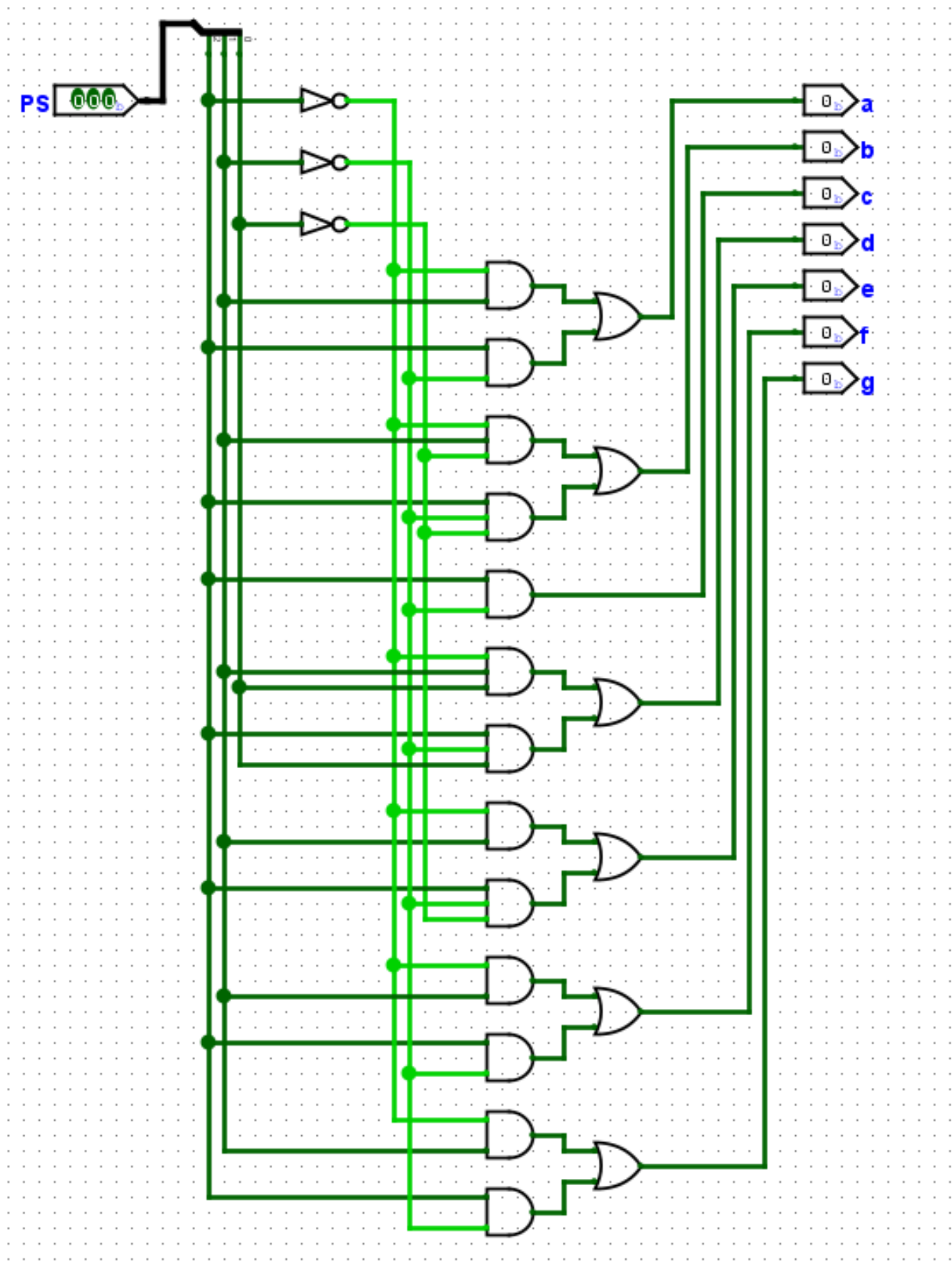
Poniżej na **Rysunku 8.** widoczna jest tabela prawdy dla wyświetlacza H2.



PS[2..0]	a	b	c	d	e	f	g
0 0 0	0	0	0	0	0	0	0
0 0 1	0	0	0	0	0	0	0
0 1 0	1	1	0	0	1	1	1
0 1 1	1	0	0	1	1	1	1
1 0 0	1	1	1	0	1	1	1
1 0 1	1	0	1	1	0	1	1
1 1 0	0	0	0	0	0	0	0
1 1 1	0	0	0	0	0	0	0

Rysunek 8: Tabela prawdy dla wyświetlacza H2.

Poniżej na **Rysunku 9.** widoczny jest układ bramkowy dla wyświetlacza H2.



Rysunek 9: Układ bramkowy dla wyświetlacza H2.

3.3.3 Wyświetlacz trzeci - d_H1

Na **Rysunku 10.** poniżej przedstawiona jest tabela prawdy dla wyświetlacza H1.

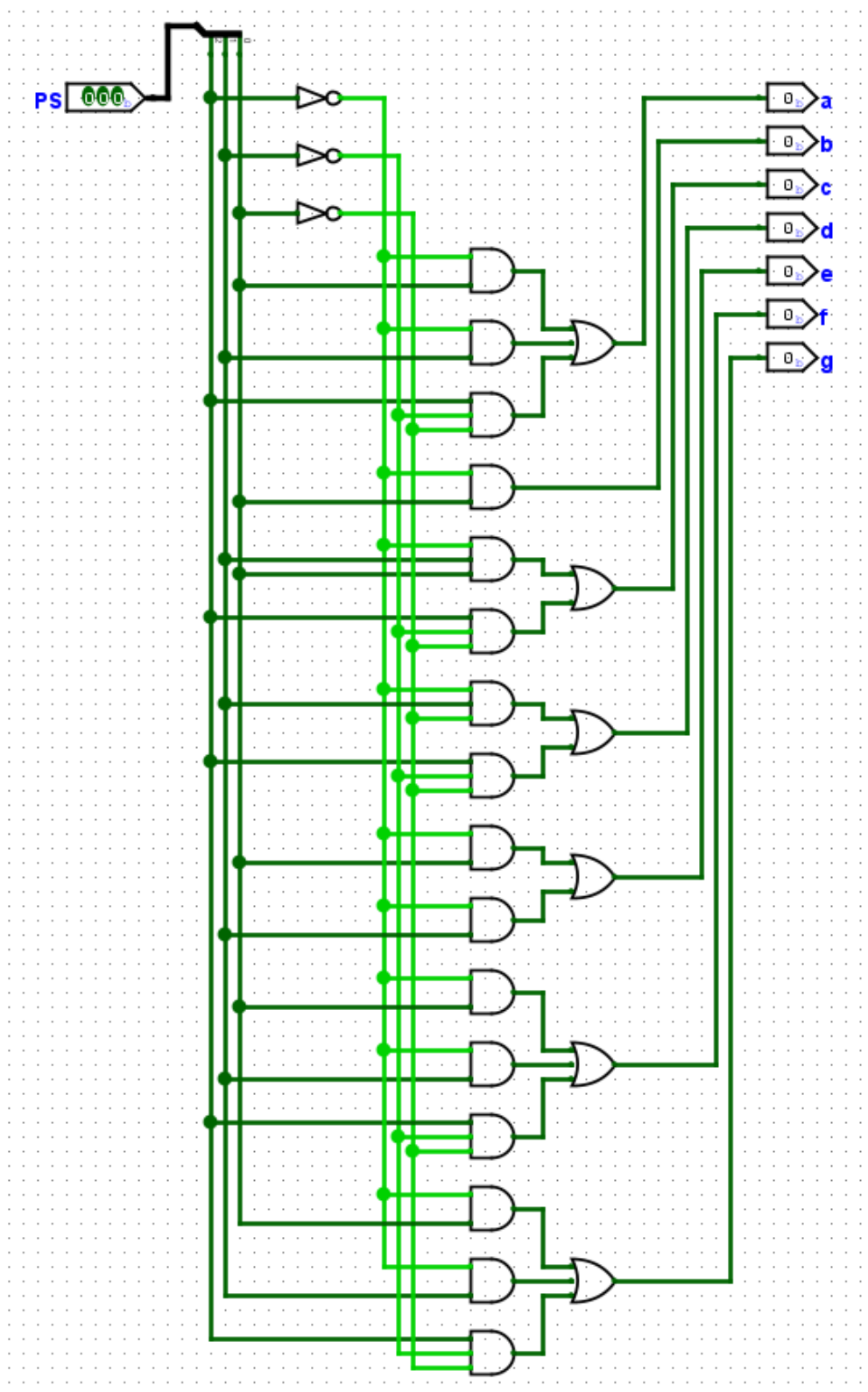
8 of 8 rows shown

PS[2..0]	a	b	c	d	e	f	g
0 0 0	0	0	0	0	0	0	0
0 0 1	1	1	0	0	1	1	1
0 1 0	1	0	0	1	1	1	1
0 1 1	1	1	1	0	1	1	1
1 0 0	1	0	1	1	0	1	1
1 0 1	0	0	0	0	0	0	0
1 1 0	0	0	0	0	0	0	0
1 1 1	0	0	0	0	0	0	0

Buttons: Import Table, Build Circuit, Optimize minterms, Optimize maxterms, Export Table, Export TeX

Rysunek 10: Tabela prawdy dla wyświetlacza H1.

Na **Rysunku 11.** poniżej znajduje się układ bramkowy dla wyświetlacza H1.



Rysunek 11: Układ bramkowy dla wyświetlacza H1.

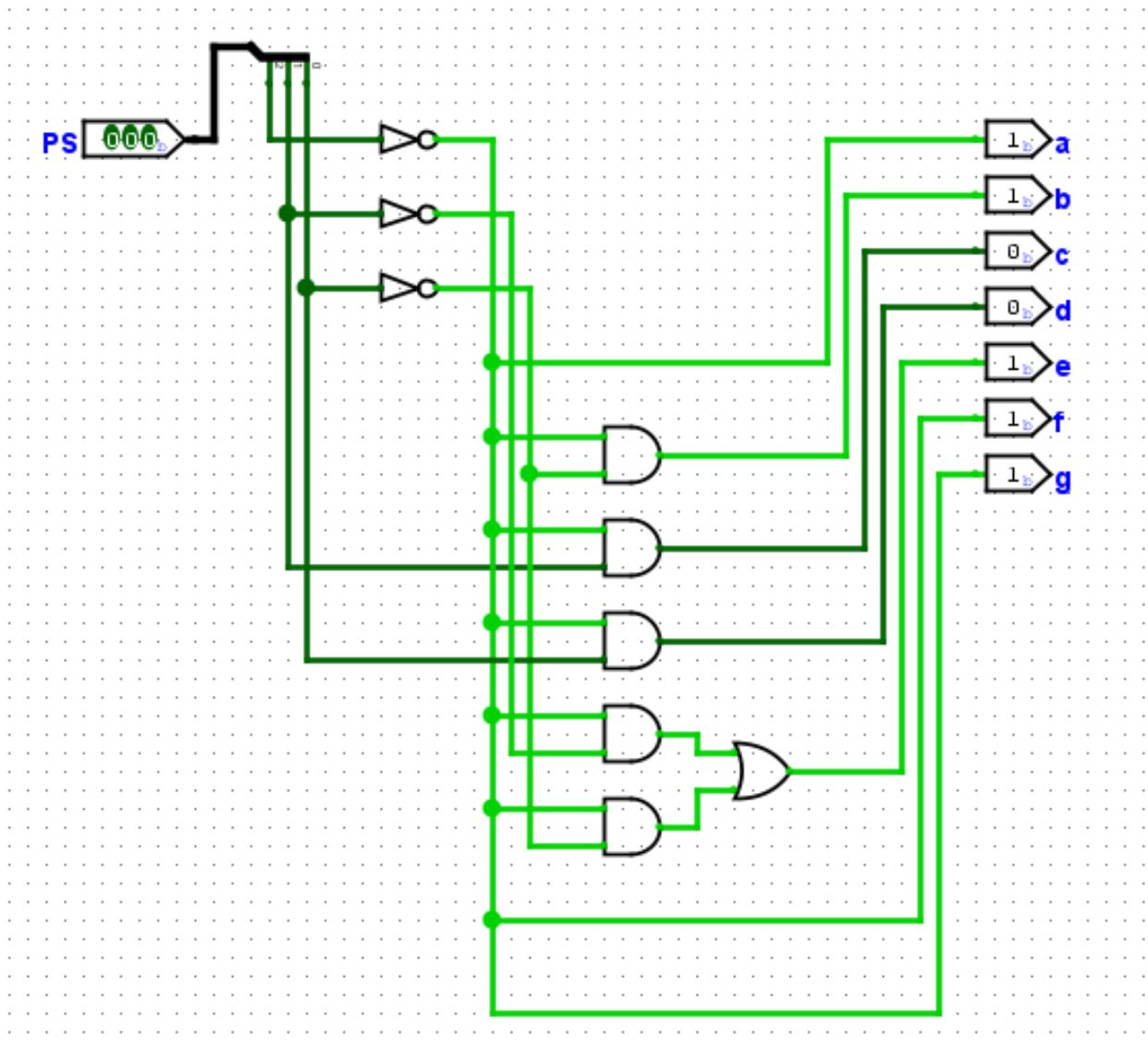
3.3.4 Wyświetlacz czwarty - d_H0

Poniżej na **Rysunku 12.** widoczna jest tabela prawdy dla wyświetlacza H0.

PS[2..0]	a	b	c	d	e	f	g
0 0 0	1	1	0	0	1	1	1
0 0 1	1	0	0	1	1	1	1
0 1 0	1	1	1	0	1	1	1
0 1 1	1	0	1	1	0	1	1
1 0 0	0	0	0	0	0	0	0
1 0 1	0	0	0	0	0	0	0
1 1 0	0	0	0	0	0	0	0
1 1 1	0	0	0	0	0	0	0

Rysunek 12: Tabela prawdy dla wyświetlacza H0.

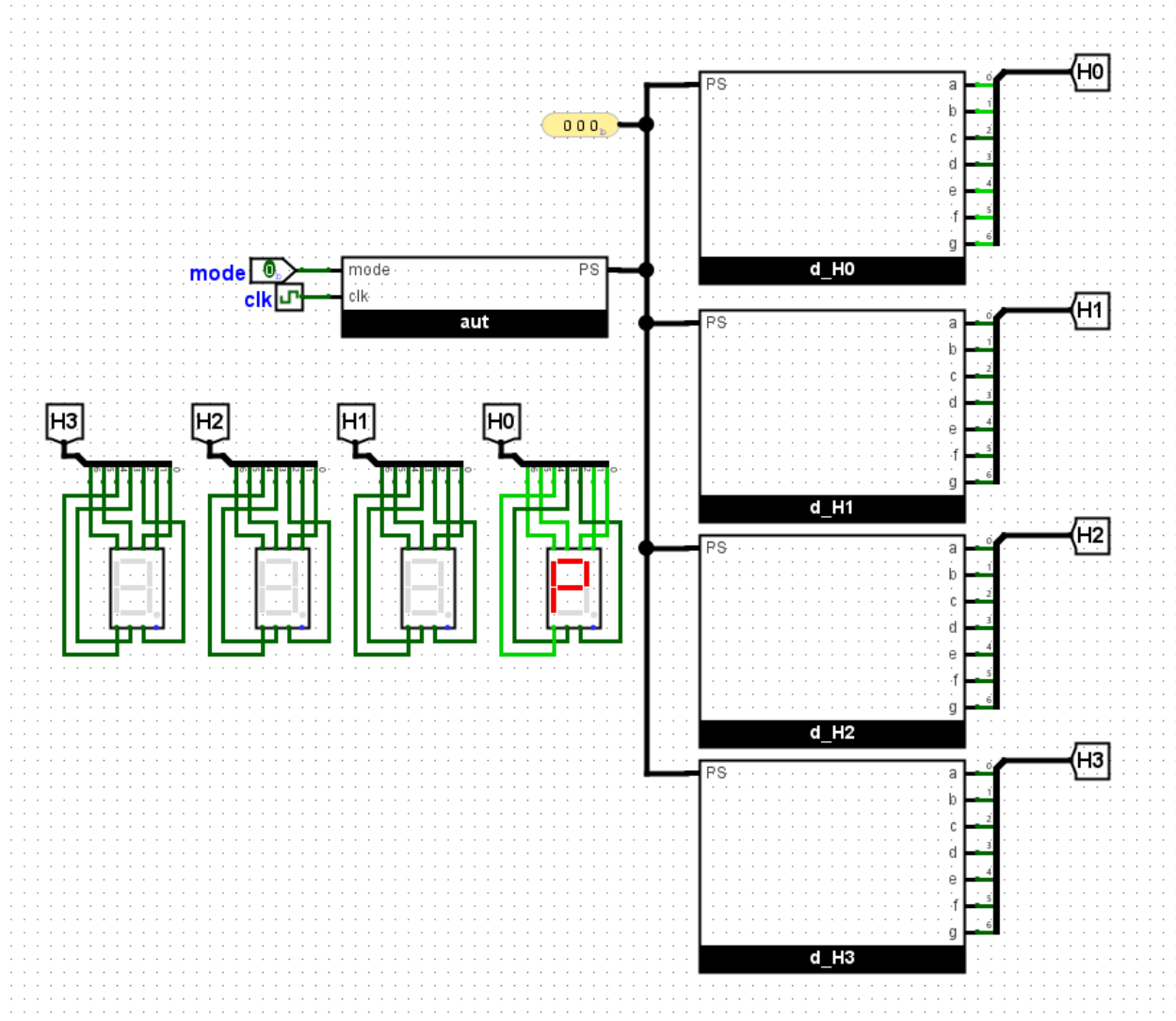
Na **Rysunku 13.** poniżej znajduje się układ bramkowy dla wyświetlacza H0.



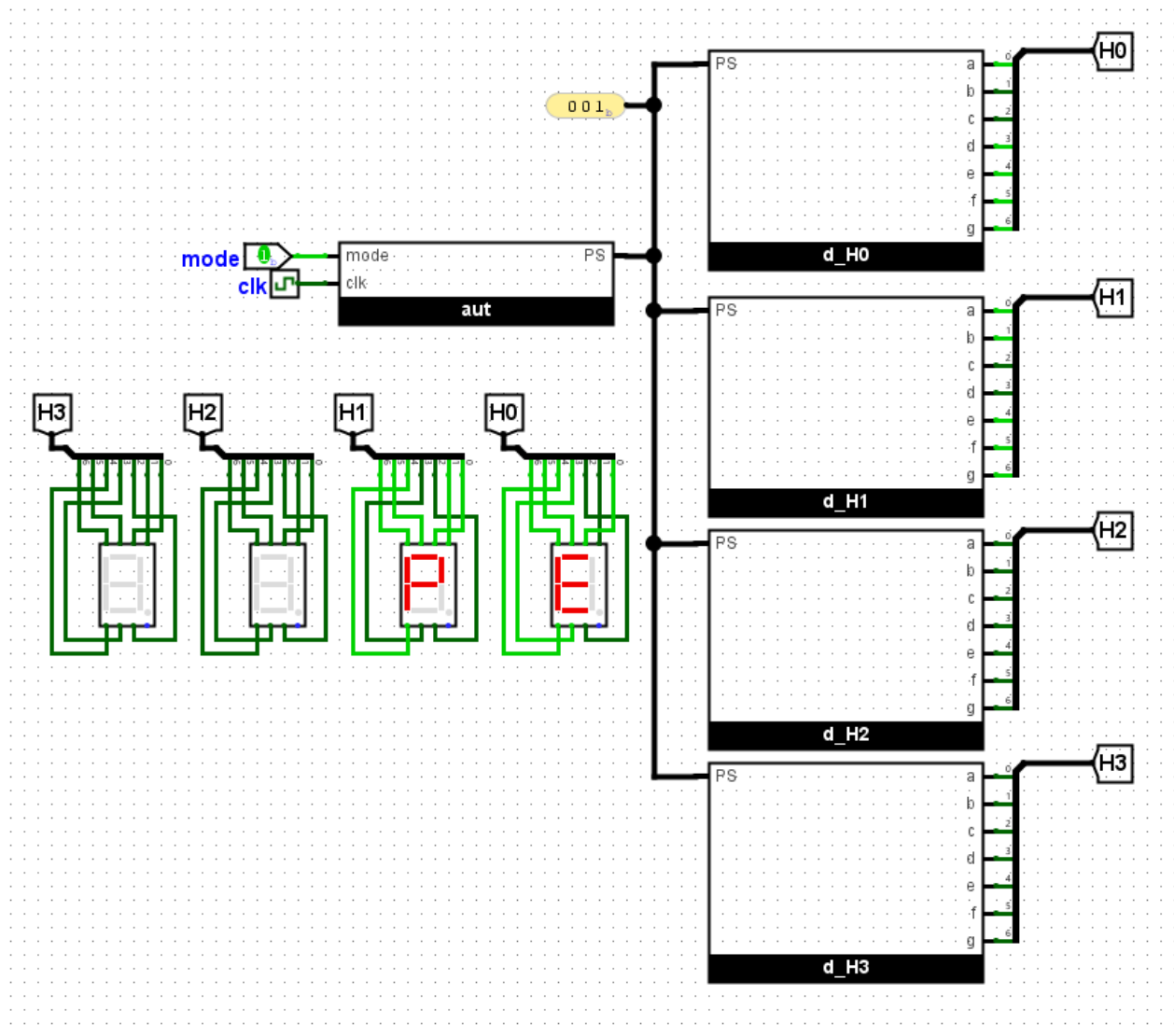
Rysunek 13: Układ bramkowy dla wyświetlacza H0.

4 Testowanie

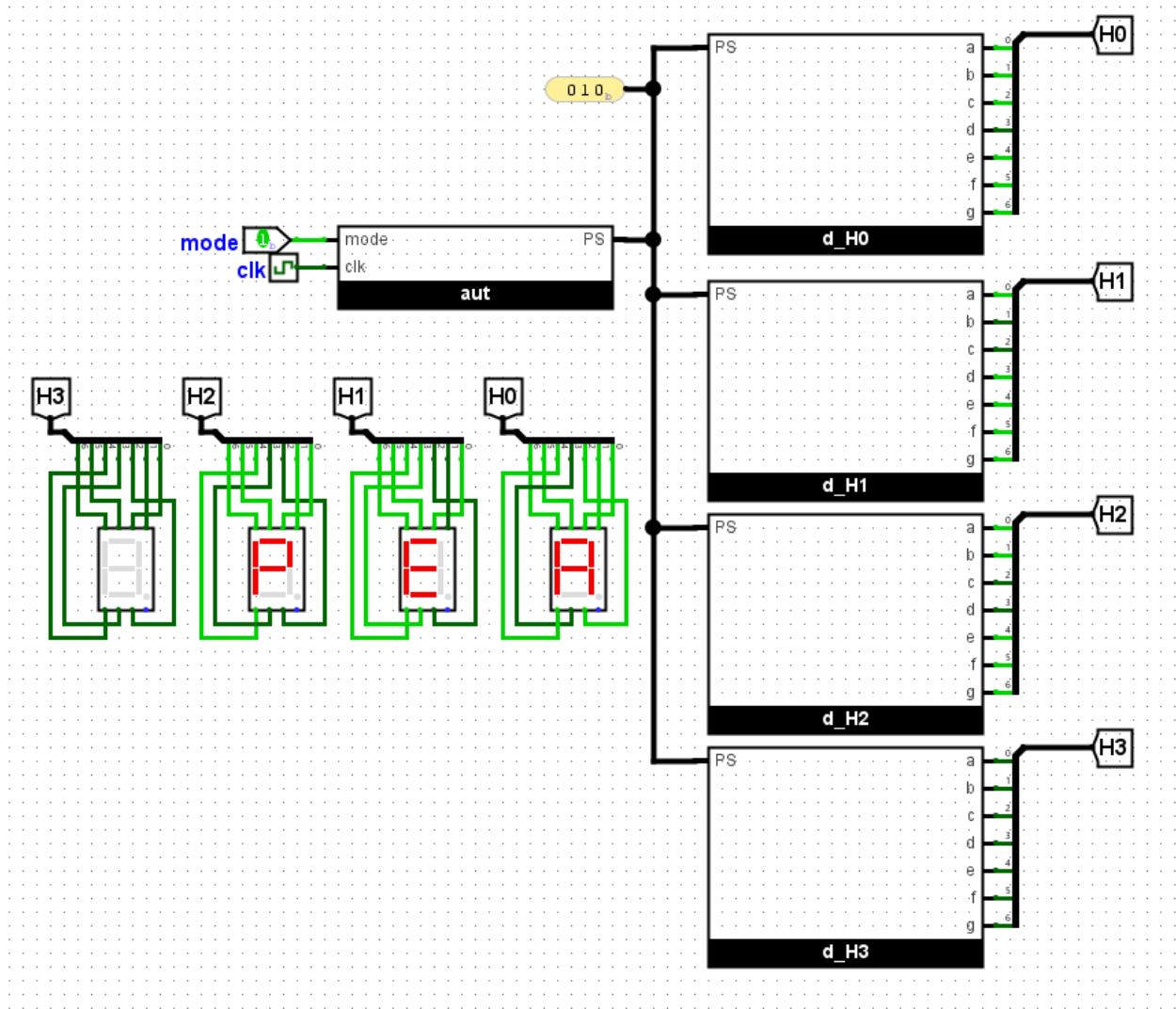
Poniżej na **Rysunku 14.**, **15.**, **16.**, **17.**, **18.**, **19.**, **20.** oraz **21.** przedstawione zostały efekty wywoływania kolejnych stanów na wyświetlaczach w module main.



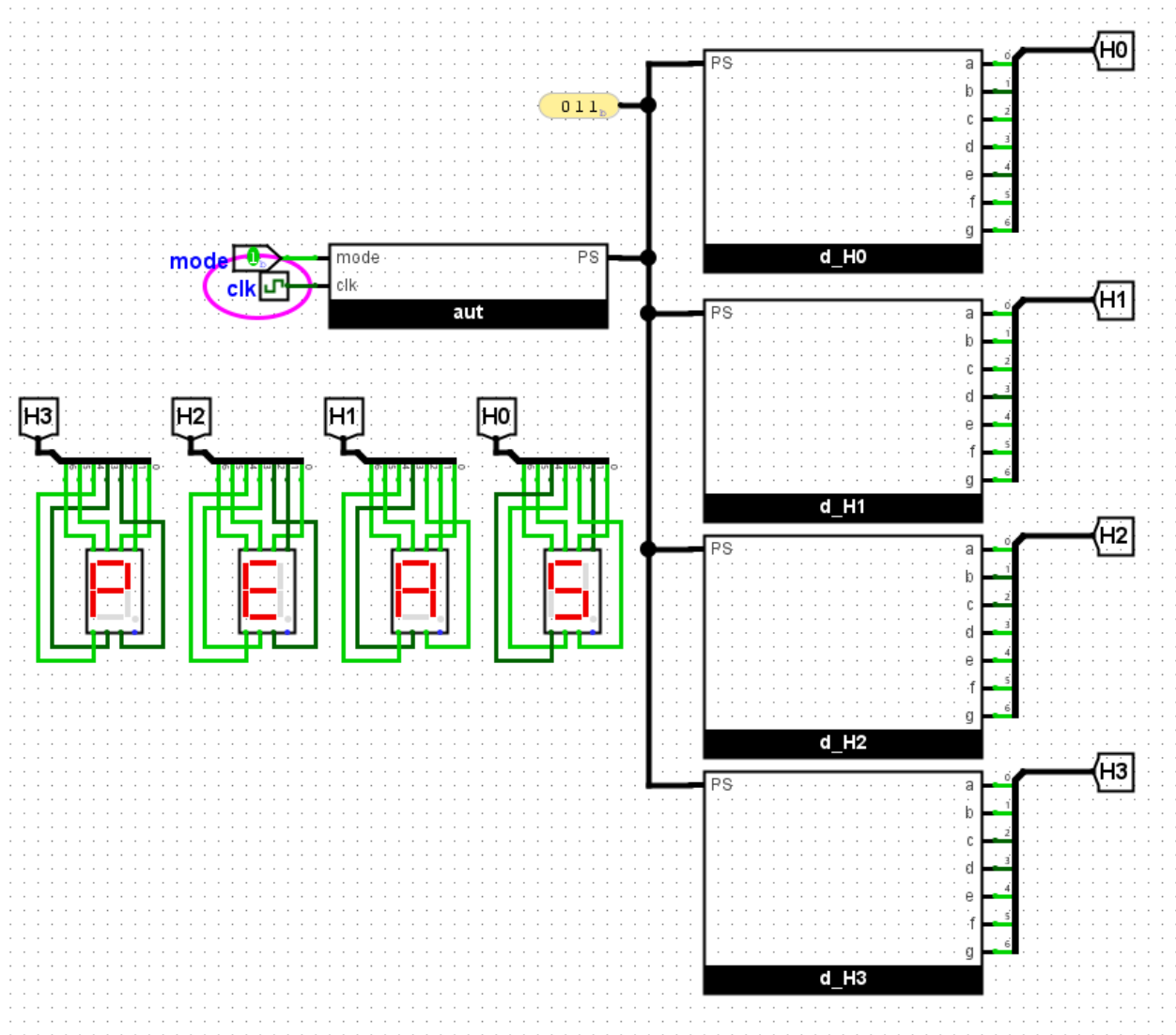
Rysunek 14: Wyświetlacz tekstu dla stanu pierwszego.



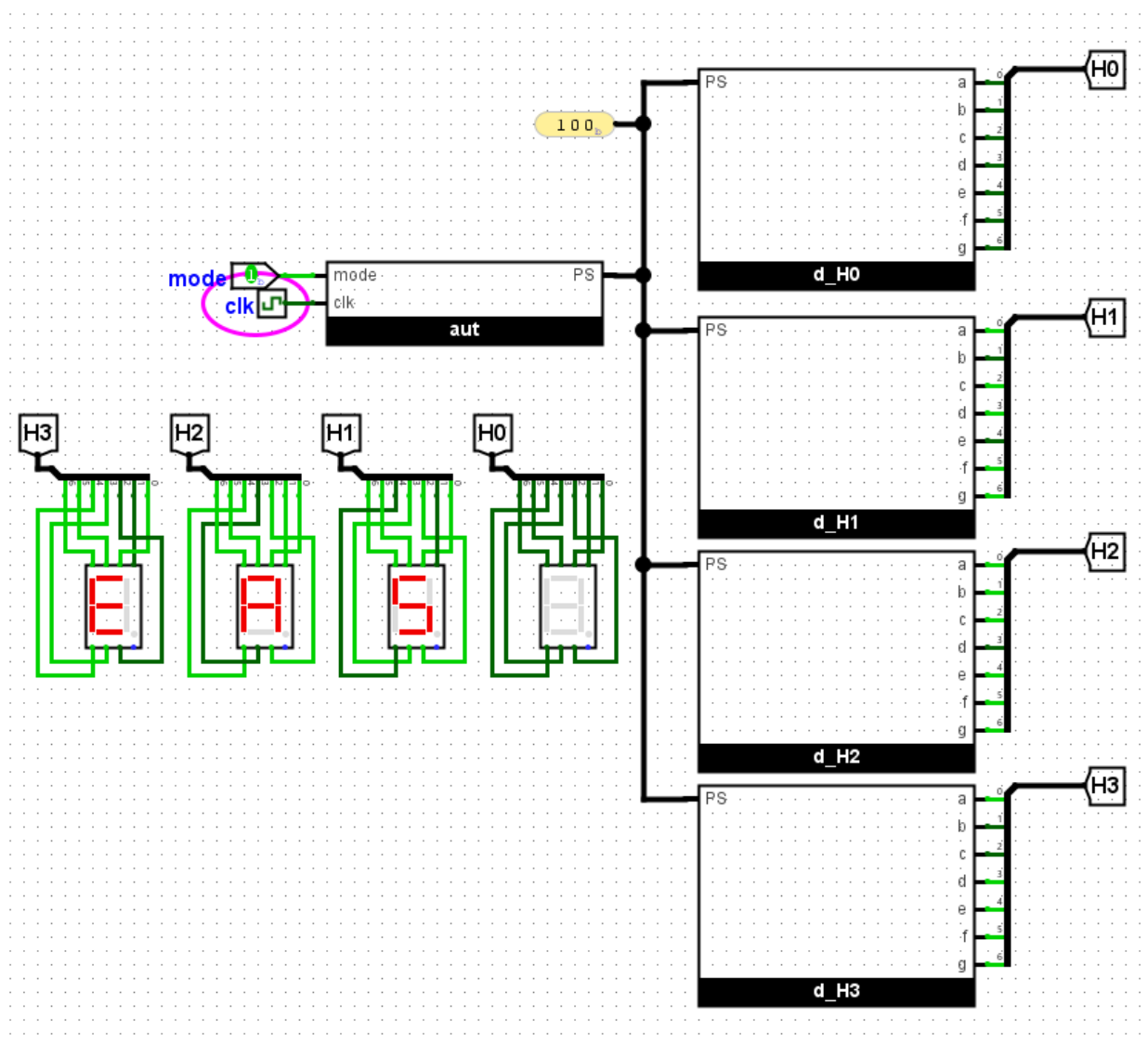
Rysunek 15: Wyświetlacz tekstu dla stanu drugiego.



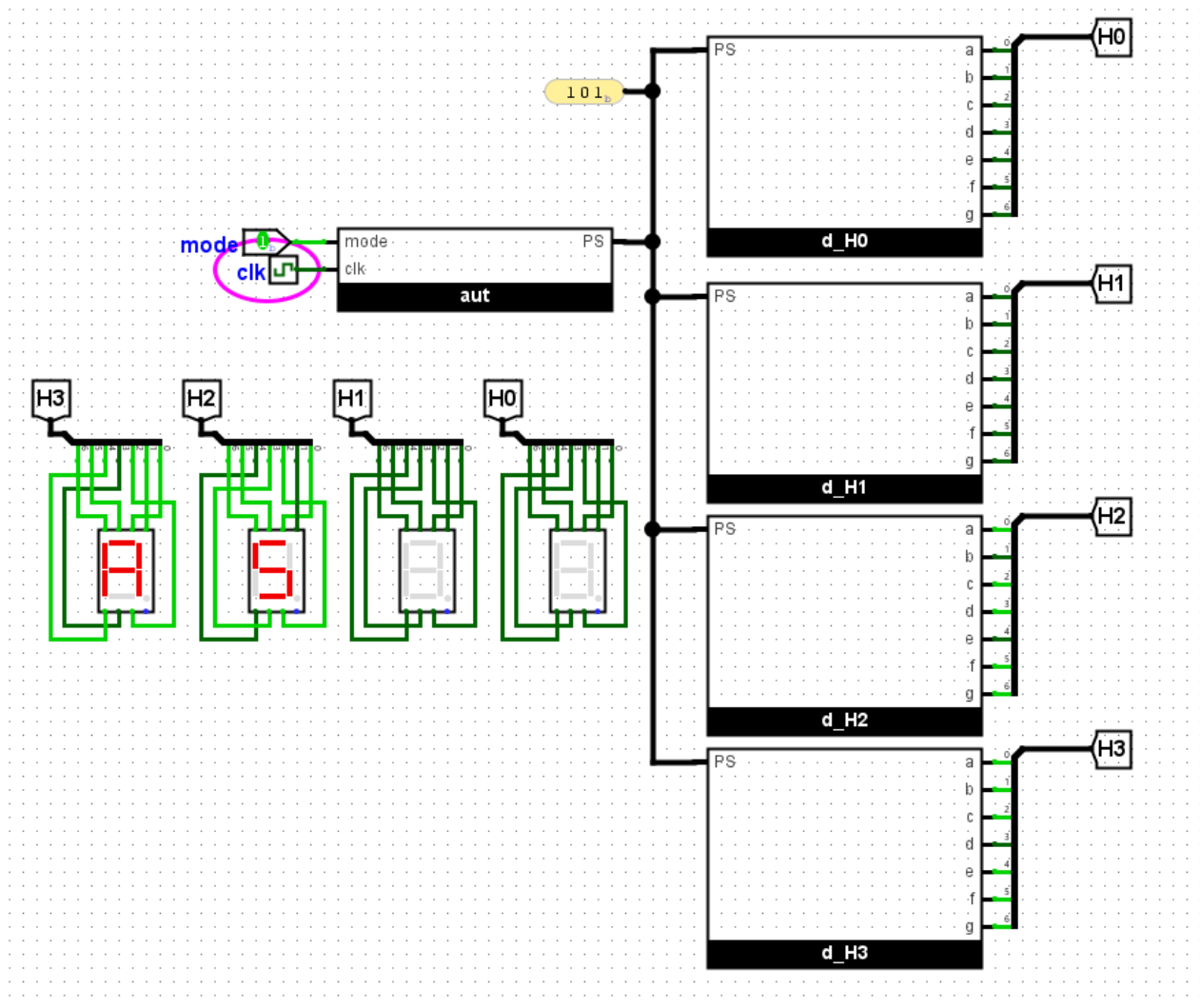
Rysunek 16: Wyświetlacz tekstu dla stanu trzeciego.



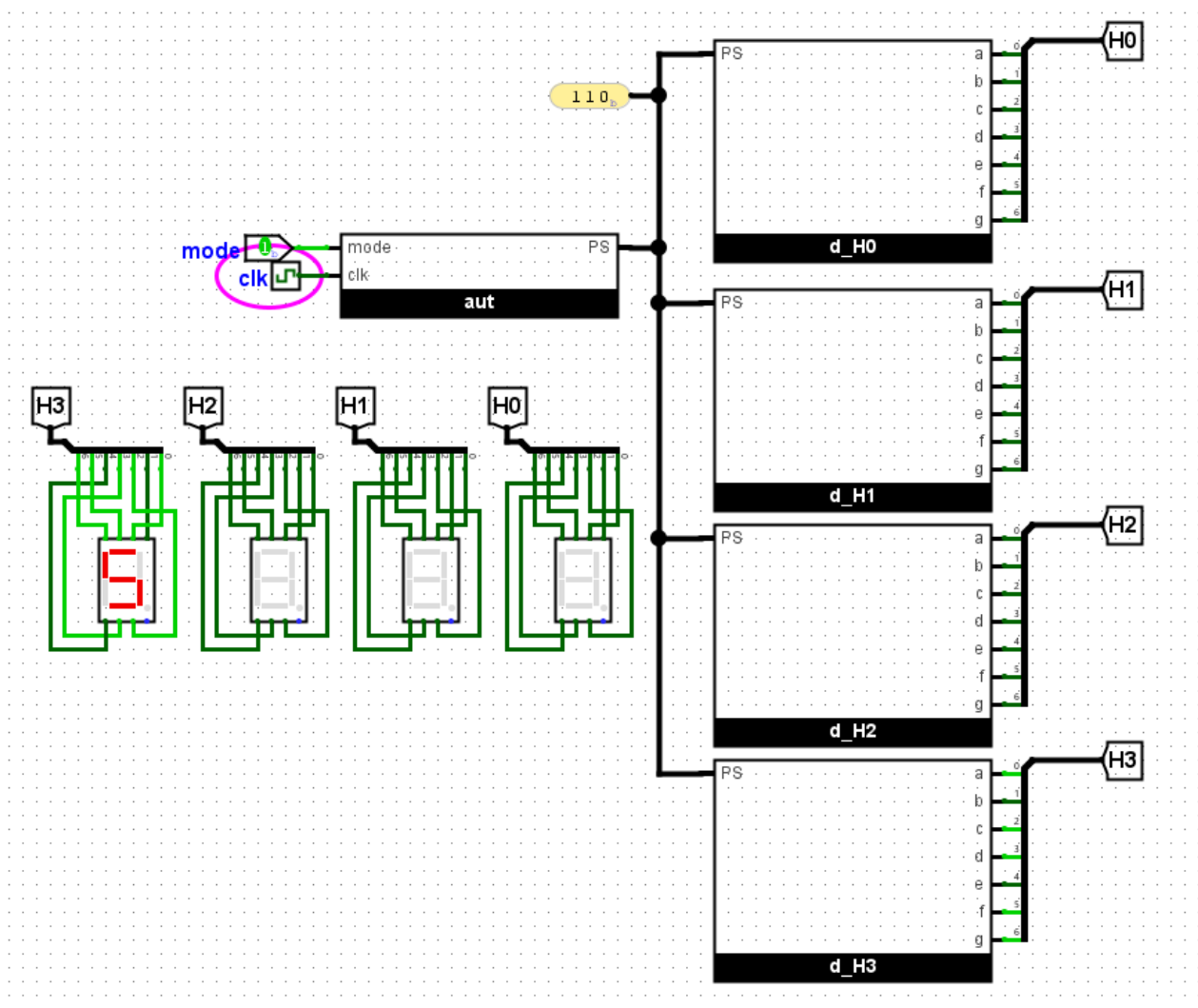
Rysunek 17: Wyświetlacz tekstu dla stanu czwartego.



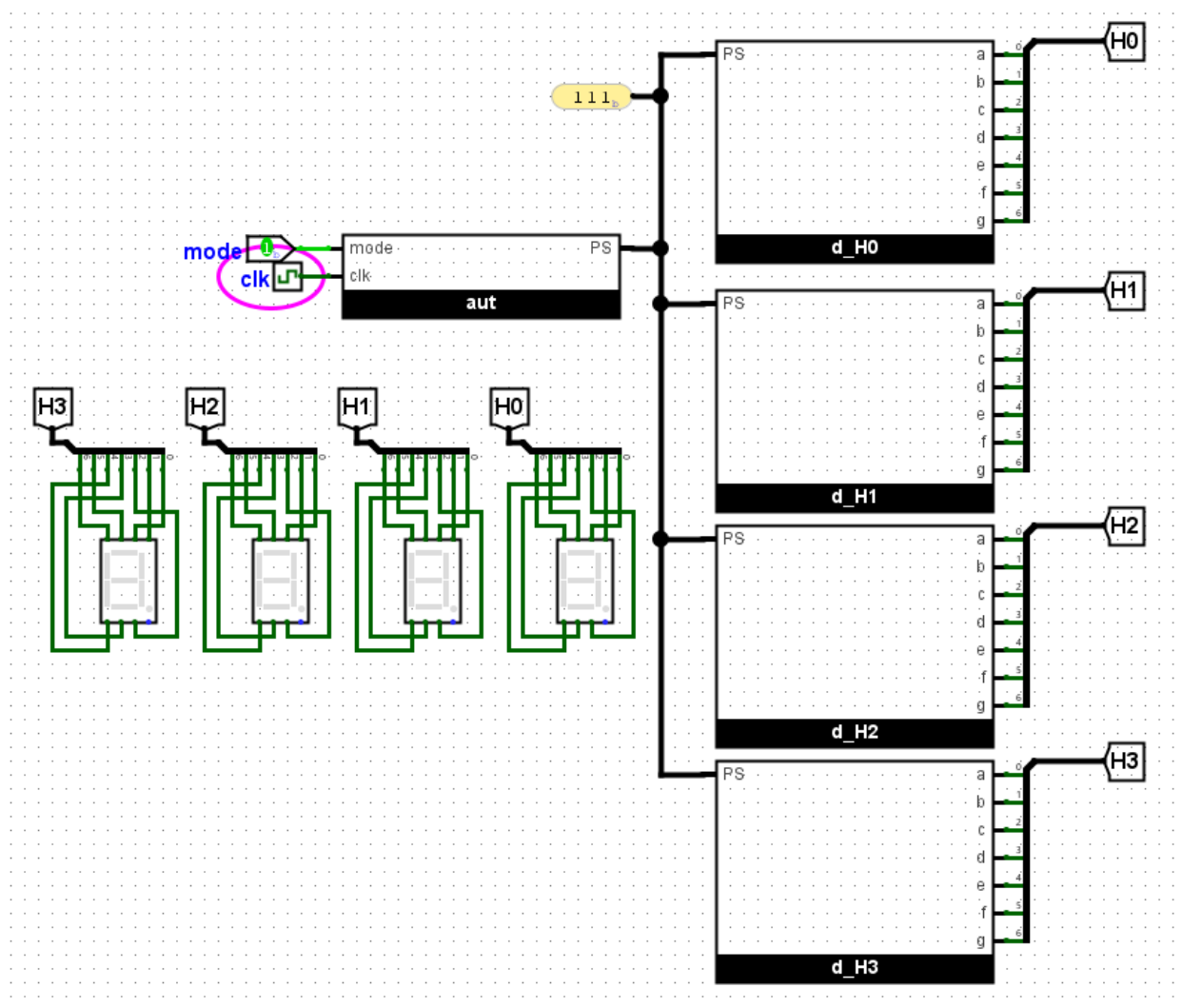
Rysunek 18: Wyświetlacz tekstu dla stanu piątego.



Rysunek 19: Wyświetlacz tekstu dla stanu szóstego.



Rysunek 20: Wyświetlacz tekstu dla stanu siódmego.



Rysunek 21: Wyświetlacz tekstu dla stanu ósmego.

W celu lepszej wizualizacji procesu testowania w folderze zip. dołączony został plik z nagraniem testowania.

5 Wnioski

Zaprojektowany układ animacji tekstu został poprawnie zaimplementowany w *Logisim-Evolution*, spełniając wszystkie założenia projektu. Animacja słowa **PEAS** działała zgodnie z oczekiwaniami – litery wyświetlały się w odpowiedniej kolejności i bez błędów. Układ działał stabilnie, a implementacja była zgodna z wymaganiami funkcjonalnymi.

Spis rysunków

1	Moduł aut przed modyfikacjami	3
2	Moduł NS przed modyfikacjami	3
3	Schemat wyświetlacza 7-segmentowego [źródło: https://cdn.forbot.pl/blog/wp-content/uploads/2016/11/kursTC_7_1_wyswietlacz_7_SEG.png]	4
4	Moduł aut po modyfikacjach.	5
5	Moduł NS po modyfikacjach.	6
6	Tabela prawdy dla wyświetlacza H3.	7
7	Układ bramkowy dla wyświetlacza H3.	8
8	Tabela prawdy dla wyświetlacza H2.	9
9	Układ bramkowy dla wyświetlacza H2.	10
10	Tabela prawdy dla wyświetlacza H1.	11
11	Układ bramkowy dla wyświetlacza H1.	12
12	Tabela prawdy dla wyświetlacza H0.	13
13	Układ bramkowy dla wyświetlacza H0.	14
14	Wyświetlacz tekstu dla stanu pierwszego.	15
15	Wyświetlacz tekstu dla stanu drugiego.	16
16	Wyświetlacz tekstu dla stanu trzeciego.	17
17	Wyświetlacz tekstu dla stanu czwartego.	18
18	Wyświetlacz tekstu dla stanu piątego.	19
19	Wyświetlacz tekstu dla stanu szóstego.	20
20	Wyświetlacz tekstu dla stanu siódmego.	21
21	Wyświetlacz tekstu dla stanu ósmego.	22

Spis tabel

1	Wyświetlane dane w zależności od stanów wejściowych dla słowa <i>PEAS</i> . ("*" oznacza pusty wyświetlacz.)	4
---	--	---

2	Dane wejściowe do wyświetlacza w zależności od wyświetlanej literki)	5
---	--	---