A Hard Problem In A Boolean Asynchronous Freezing Cellular Automata

Eric Goles, Diego Maldonado, Pedro Montealegre, and Martín Ríos-Wilson

Facultad de Ingeniería y Ciencias

International Workshop on Boolean Neworks, 9 de enero de 2020

Cellular automata

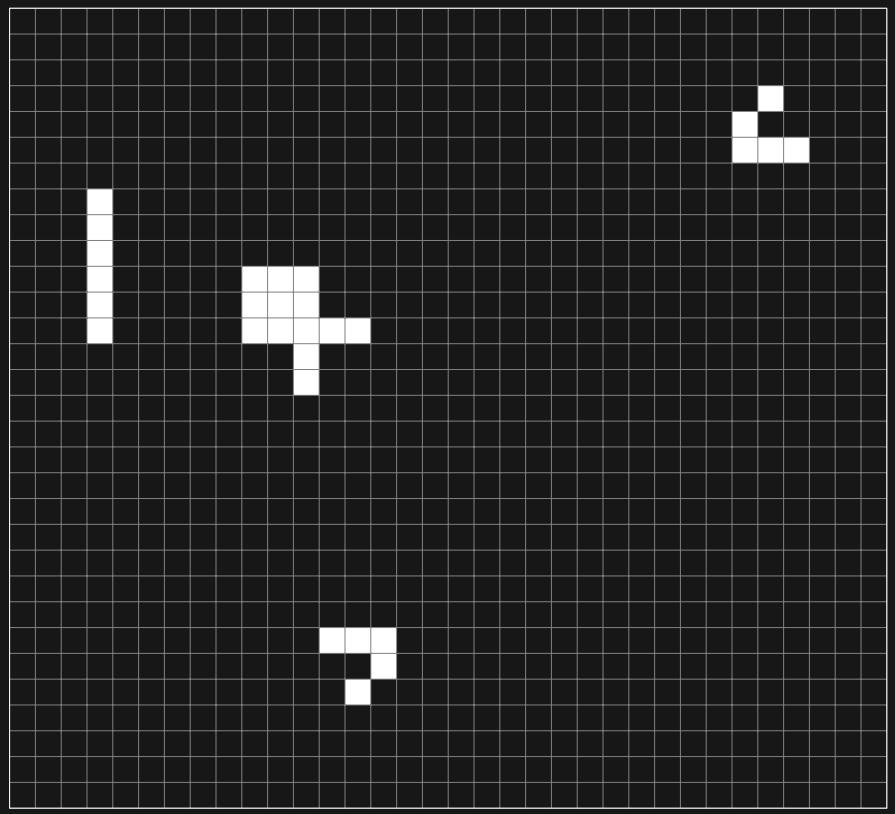
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

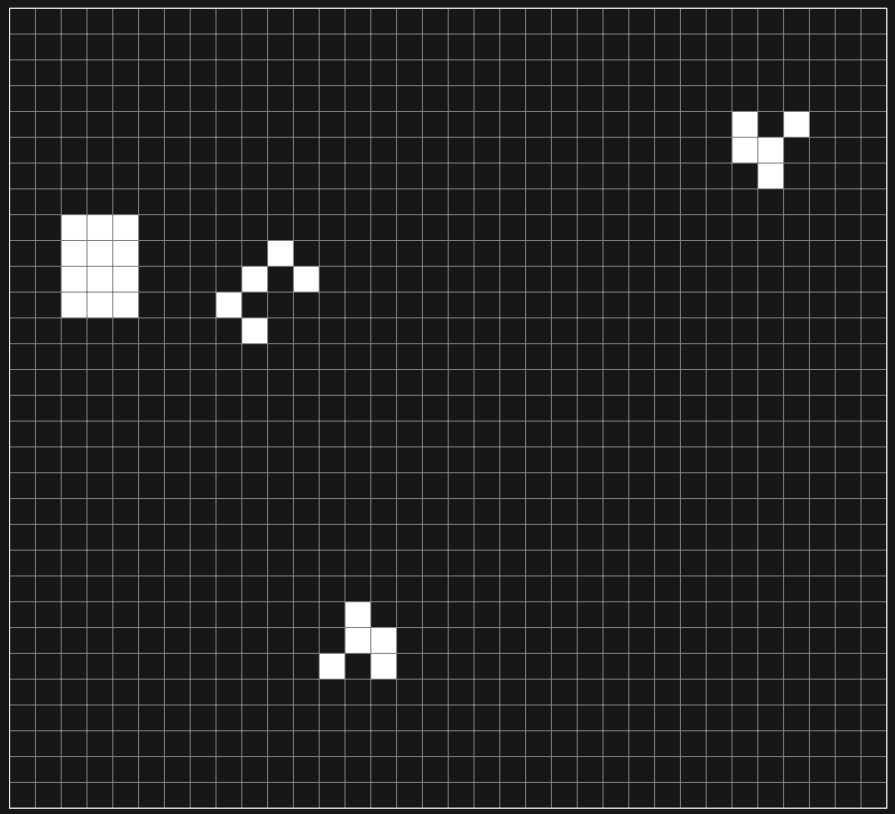
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

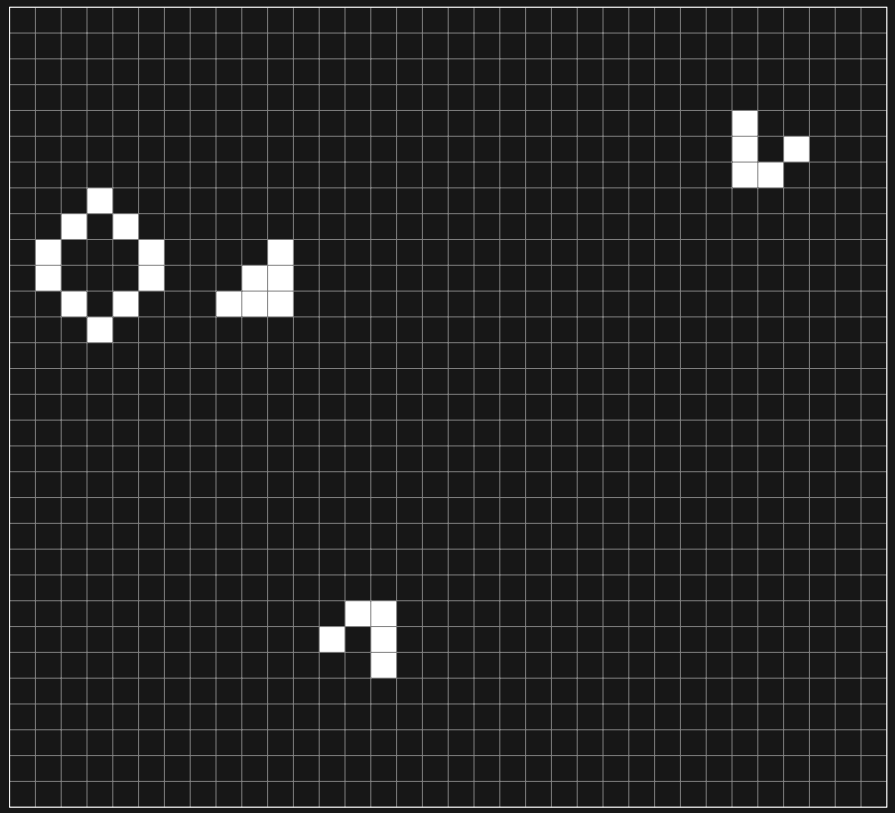
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

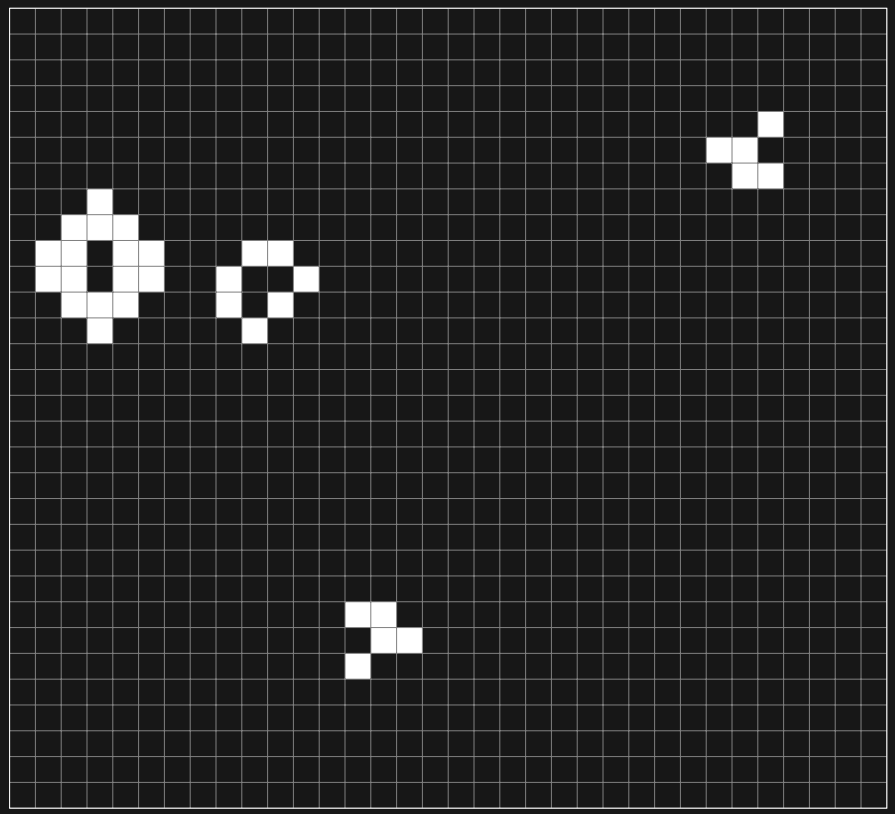
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

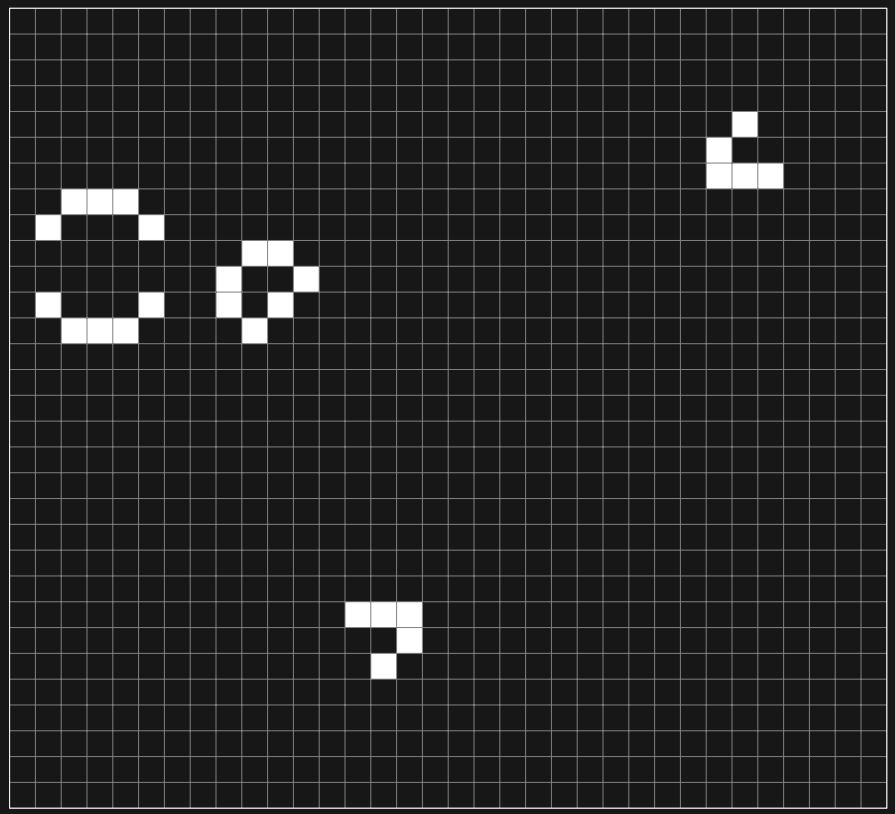
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

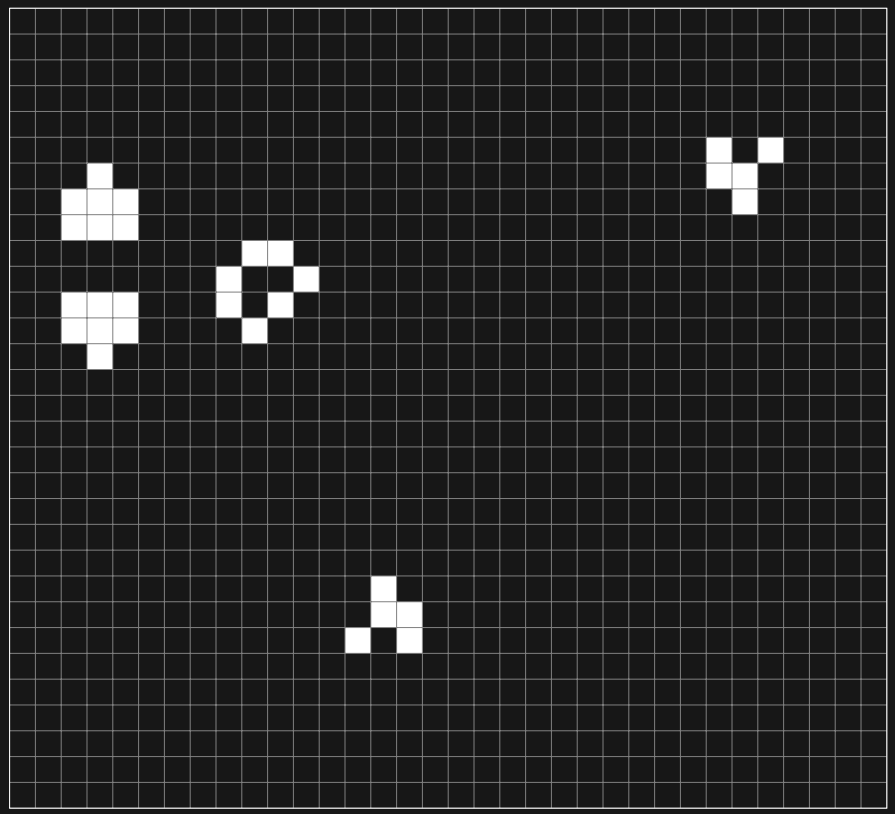
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

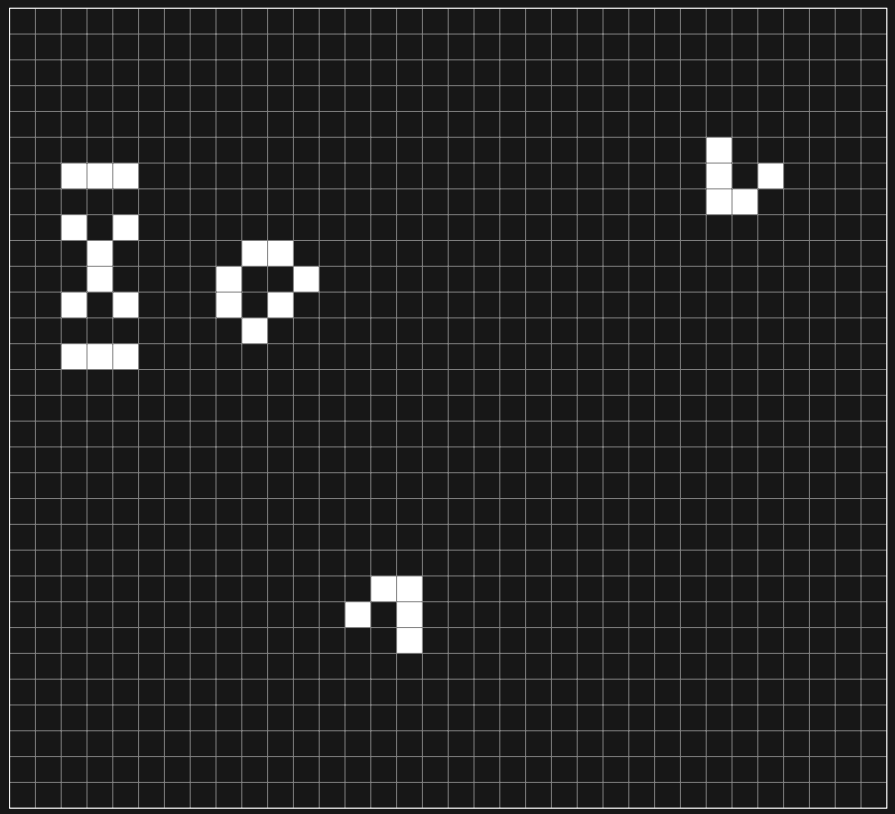
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

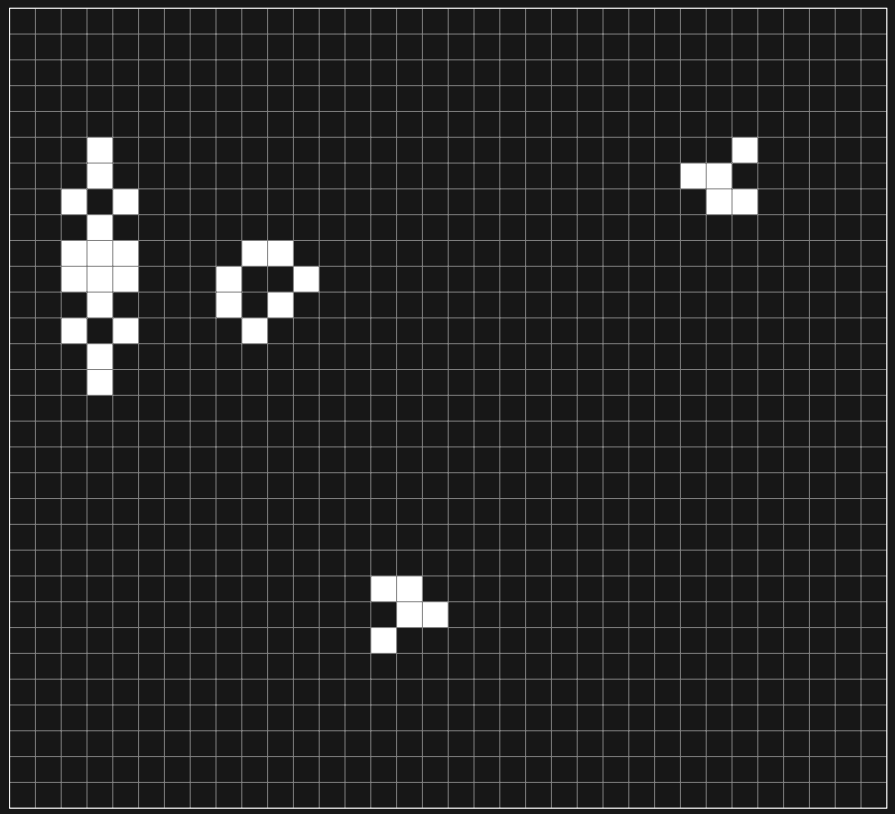
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

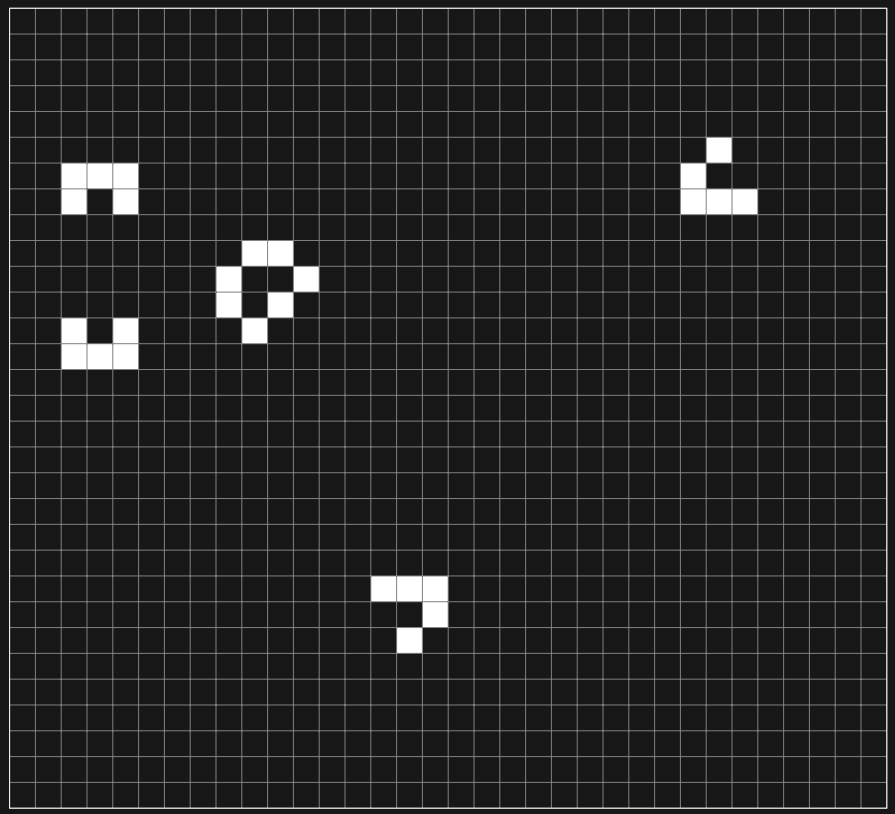
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

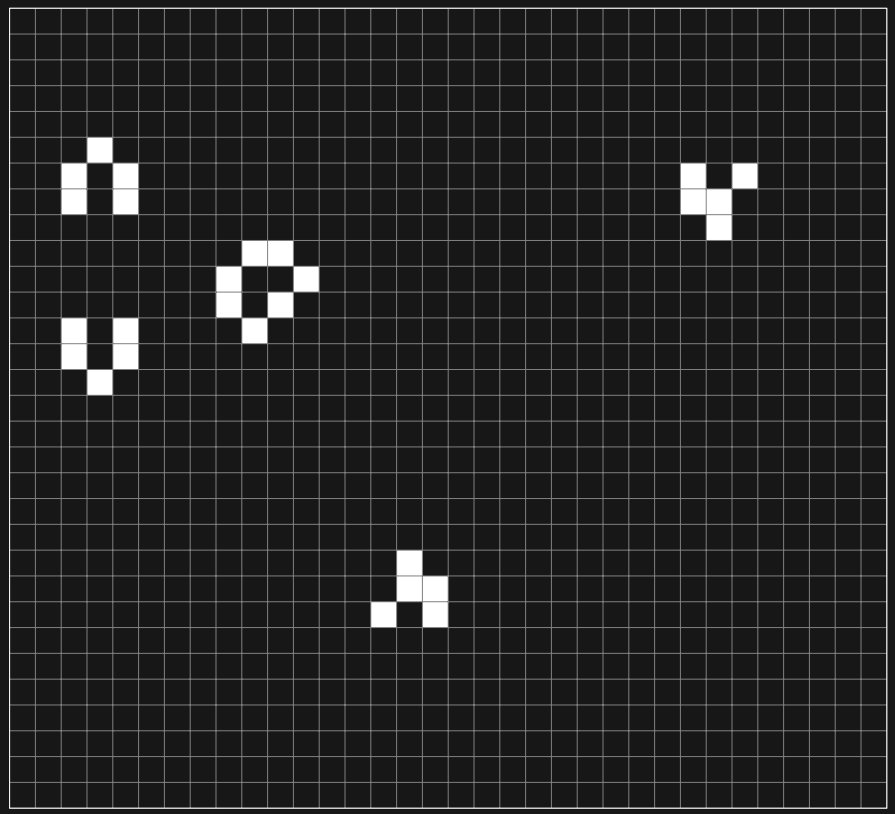
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

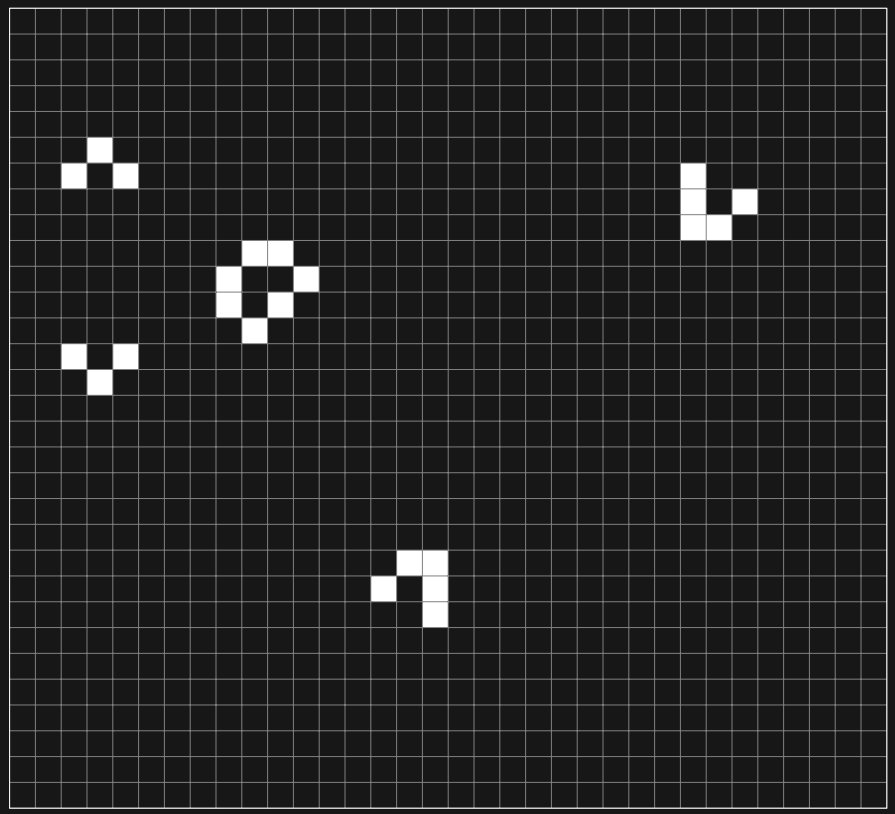
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

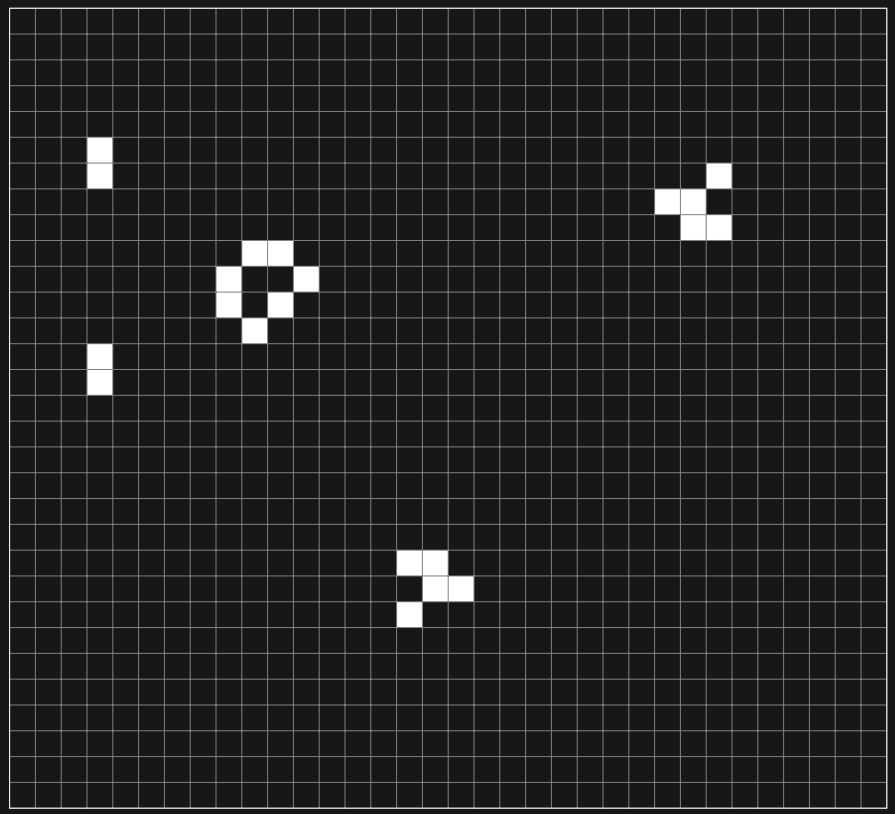
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

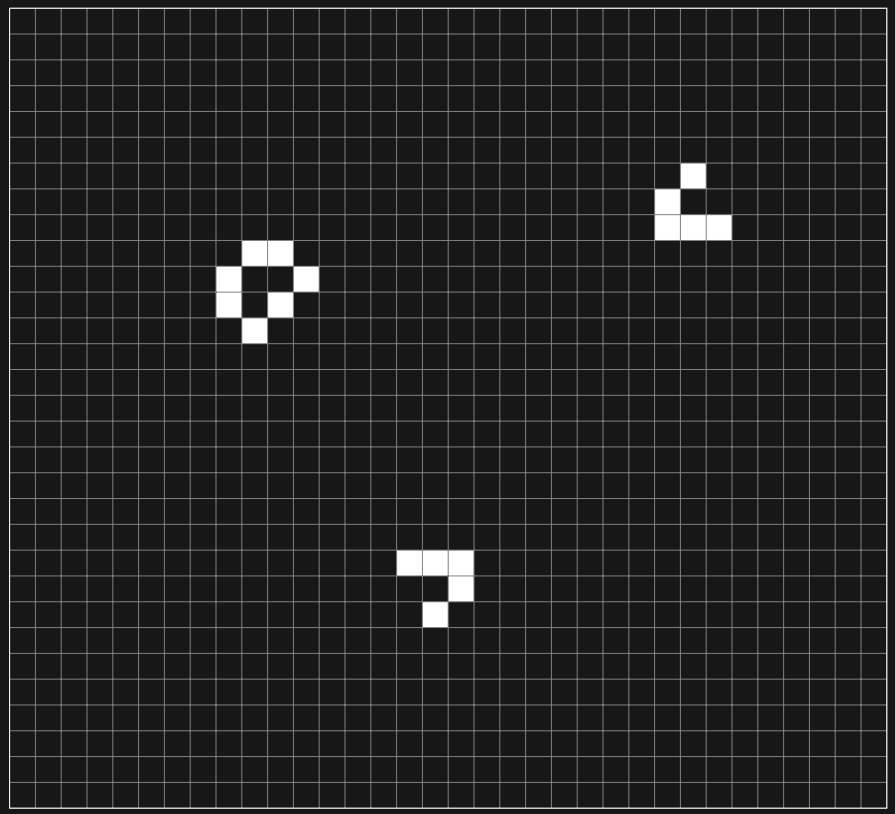
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

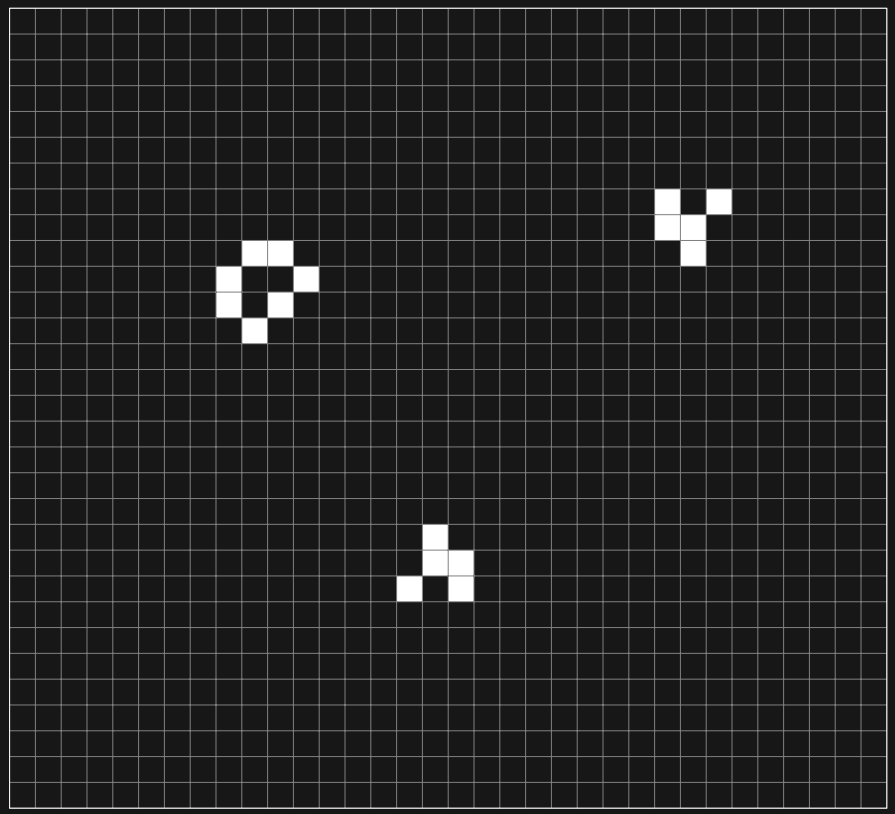
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

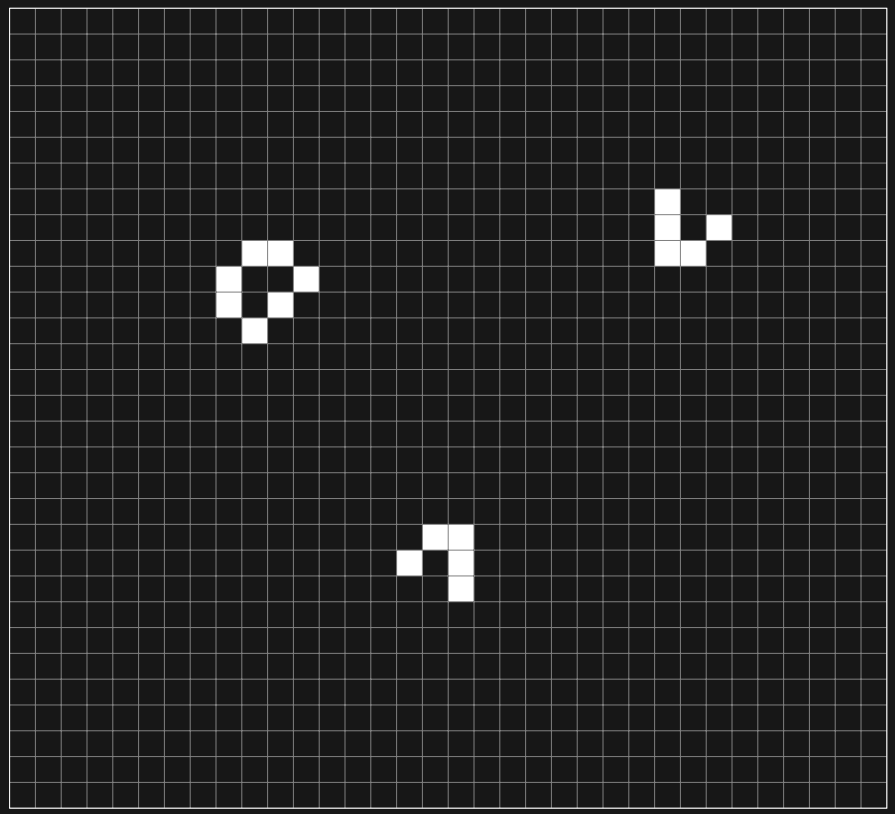
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

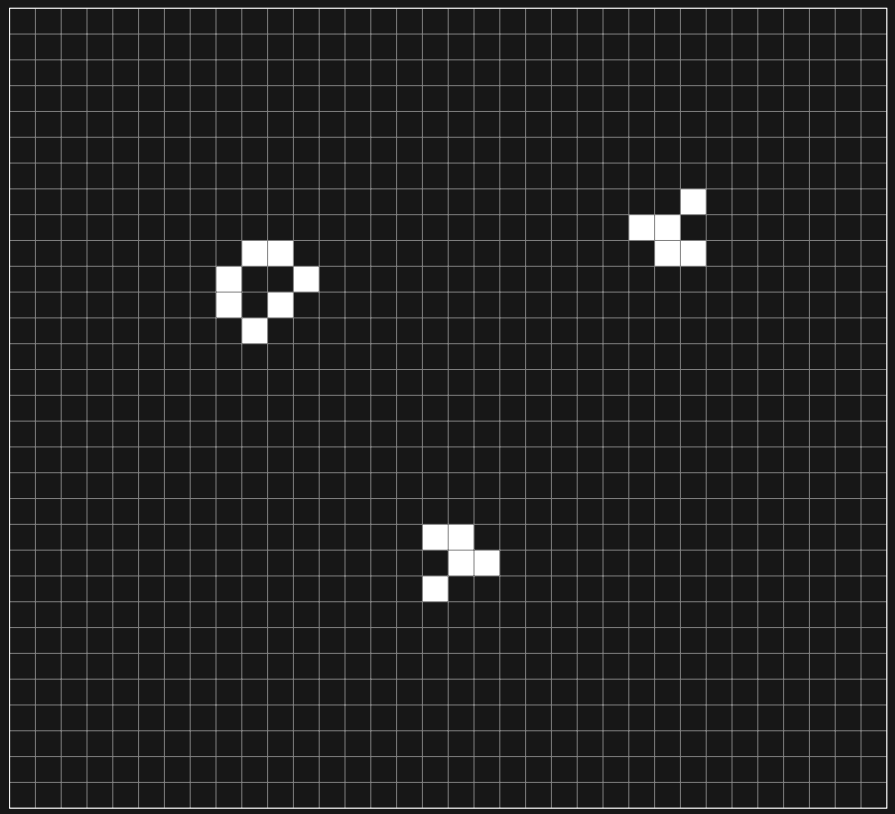
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

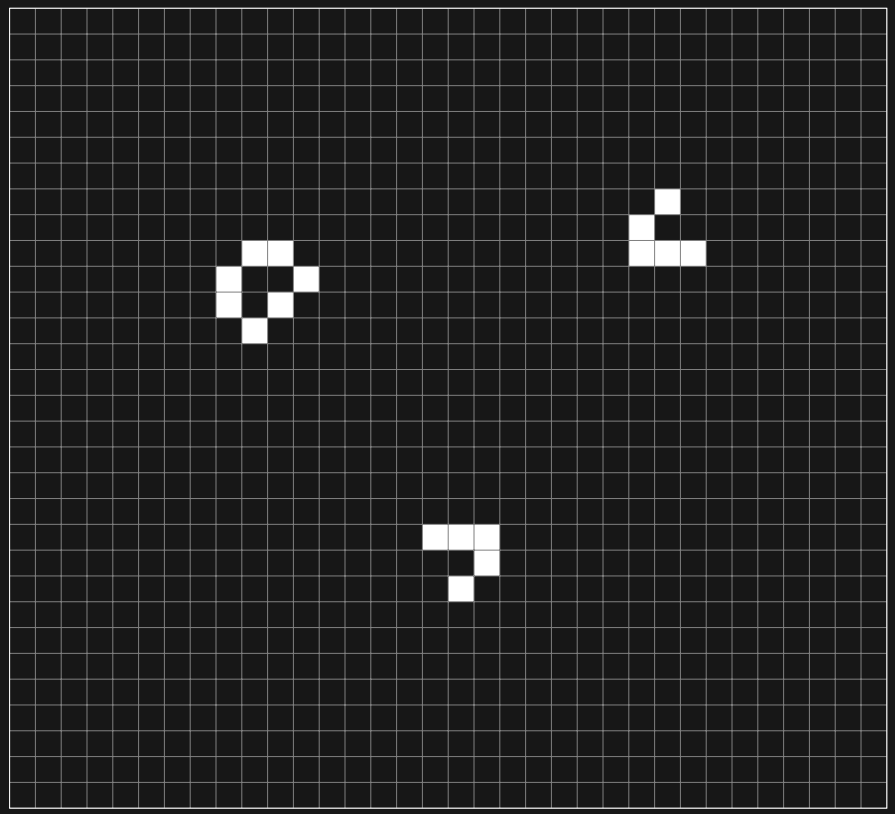
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

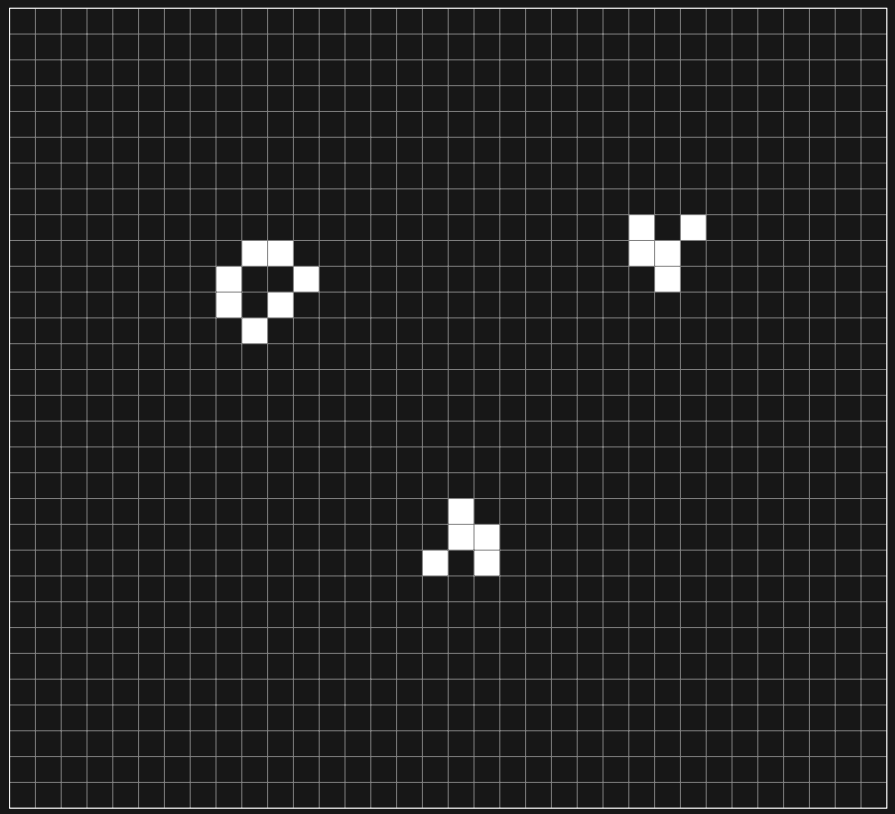
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

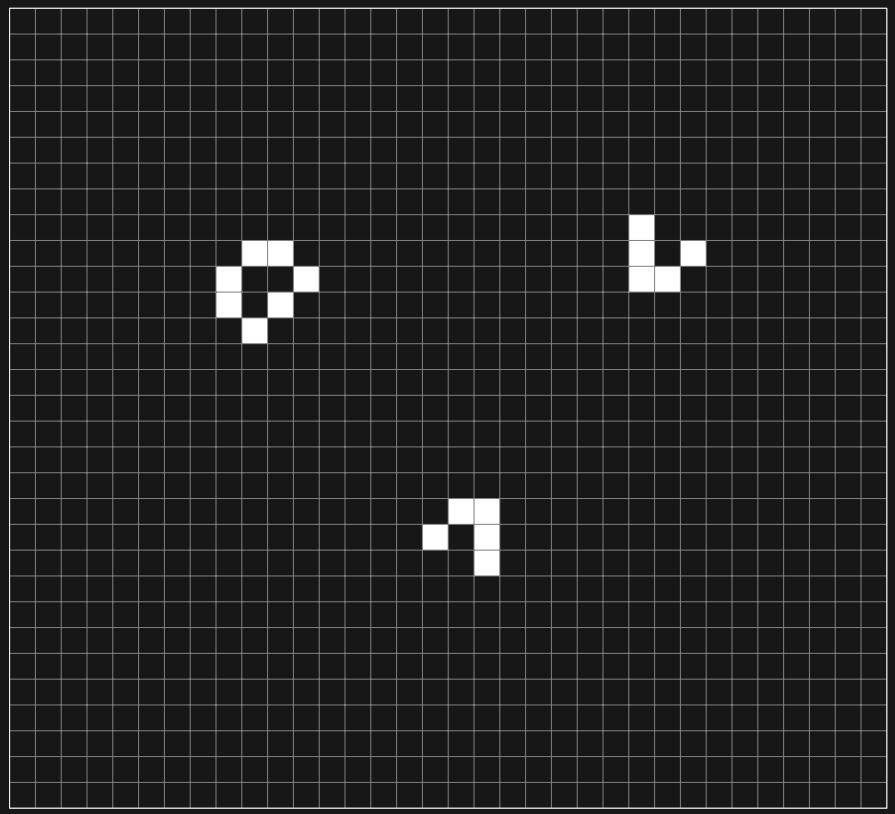
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

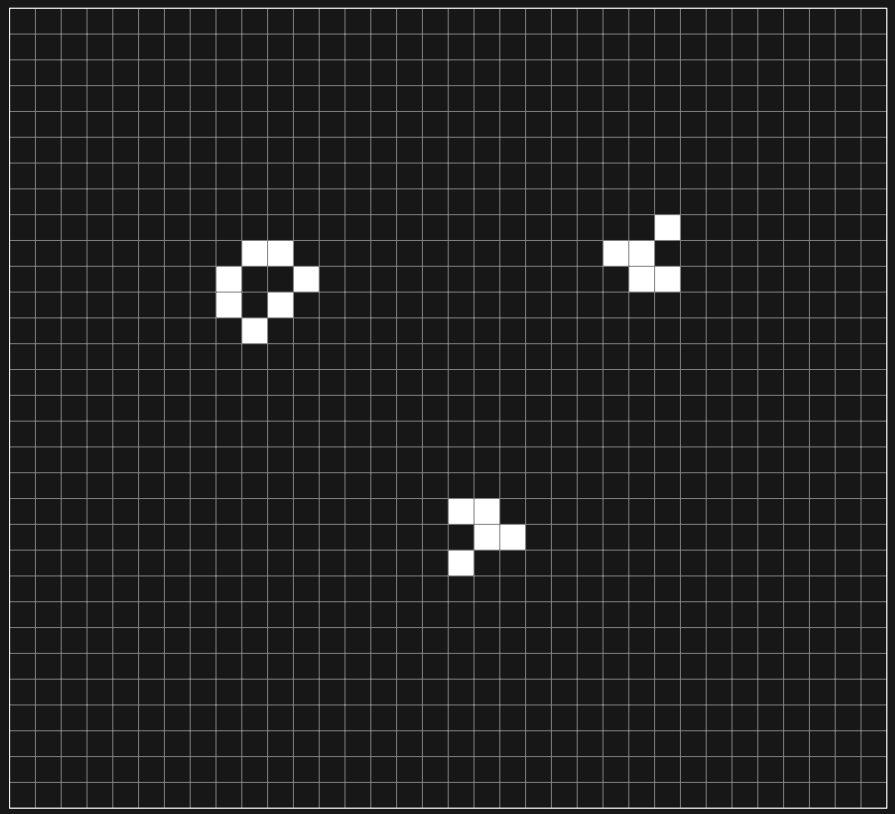
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

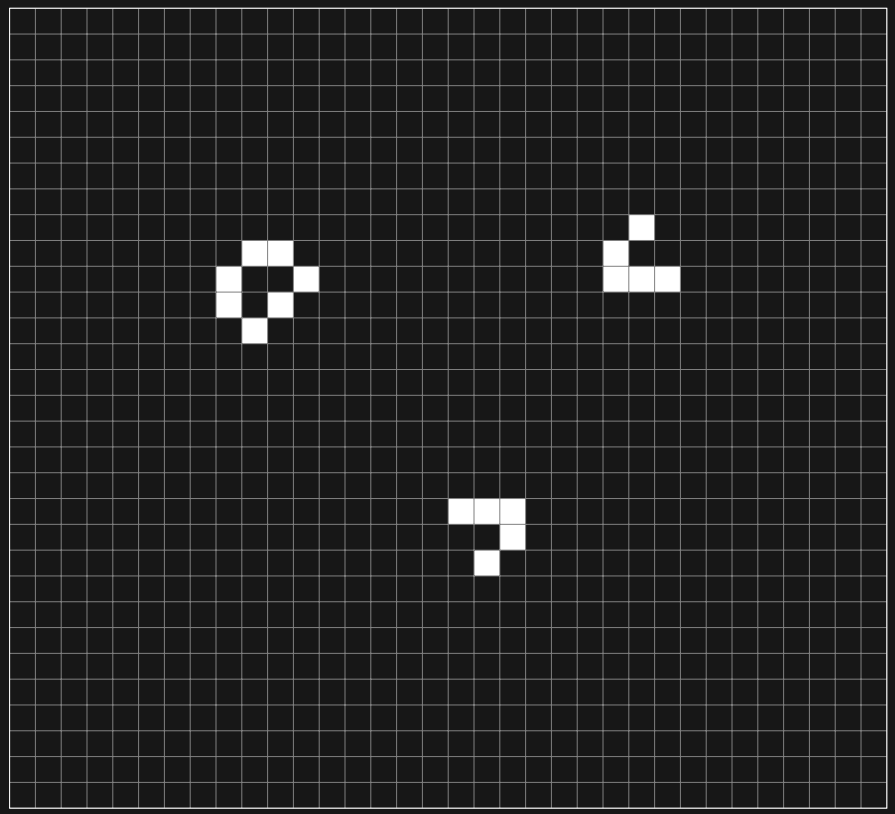
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

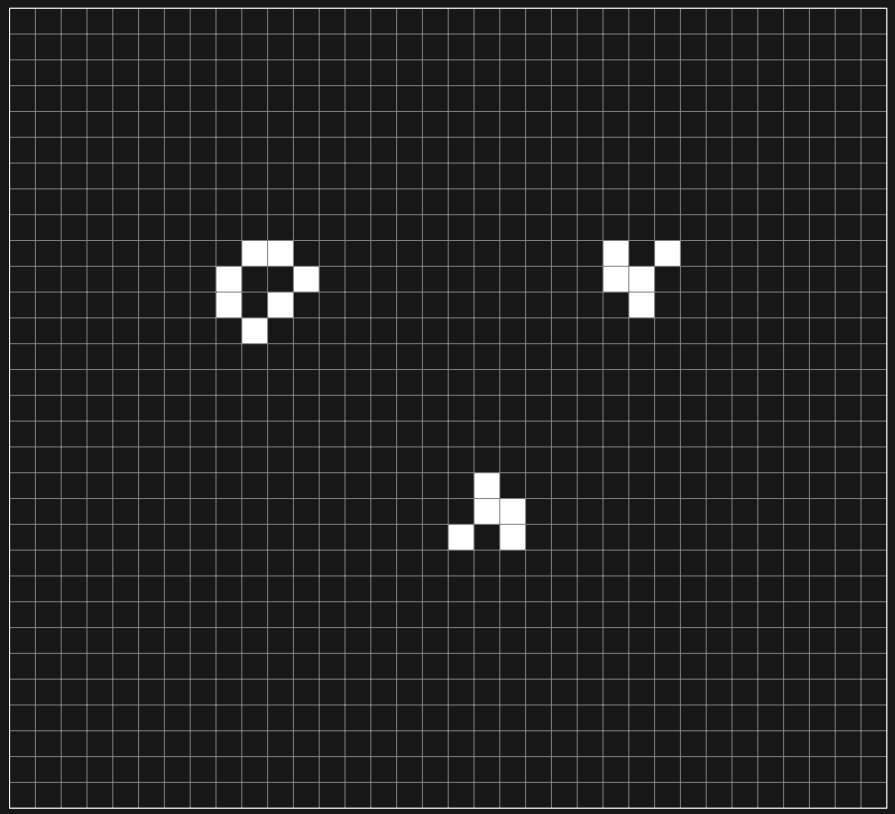
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

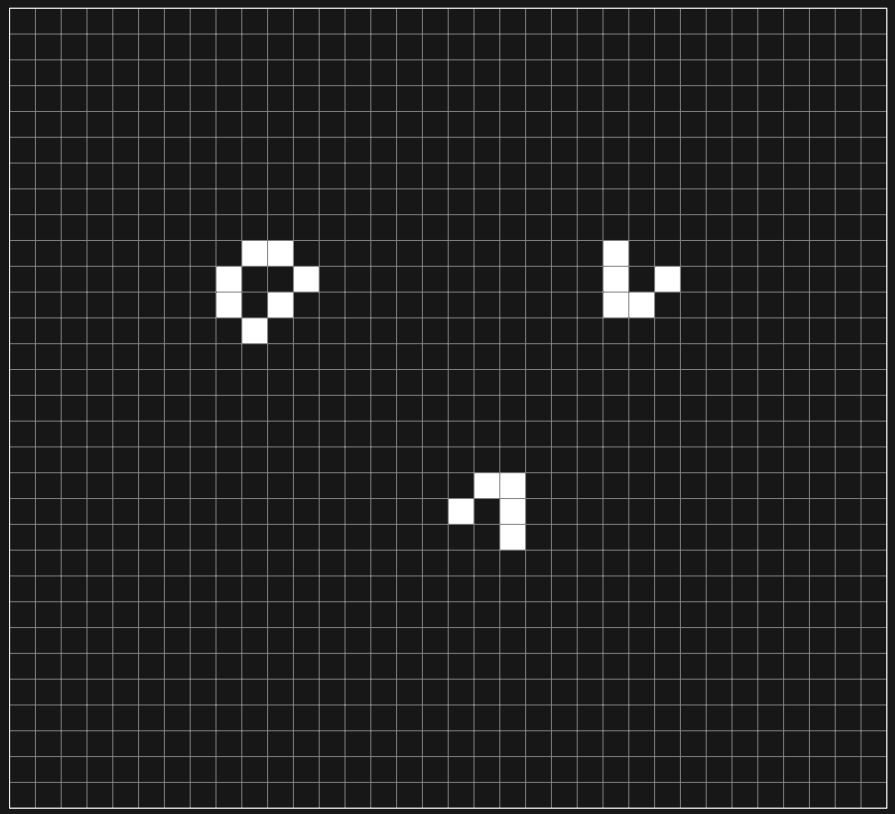
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

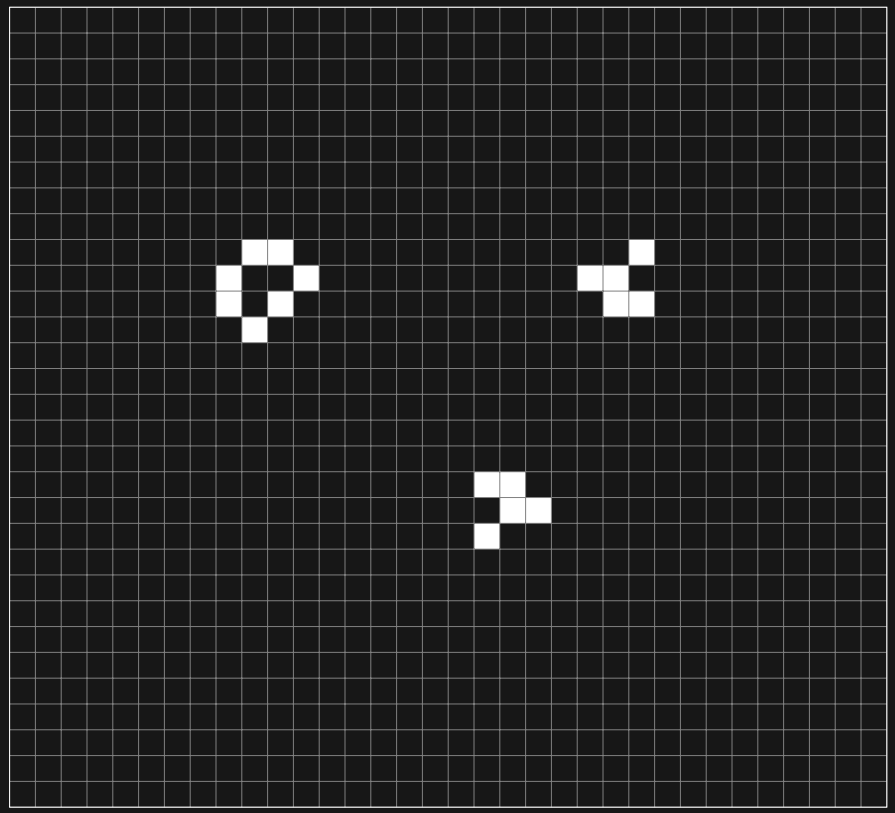
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

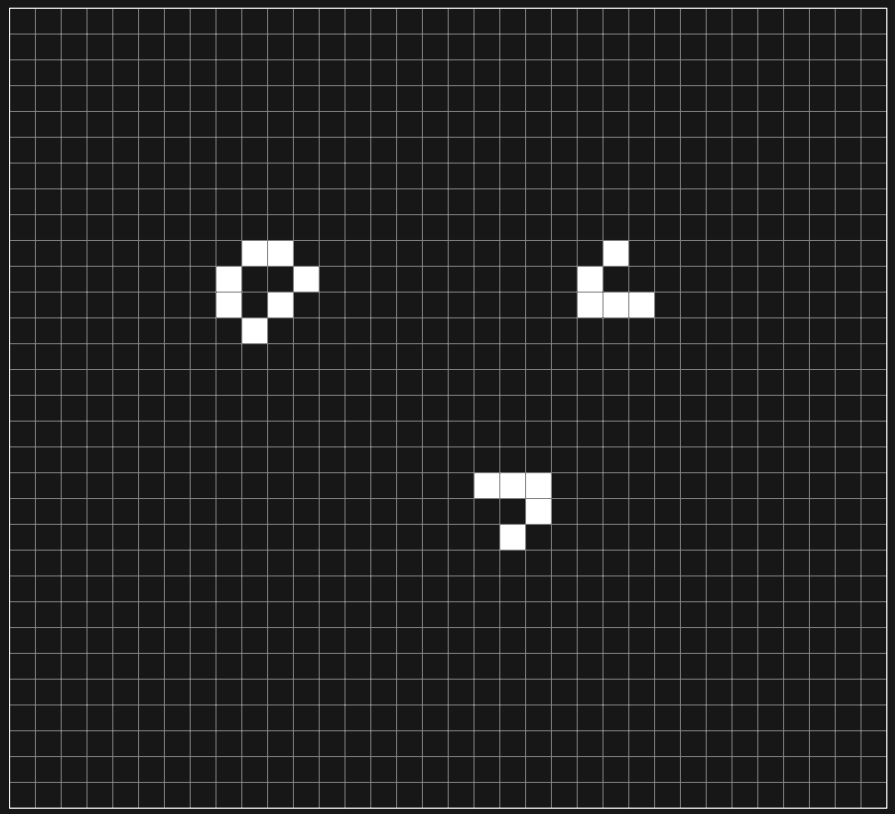
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

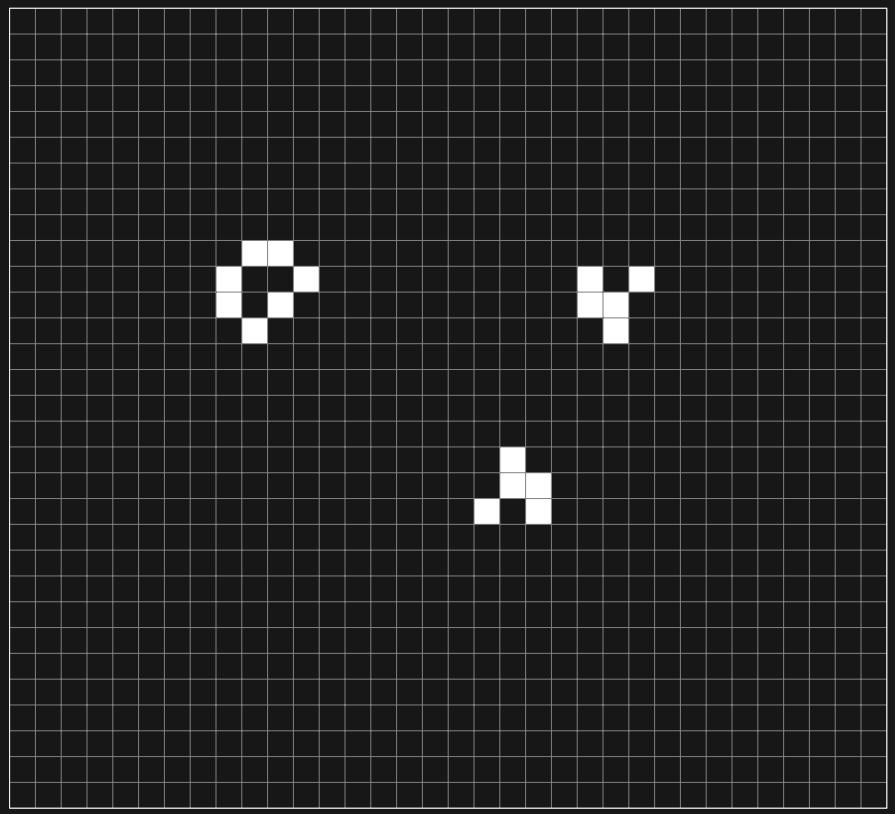
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

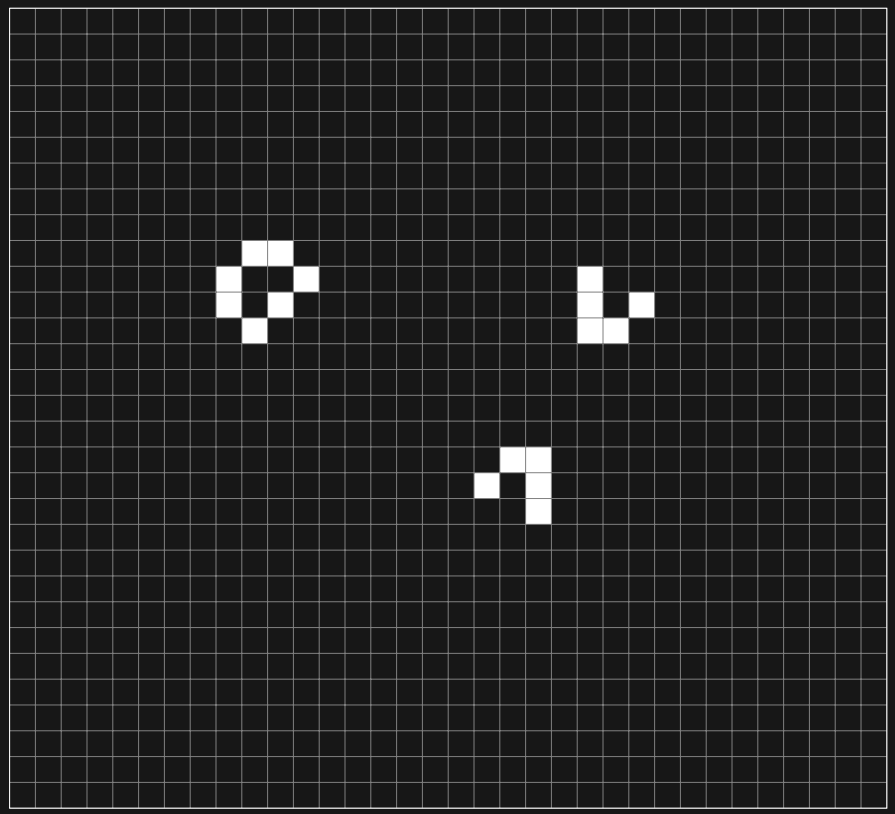
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

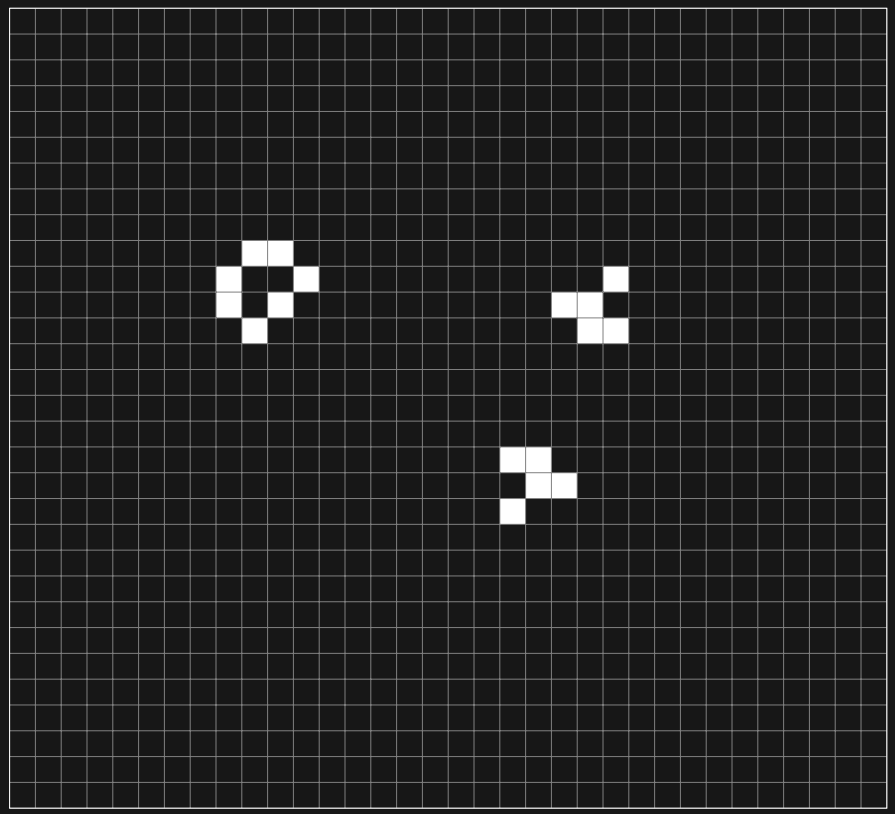
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

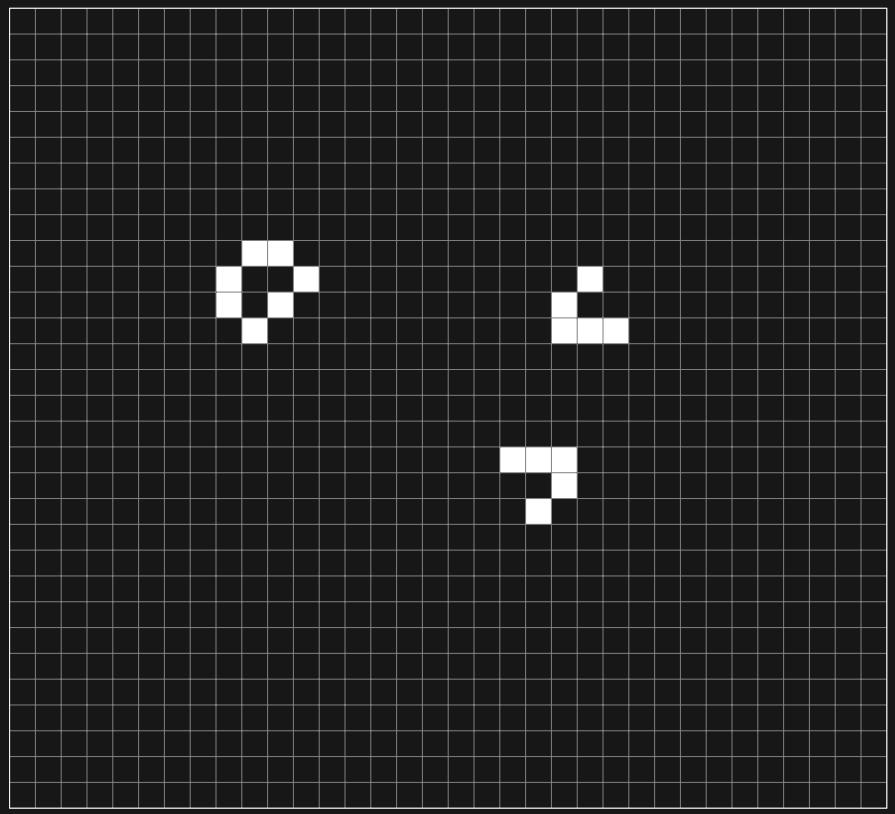
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

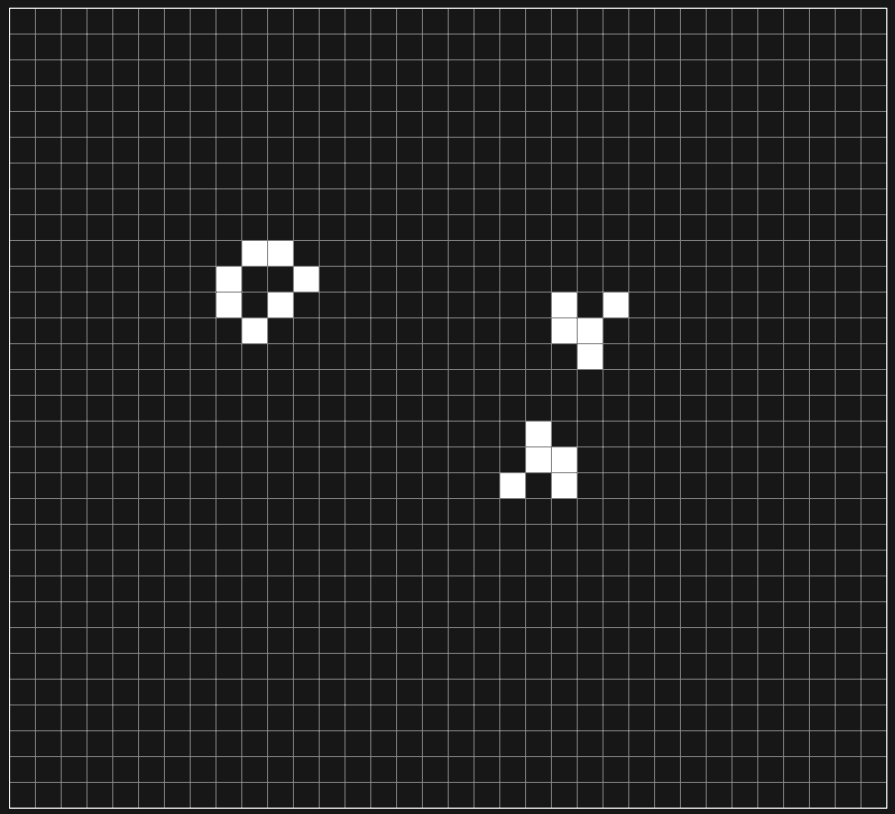
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

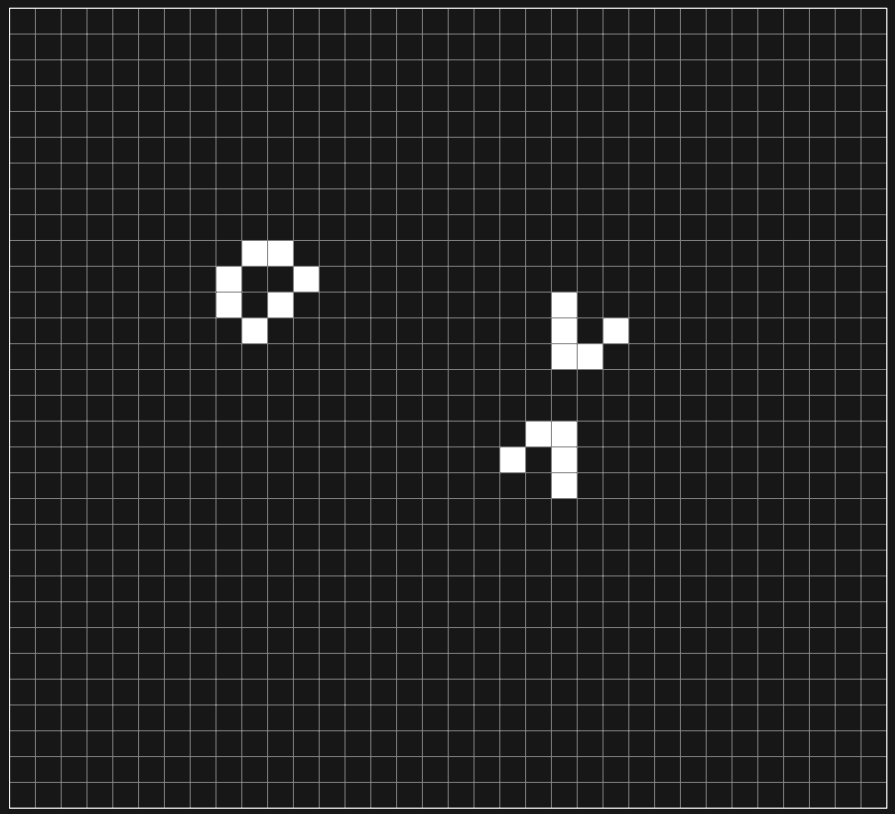
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

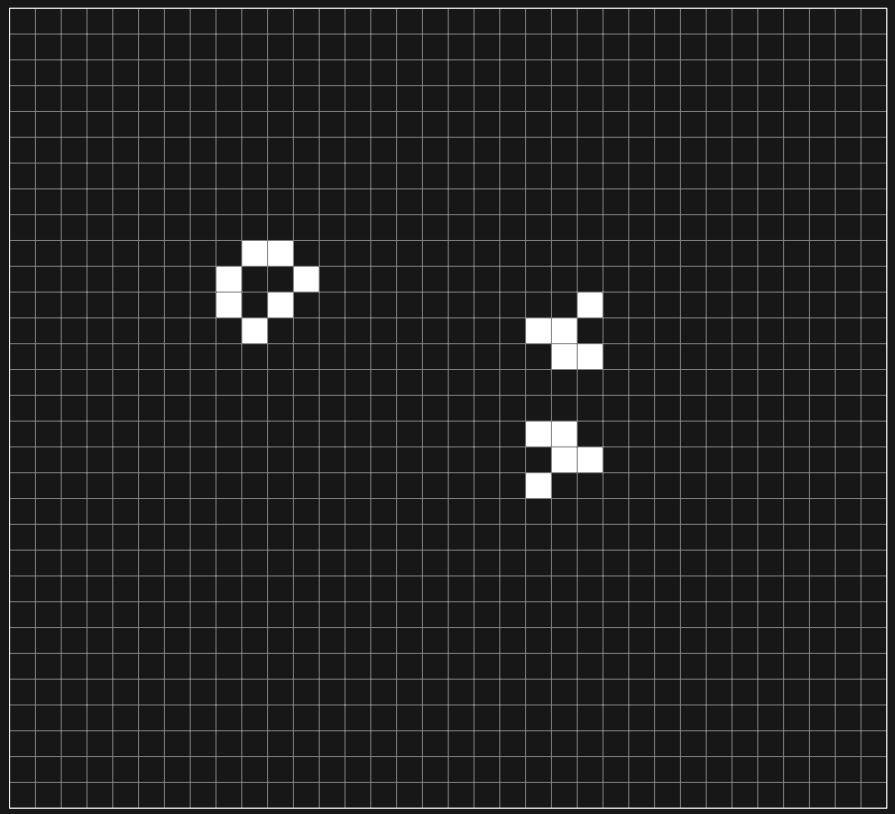
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

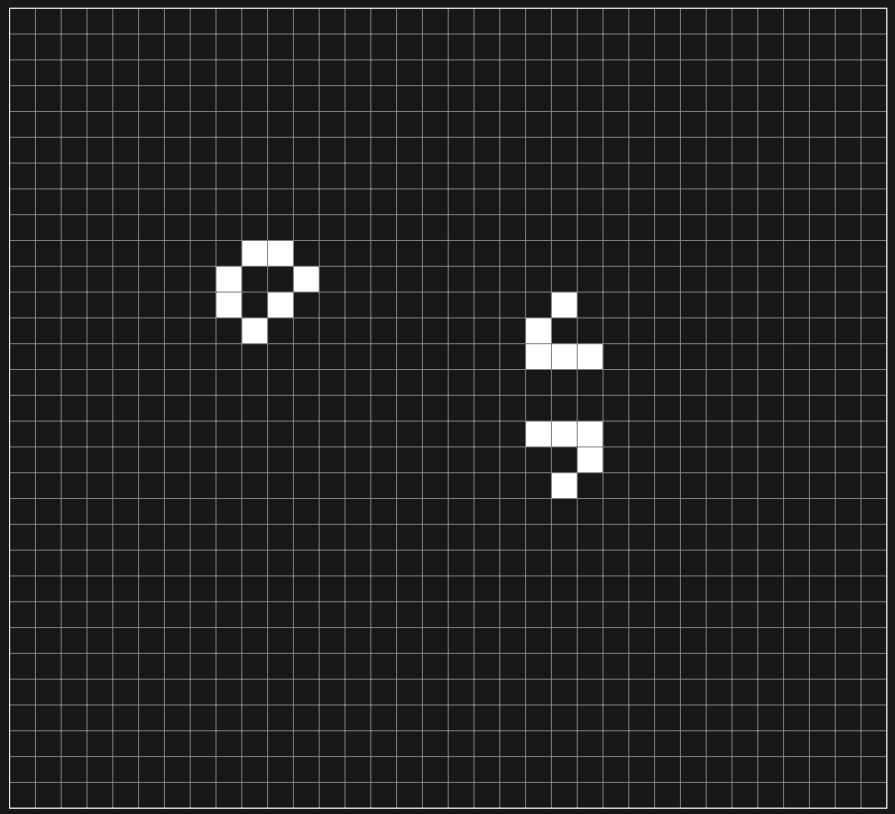
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

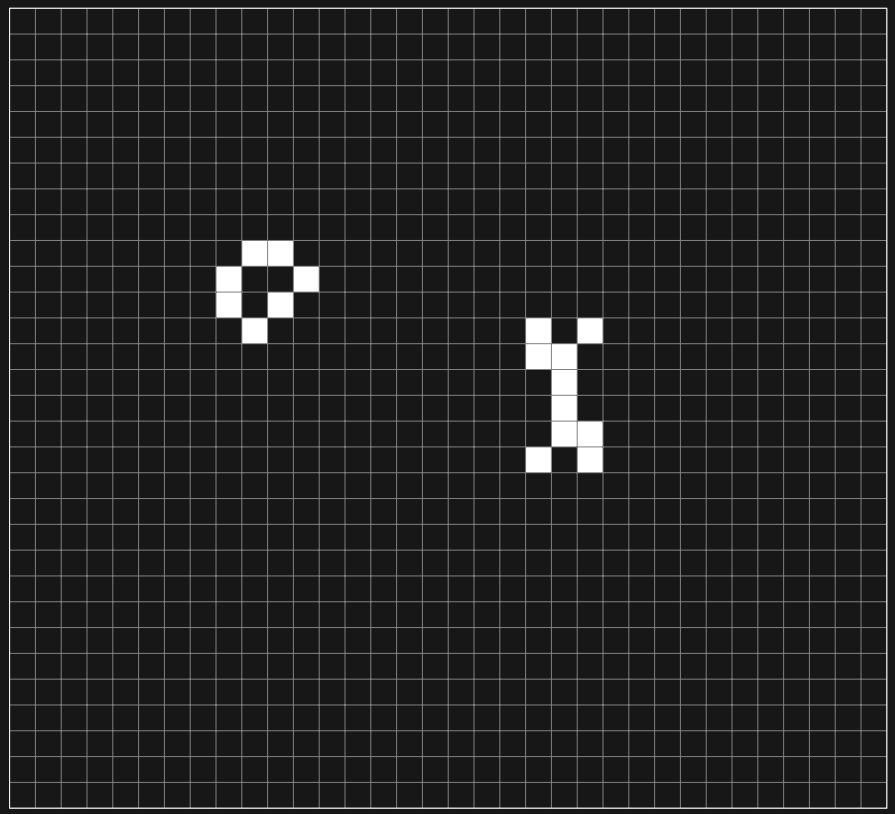
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Cellular automata

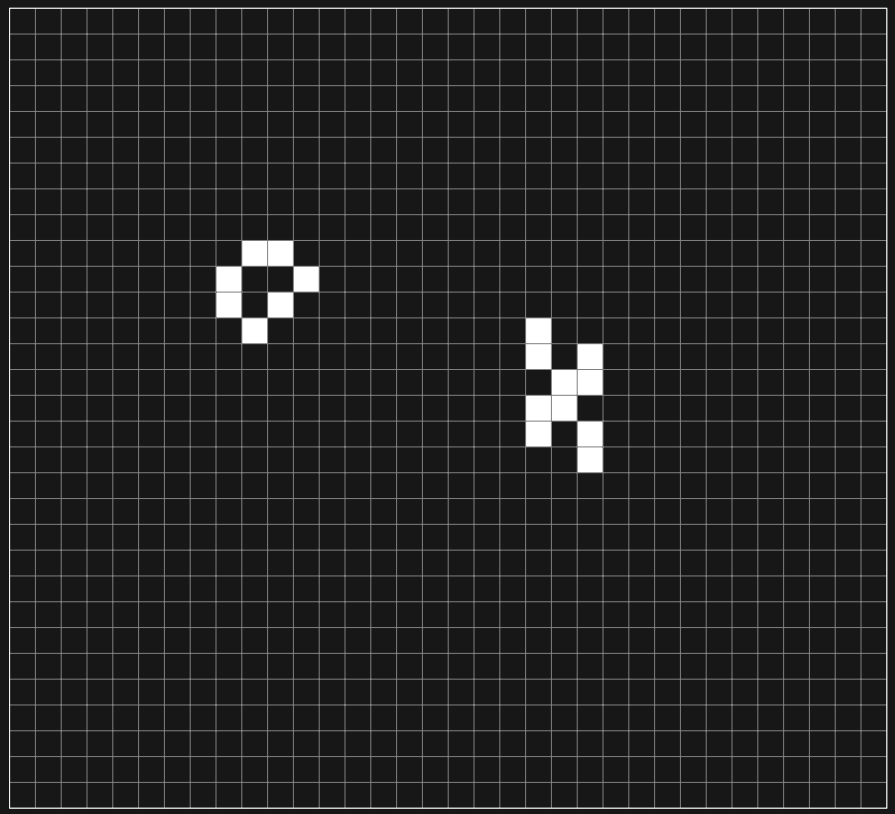
Recipe

Ingredients:

- n States
- 1 Grid with cells
- 1 Neighborhood
- 1 Local function

Directions:

Apply the local function on each cell in parallel (synchronously) once. Repeat until obtain a dynamical system.



Remark

In general, CA shows a complex behavior.

Asynchronous Cellular Automata

An Asynchronous Cellular Automaton is a function $F : Q^{\mathbb{Z}^d} \to Q^{\mathbb{Z}^d}$:

$$F^{\sigma(0)}(x) = x; \qquad F^{\sigma(t)}(x)_z = \begin{cases} f(F^{\sigma(t-1)}(x)_{N(z)}) & \text{if } z = \sigma(t) \\ x_z & \text{otherwise.} \end{cases}$$

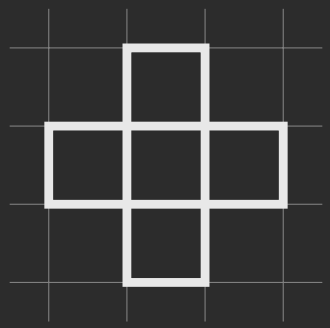
- $N \subset \mathbb{Z}^d$ is called the neighborhood
- f is the local function
- $\sigma : \mathbb{N} \to \mathbb{Z}^d$ is the update scheme

Remark

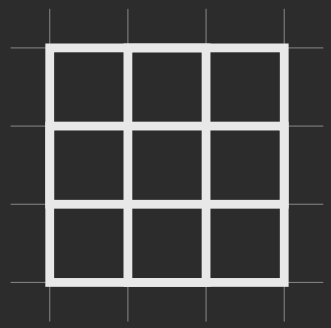
In asynchronous cellular automata only changes one cell at each time step.

Neighborhoods and Boolean Network

Main neighborhoods

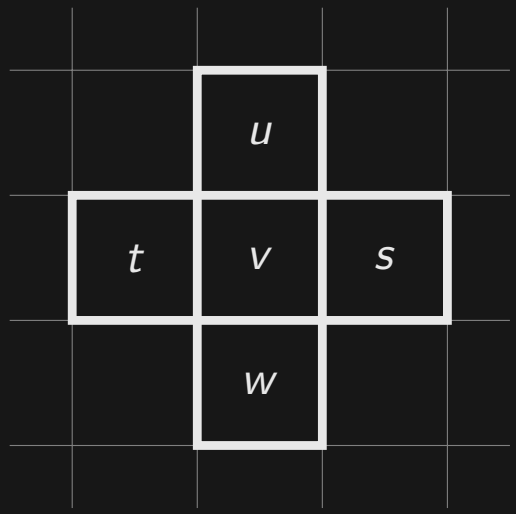


von Neumann neighborhood

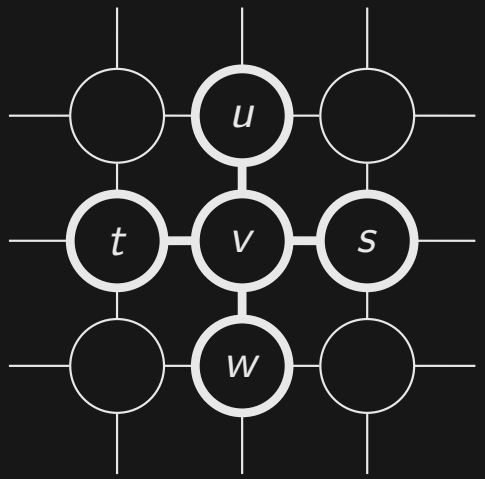


Moore neighborhood

Neighborhoods and Boolean Network

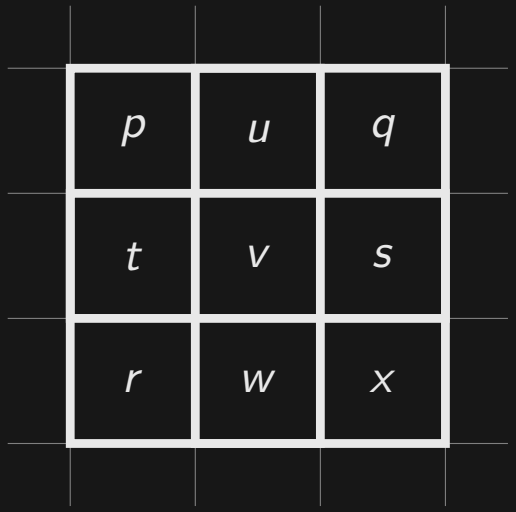


von Neumann Cellular automata grid

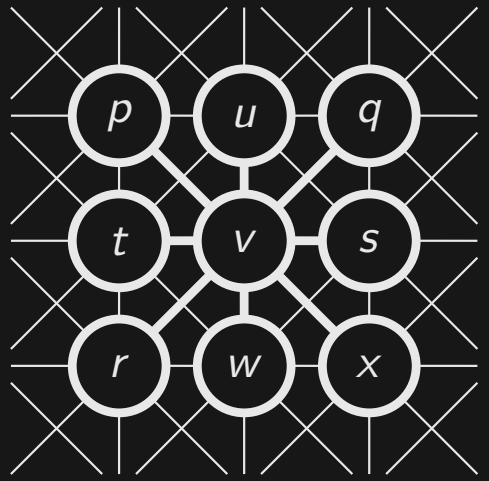


Boolean network interaction graph

Neighborhoods and Boolean Network



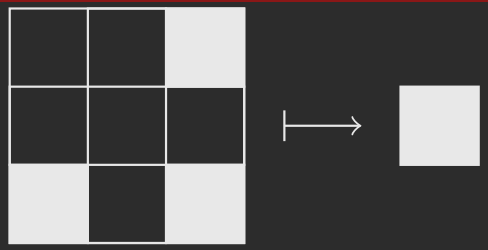
Moore Cellular automata grid

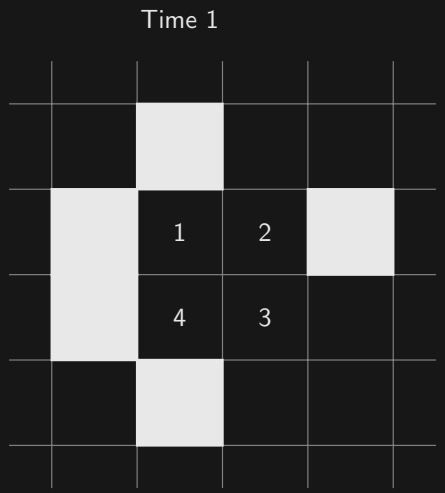


Boolean network interaction graph

Example: Life without Death.

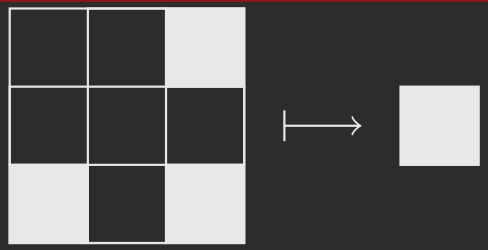
Local funcation of Life without death

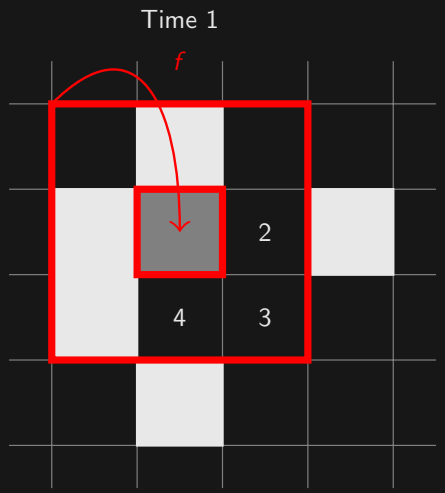
f:  Or any permutation



Example: Life without Death.

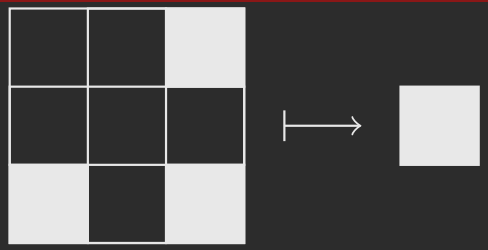
Local funcation of Life without death

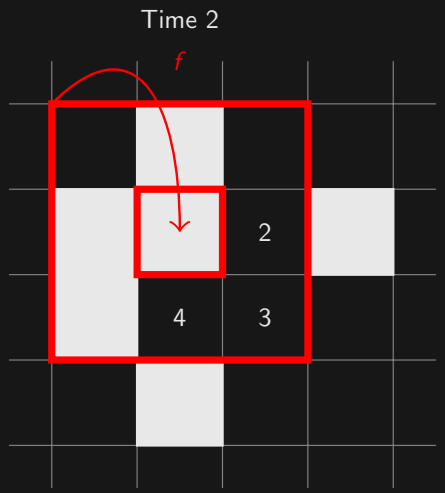
f:  Or any permutation



Example: Life without Death.

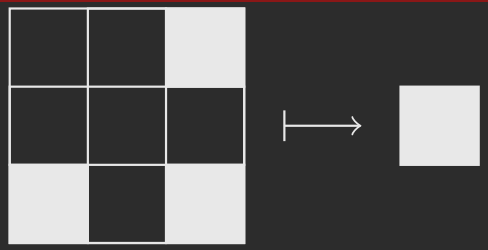
Local funcation of Life without death

f:  Or any permutation

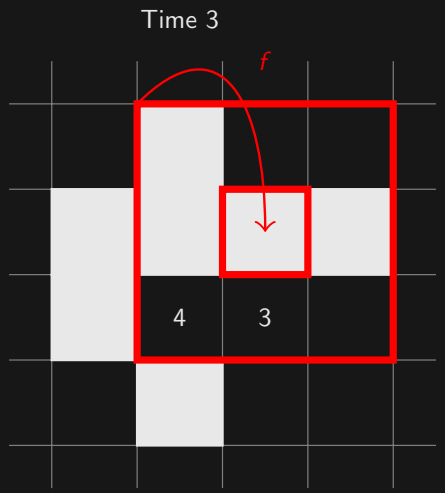


Example: Life without Death.

Local funcation of Life without death

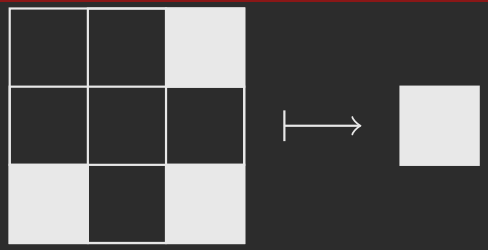
f:  $\longmapsto$

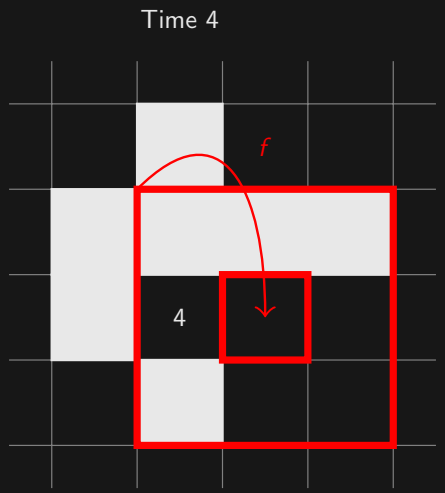
Or any permutation



Example: Life without Death.

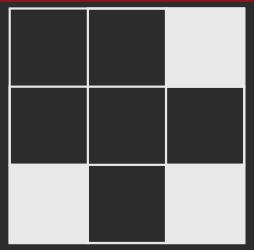
Local funcation of Life without death

f:  Or any permutation

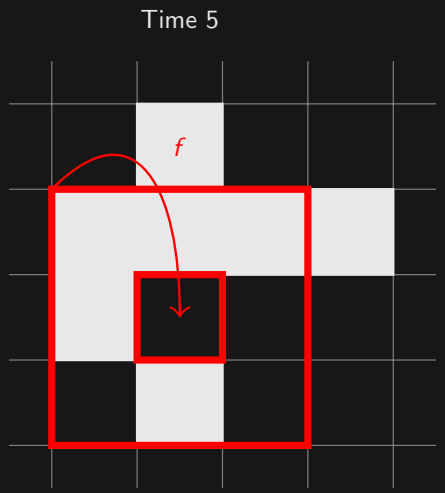


Example: Life without Death.

Local funcation of Life without death

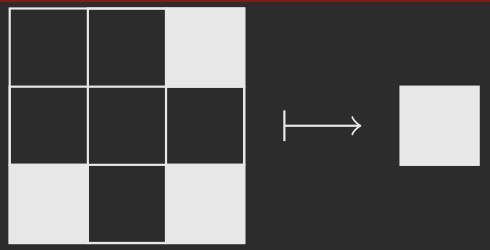
f:  $\longmapsto$ 

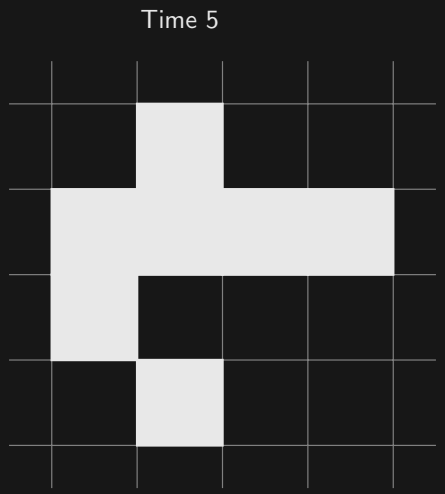
Or any permutation



Example: Life without Death.

Local funcation of Life without death

f:  $\longmapsto$ Or any permutation



Definitions

Definition

An ACA is freezing if $f : \blacksquare; * \to \blacksquare$

Definitions

Definition

An ACA is freezing if $f : \blacksquare; * \to \blacksquare$

Definition

Given a configuration x, a cell $z \in \mathbb{Z}^2$ is unstable if

$$\exists \sigma, \exists T : F^{\sigma(T)}(x)_z \neq x_z$$

Definitions

Definition

An ACA is freezing if $f : \blacksquare; * \to \blacksquare$

Definition

Given a configuration x, a cell $z \in \mathbb{Z}^2$ is unstable if

$$\exists \sigma, \exists T : F^{\sigma(T)}(x)_z \neq x_z$$

Definition ($\texttt{AsyncUnstability}_F$)

F is a FACA.

INPUT: A $n \times n$-periodic configuration x and a cell z.

QUESTION: Does there exist a updating scheme σ and $T > 0$ such that $F^{\sigma(T)}(x)_z \neq x_z$?

History

- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)

History

- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)
- Given a FCA and a $n \times n$ conf. x, $F^{O(n^2)}(x)$ is a fixed point. (Goles, Ollinger, Theyssier 2015)

History

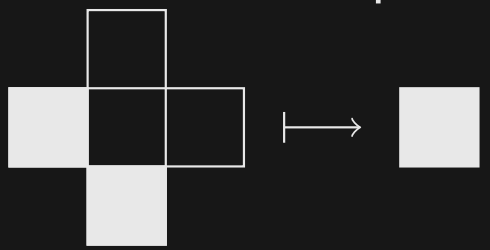
- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)
- Given a FCA and a $n \times n$ conf. x, $F^{O(n^2)}(x)$ is a fixed point. (Goles, Ollinger, Theyssier 2015)
- Unstability problem. `Input`: F, z, x, `question`: $F^{O(n^2)}(x)_z = x_z$?. (Goles, M, Montealegre, Ollinger 2017)

History

- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)
- Given a FCA and a $n \times n$ conf. x, $F^{O(n^2)}(x)$ is a fixed point. (Goles, Ollinger, Theyssier 2015)
- Unstability problem. `Input`: F, z, x, `question`: $F^{O(n^2)}(x)_z = x_z$?. (Goles, M, Montealegre, Ollinger 2017)
- We study boolean FCA with von Neumann neighborhood and we found one "complex".

History

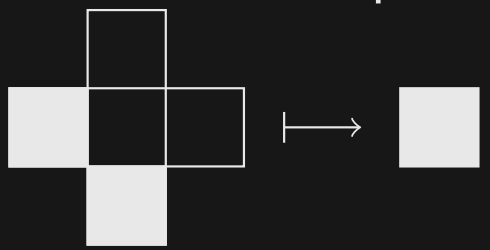
- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)
- Given a FCA and a $n \times n$ conf. x, $F^{O(n^2)}(x)$ is a fixed point. (Goles, Ollinger, Theyssier 2015)
- Unstability problem. `Input`: F, z, x, `question`: $F^{O(n^2)}(x)_z = x_z$?. (Goles, M, Montealegre, Ollinger 2017)
- We study boolean FCA with von Neumann neighborhood and we found one "complex".

f_2:  Or any permutation

- Is f_2 complex in others context too? (today, Asynchronous)

History

- Prediction problem. `Input`: F, z, x, q, t, `question`: $F^t(x)_z = q$?. (Banks 1971)
- Given a FCA and a $n \times n$ conf. x, $F^{O(n^2)}(x)$ is a fixed point. (Goles, Ollinger, Theyssier 2015)
- Unstability problem. `Input`: F, z, x, `question`: $F^{O(n^2)}(x)_z = x_z$?. (Goles, M, Montealegre, Ollinger 2017)
- We study boolean FCA with von Neumann neighborhood and we found one "complex".

f_2:  Or any permutation

- Is f_2 complex in others context too? (today, Asynchronous)

Computational Complexity

Definition

A decision problem A is NP-Complete if A is in NP and for each problem B in P B can by reduced to A, i.e. there is a ϕ polynomial s.t. $\forall x, B(x) = \mathit{true} \Leftrightarrow A(\phi(x)) = \mathit{true}$

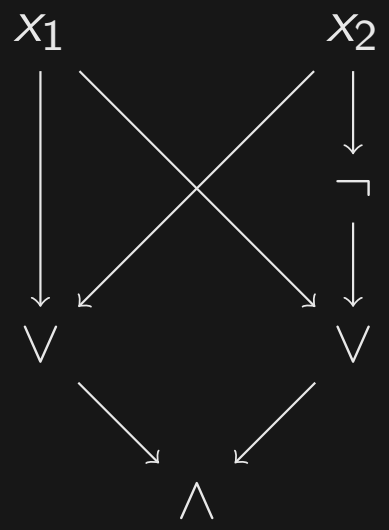
Computational Complexity

Definition

A decision problem A is NP-Complete if A is in NP and for each problem B in P B can by reduced to A, i.e. there is a ϕ polynomial s.t. $\forall x, B(x) = \mathit{true} \Leftrightarrow A(\phi(x)) = \mathit{true}$

- If A is NP-complete and P $\subsetneq$ NP, then $A \notin P$
- `Boolean satisfiability problem (SAT)` and `Circuit SAT` are NP-complete

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$$



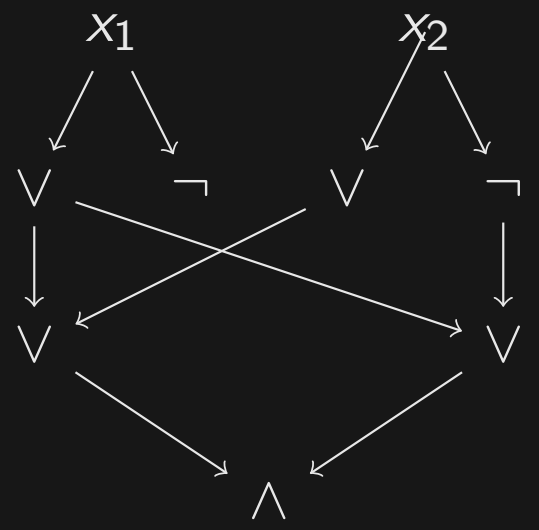
Computational Complexity

Definition

A decision problem A is NP-Complete if A is in NP and for each problem B in P B can by reduced to A, i.e. there is a ϕ polynomial s.t. $\forall x, B(x) = \mathit{true} \Leftrightarrow A(\phi(x)) = \mathit{true}$

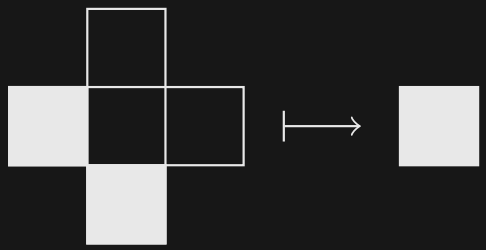
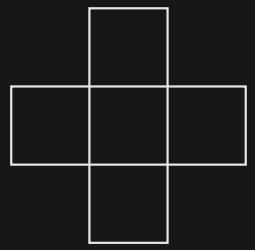
- If A is NP-complete and P $\subsetneq$ NP, then $A \notin P$
- `Boolean satisfiability problem (SAT)` and `Circuit SAT` are NP-complete

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$$



The Problem

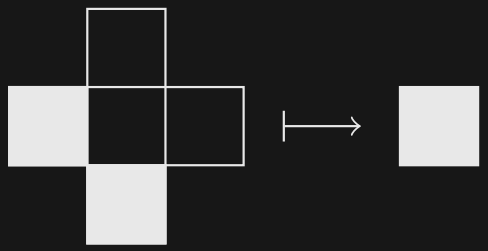
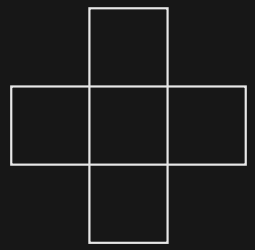
We will study the following FACA with von Neumann neighborhood:

f_2:  Or any permutation 

von Neumann neighborhood

The Problem

We will study the following FACA with von Neumann neighborhood:

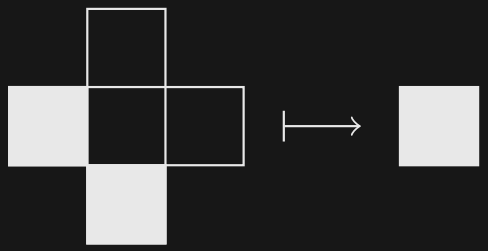
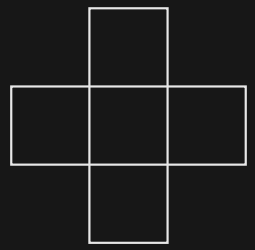
f_2:  Or any permutation 

von Neumann neighborhood

`Unstability` (synchronous version of `AsyncUnstability`) is P-complete for f_2, then the question is :

The Problem

We will study the following FACA with von Neumann neighborhood:

f_2:  Or any permutation 

von Neumann neighborhood

`Unstability` (synchronous version of `AsyncUnstability`) is P-complete for f_2, then the question is :

Problem

Is `AsyncUnstability` NP-complete for f_2?

Previous Works

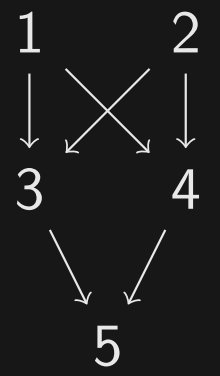
Theorem (Goldschlager 1977)

Planar circuit value problem is P-complete.

Previous Works

Theorem (Goldschlager 1977)

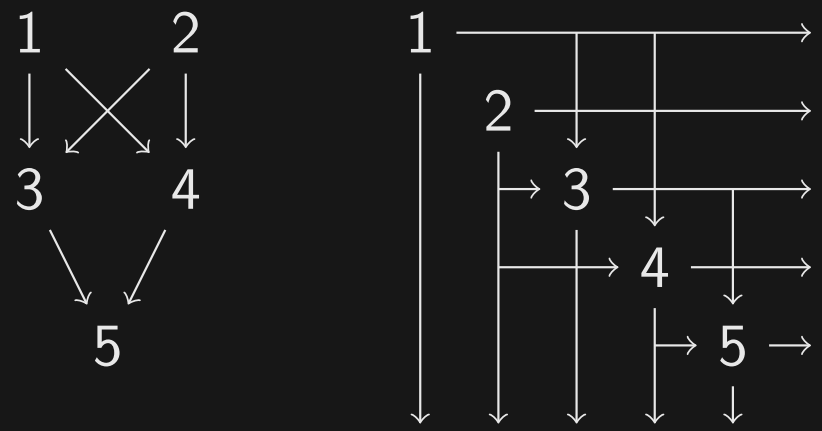
Planar circuit value problem is P-complete.



Previous Works

Theorem (Goldschlager 1977)

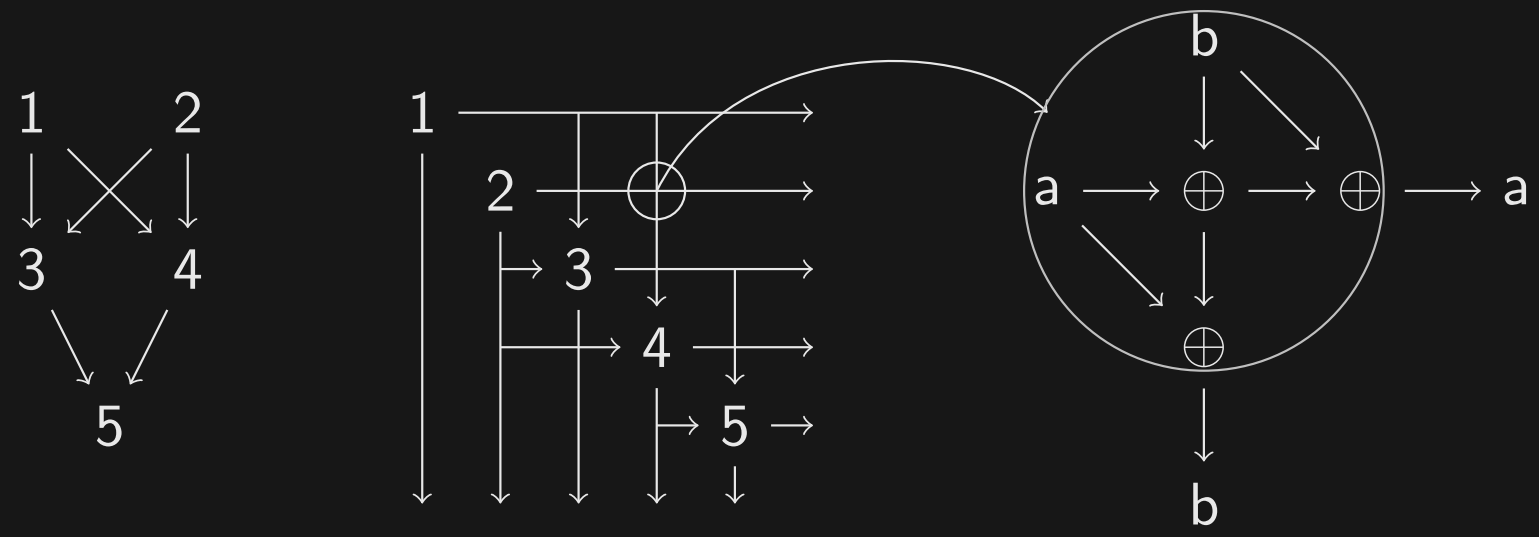
Planar circuit value problem is P-complete.



Previous Works

Theorem (Goldschlager 1977)

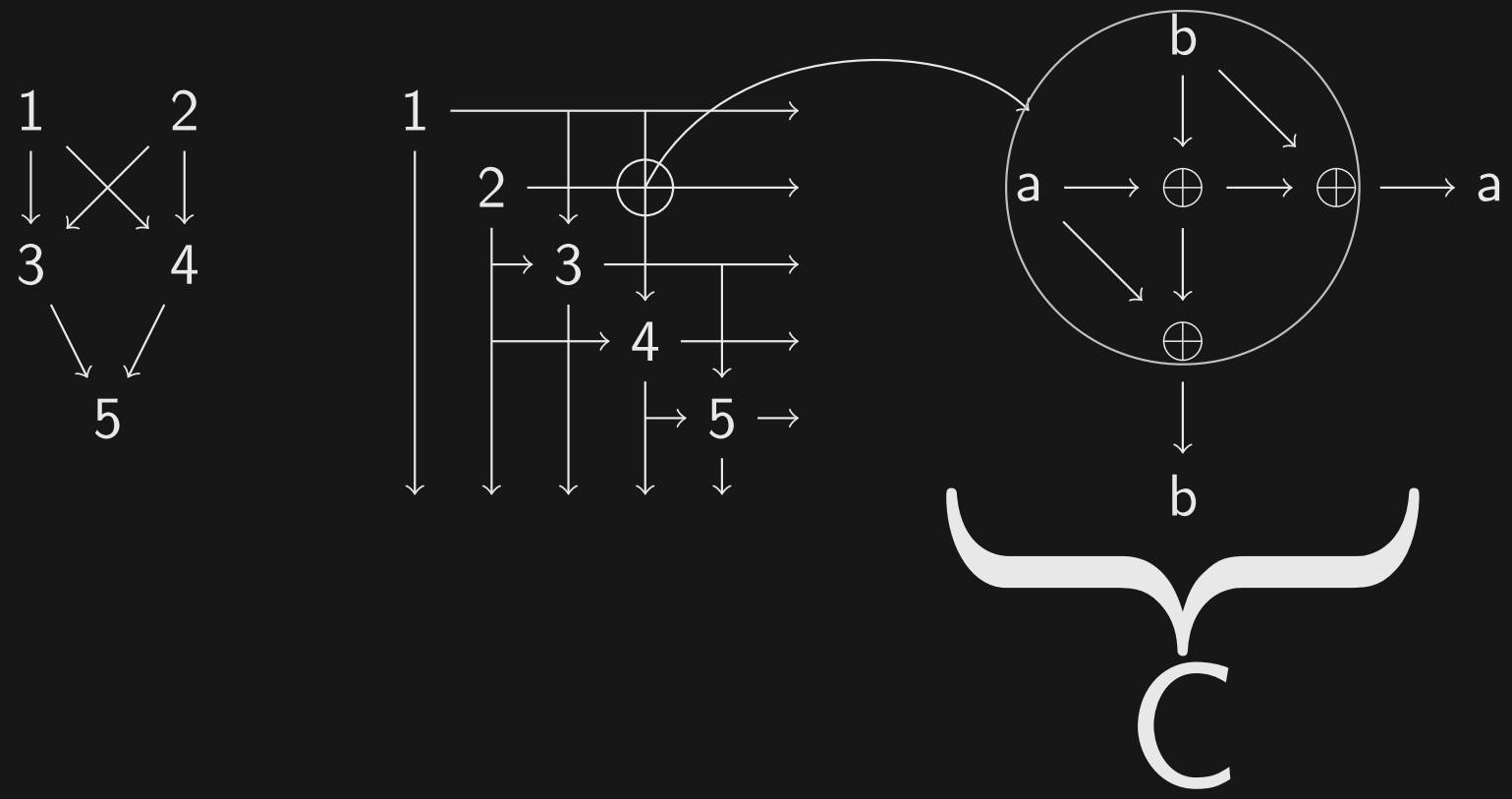
Planar circuit value problem is P-complete.



Previous Works

Theorem (Goldschlager 1977)

Planar circuit value problem is P-complete.

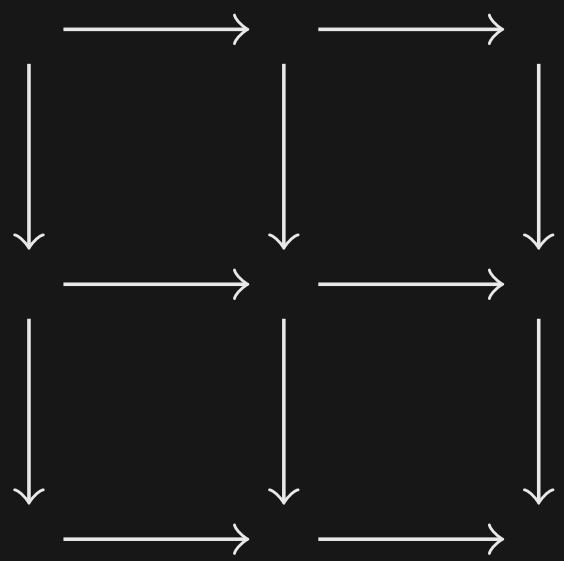


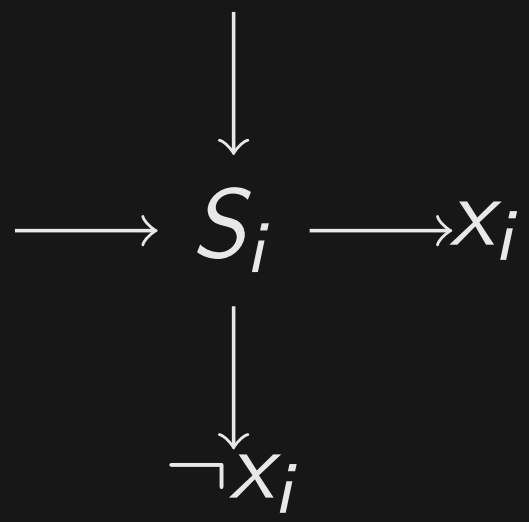
Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

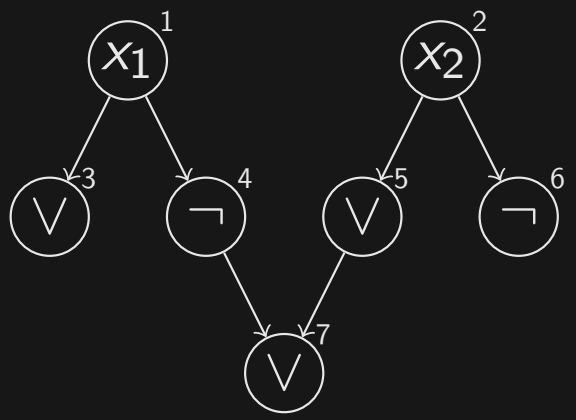




Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$

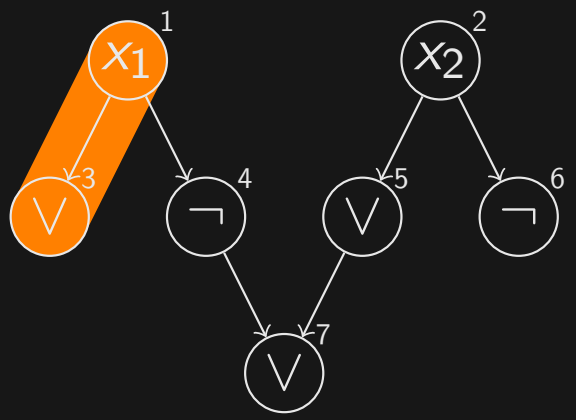


	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$

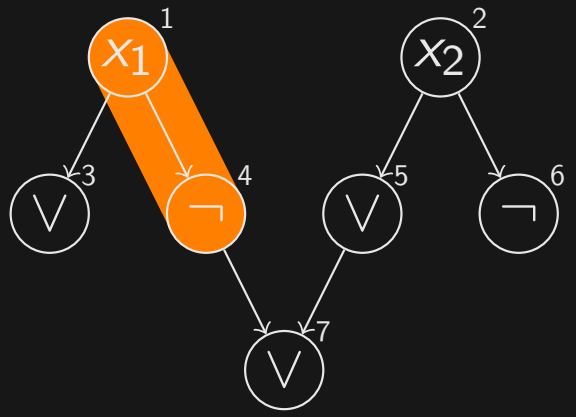


	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$

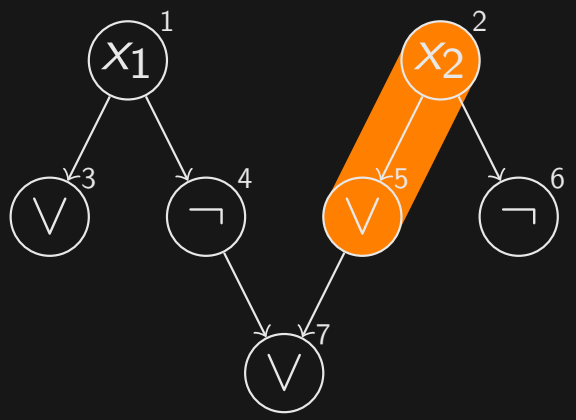


	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$

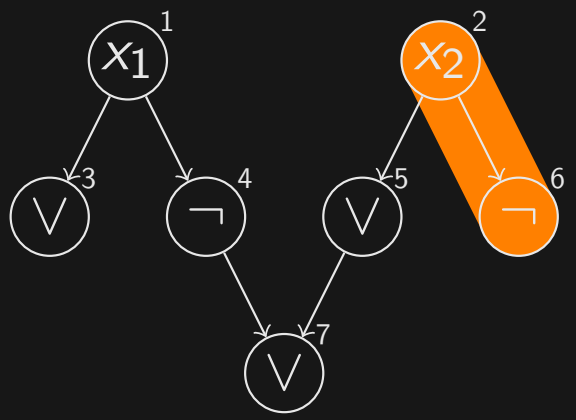


	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$

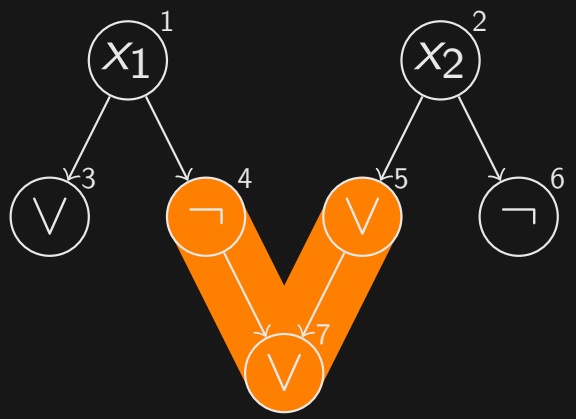


	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

Lemma I

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0, C$ and S_i, where S_i sends a value for the South output and the opposite value for the East output and C is a crossing gate.

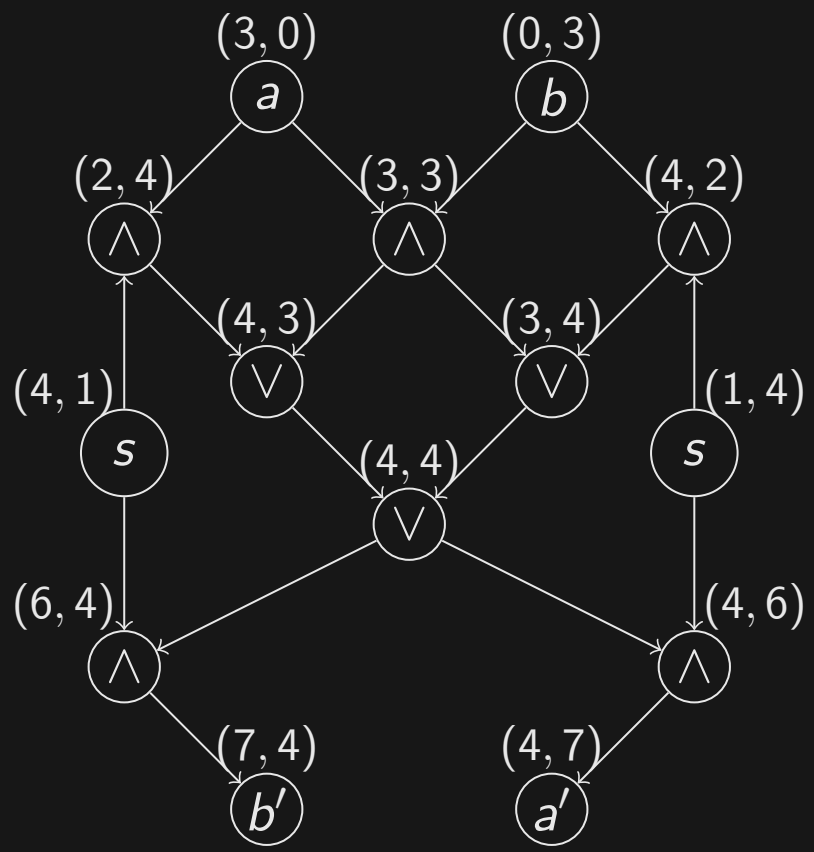
$$\phi(x_1, x_2) = \neg x_1 \vee x_2$$



	1	2	3	4	5	6	7
1	S_1	C	$\top$	C	C	C	C
2	C	S_2	C	C	$\top$	C	C
3	C	C	$\vee$	C	C	C	C
4	$\vdash$	C	C	$\vee$	C	C	C
5	C	C	C	C	$\vee$	C	$\top$
6	C	$\vdash$	C	C	C	$\vee$	C
7	C	C	C	$\vdash$	C	C	$\vee$

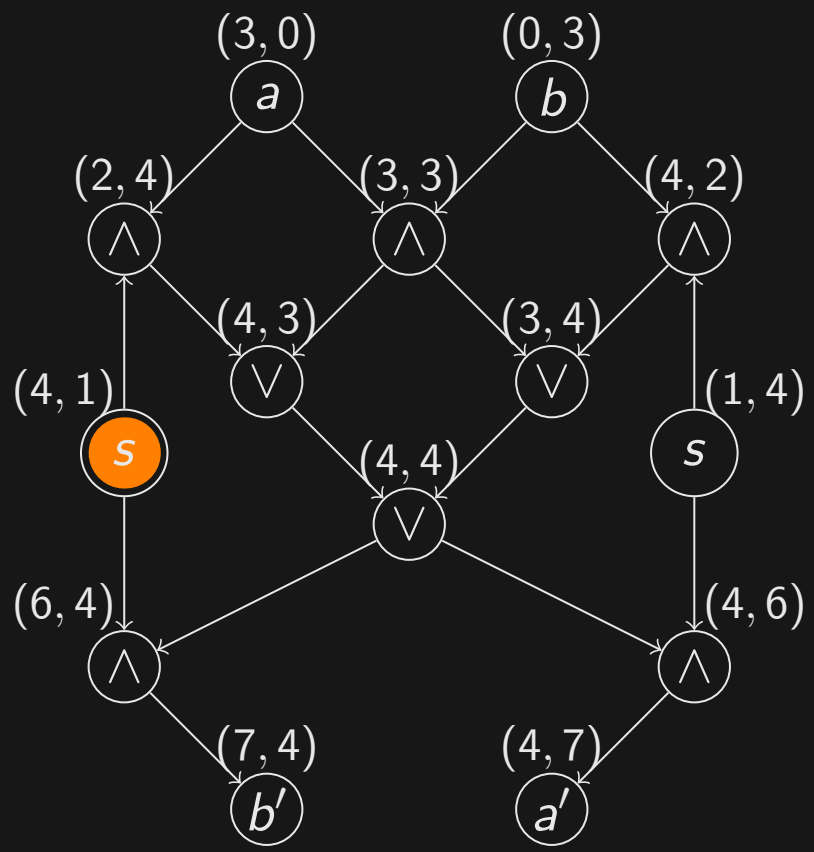
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



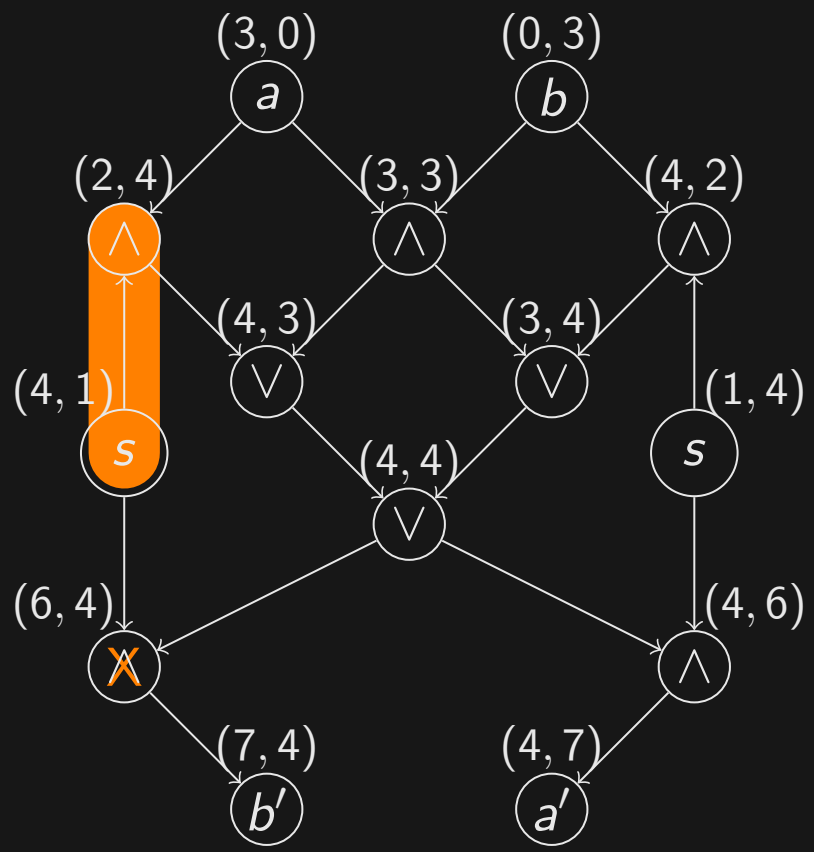
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



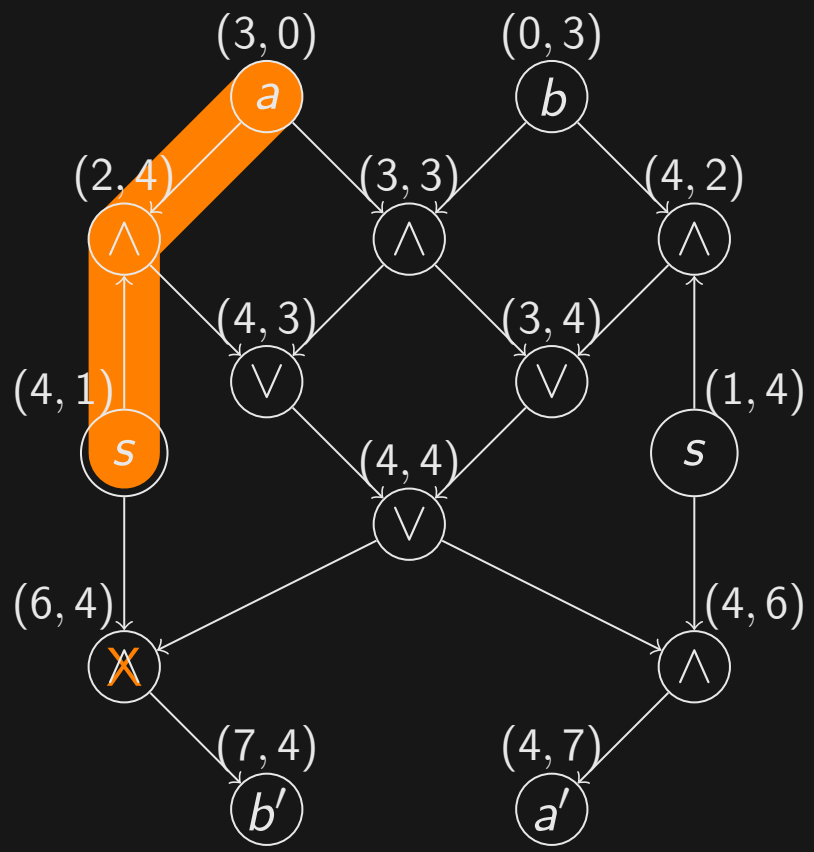
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



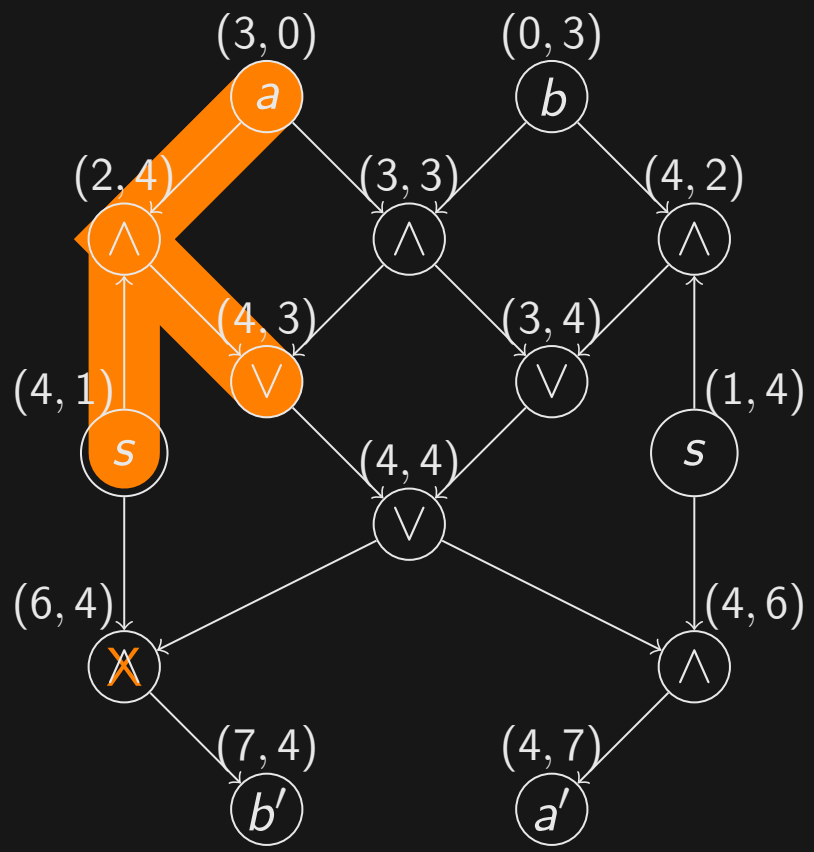
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



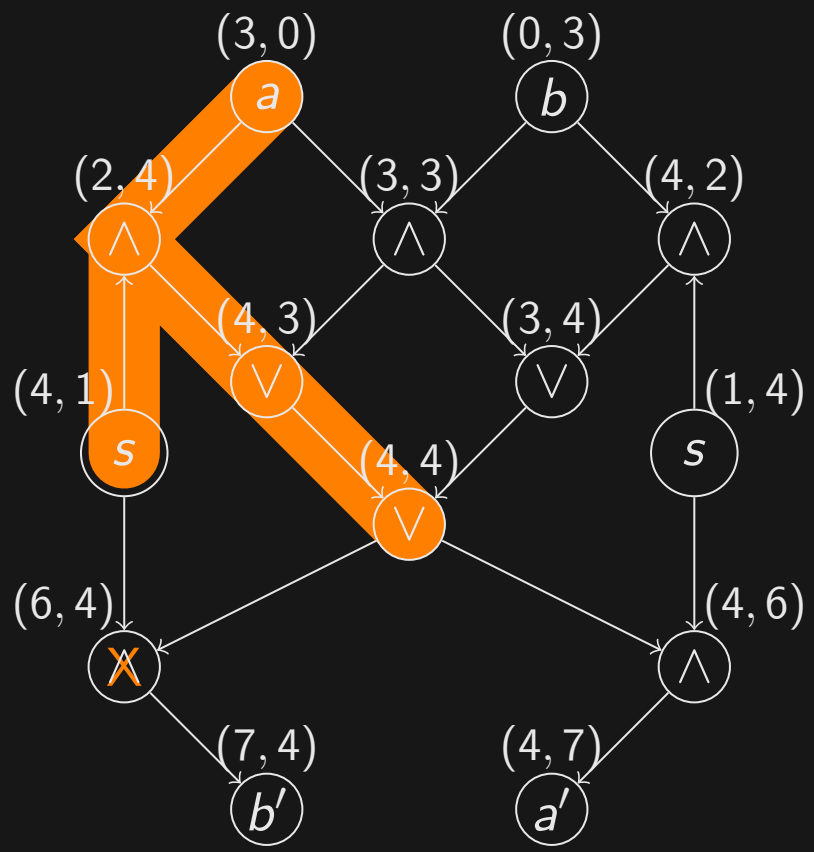
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



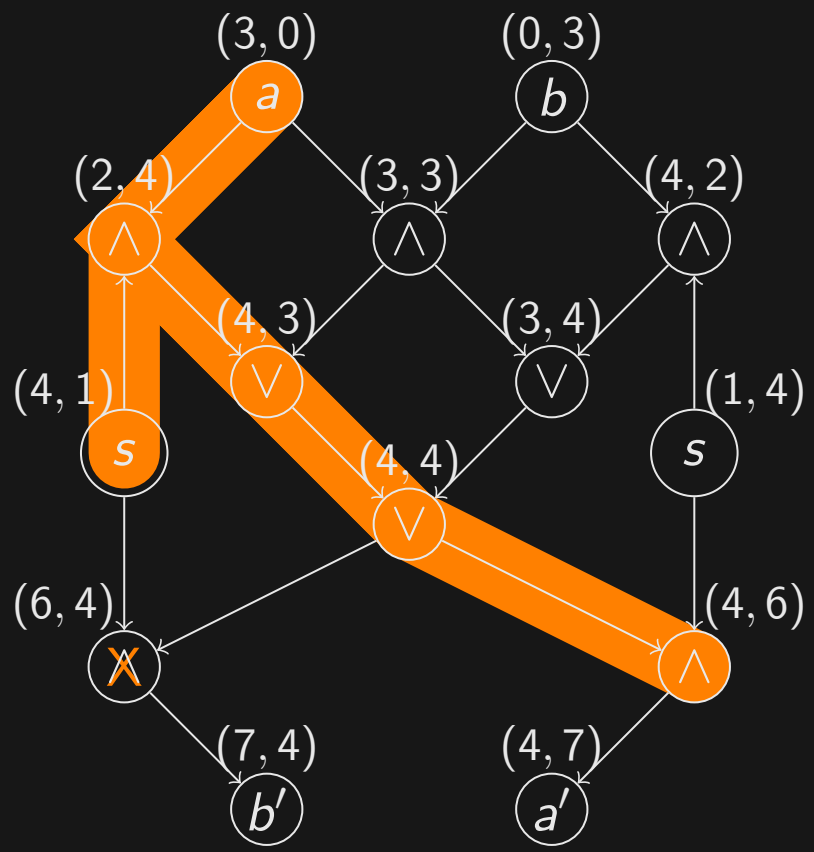
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



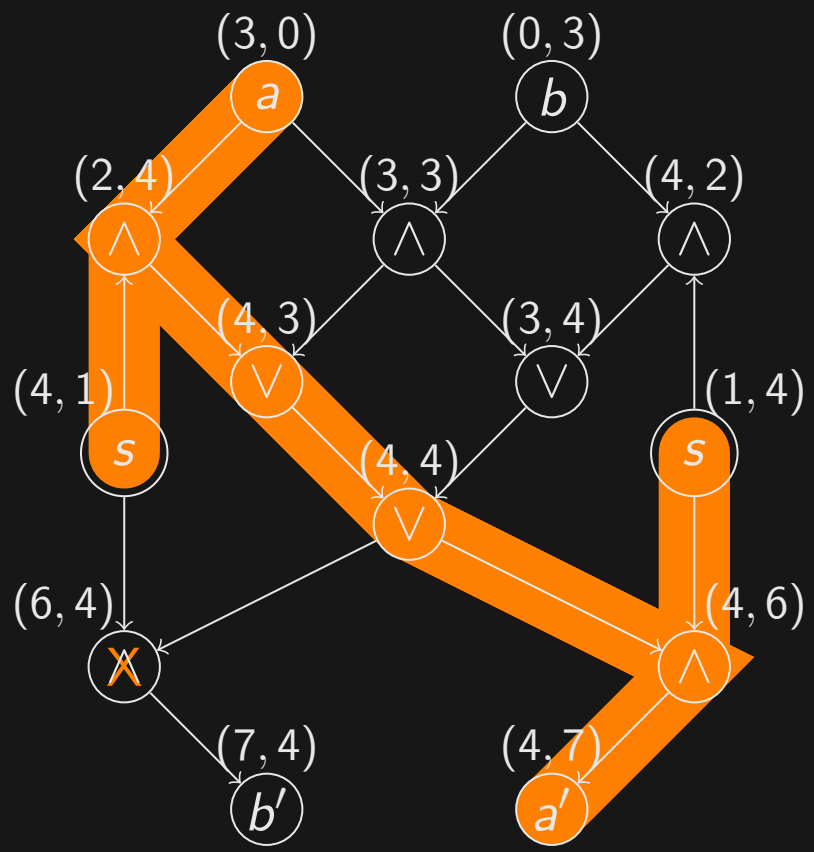
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



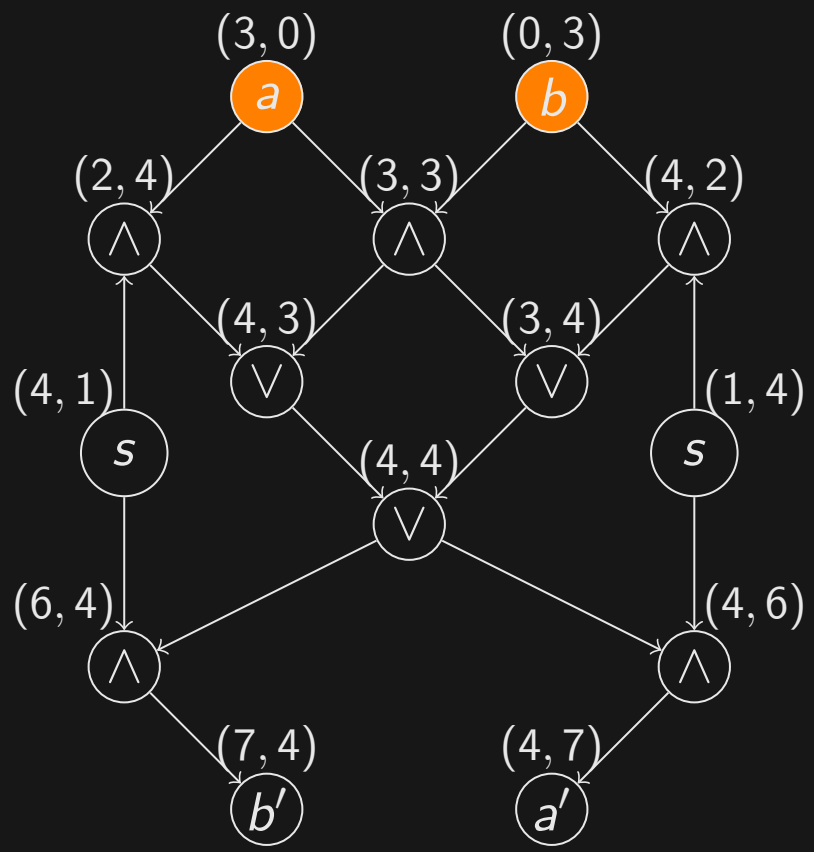
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



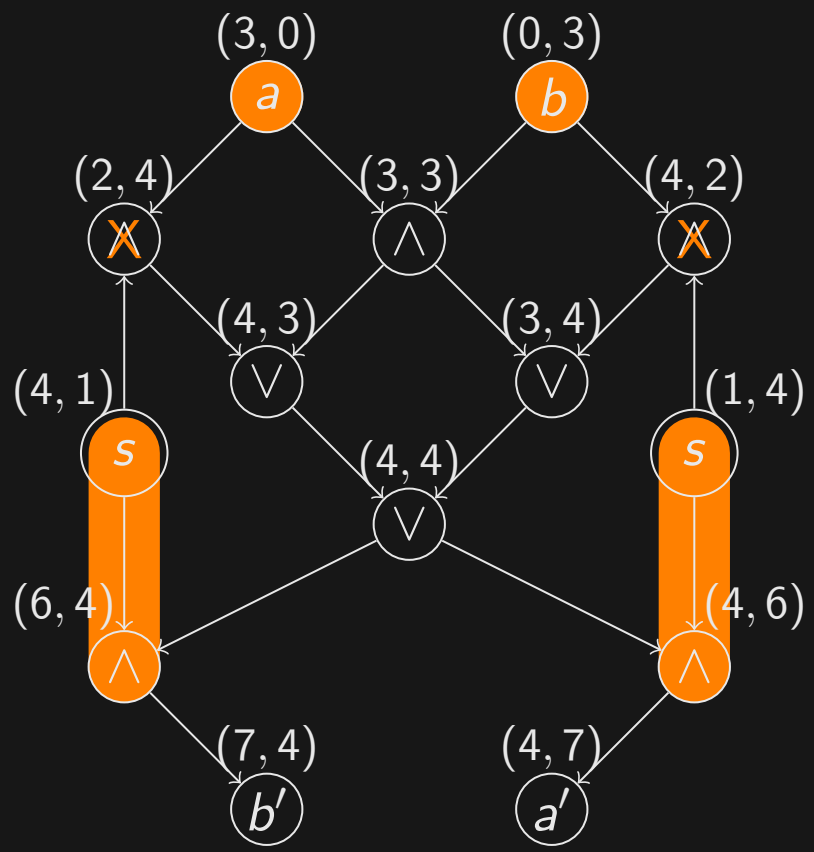
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



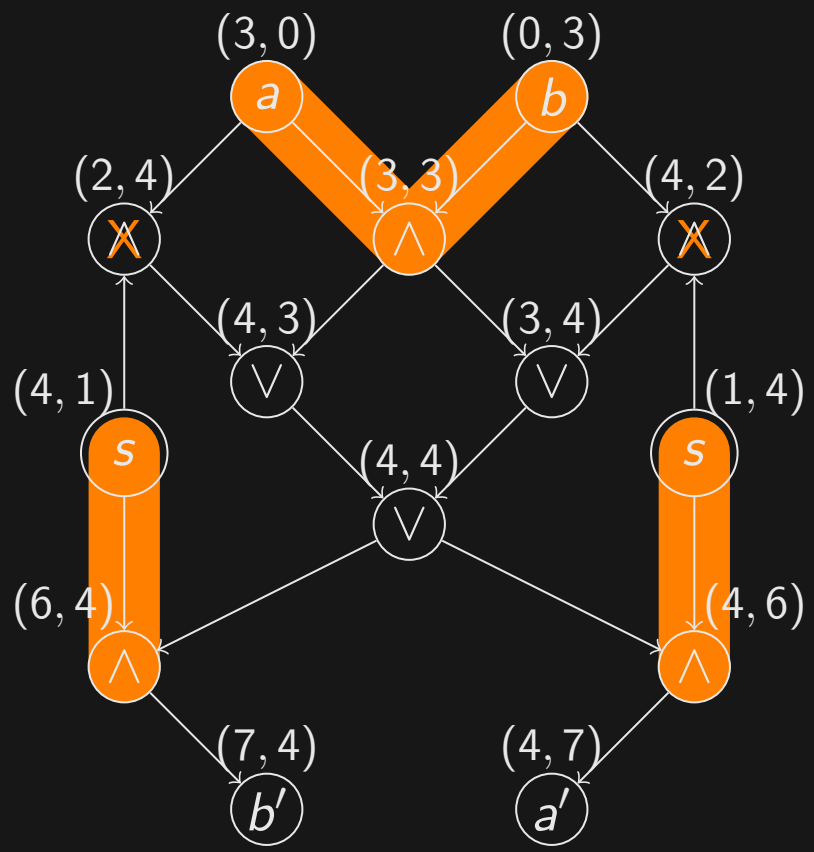
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



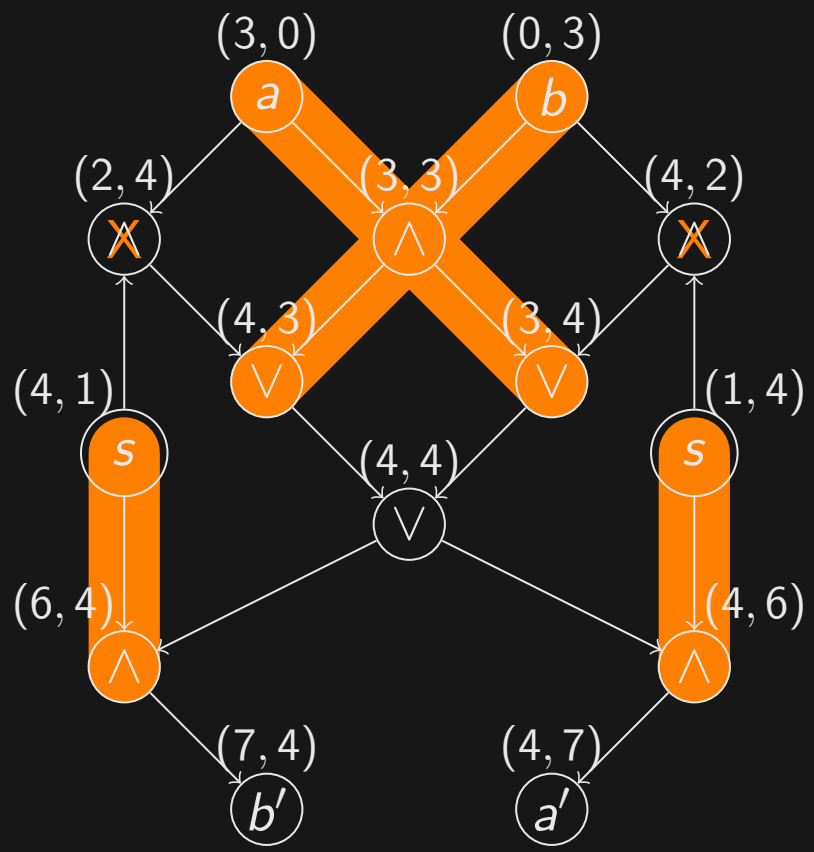
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



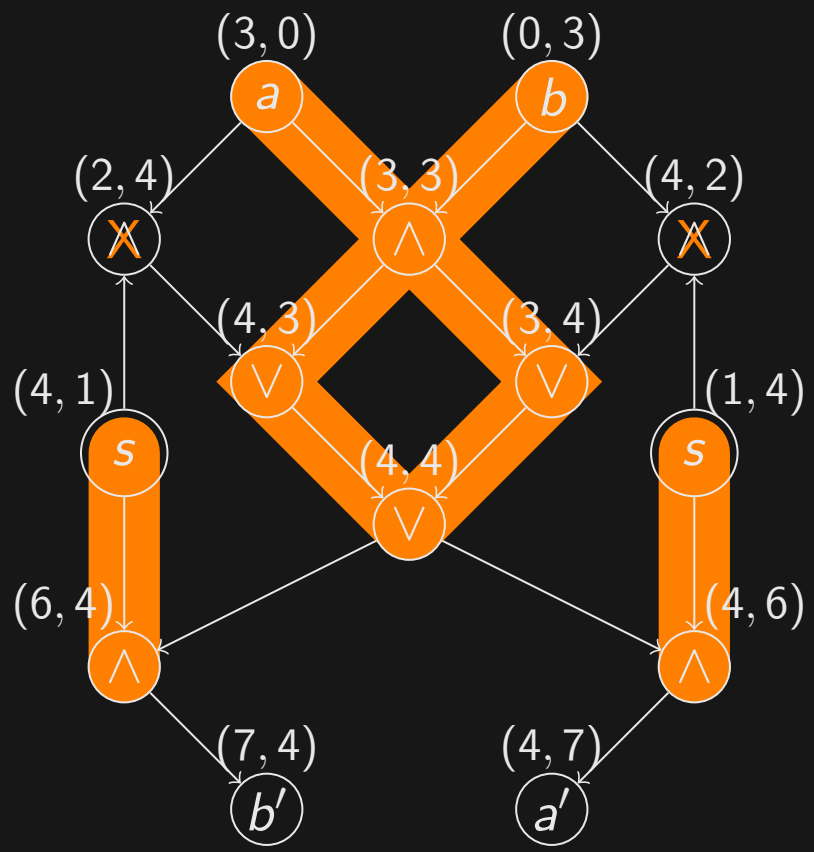
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



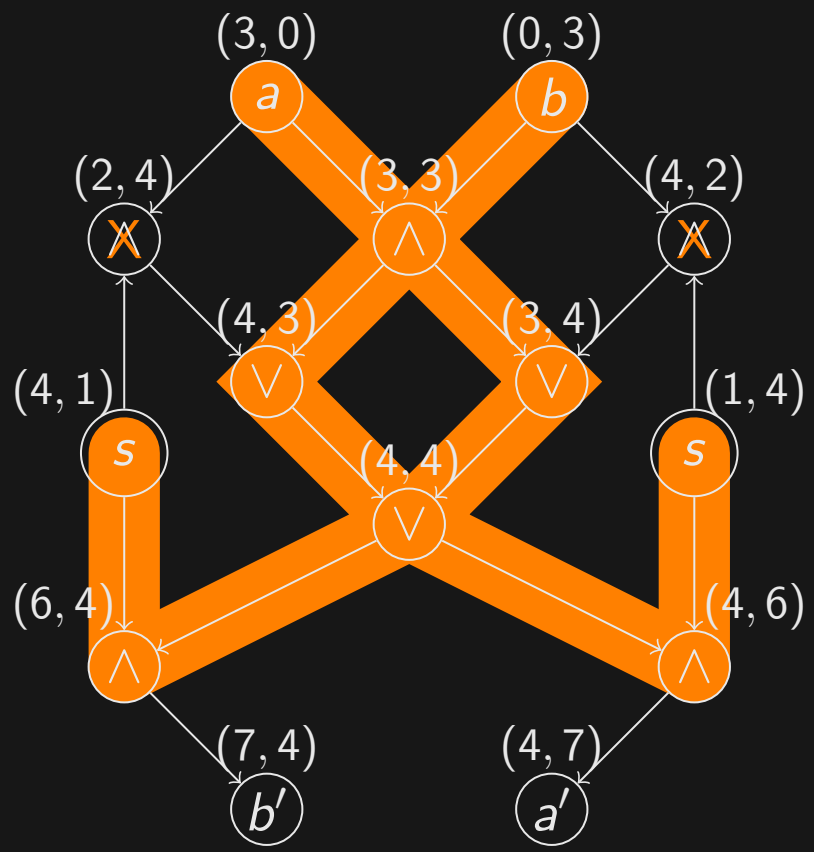
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



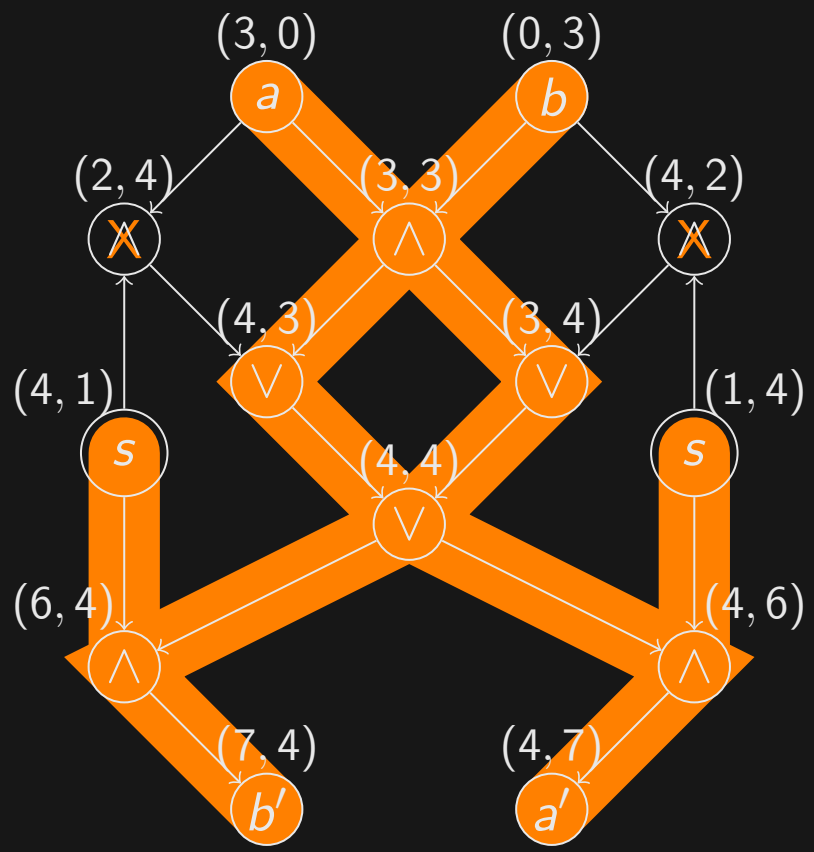
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



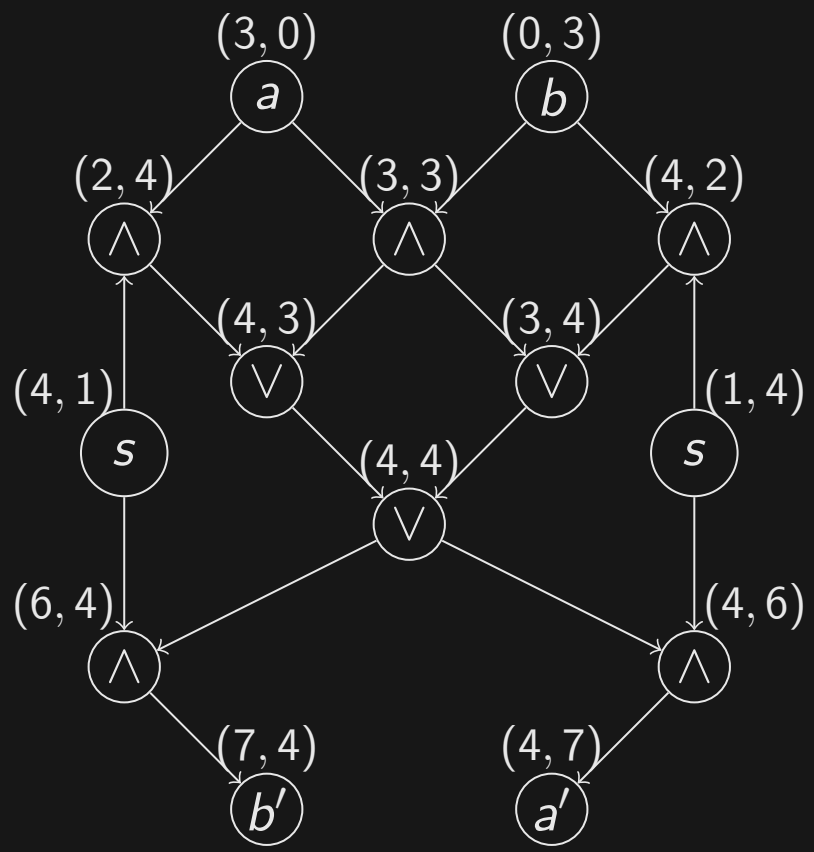
Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



Lemma II

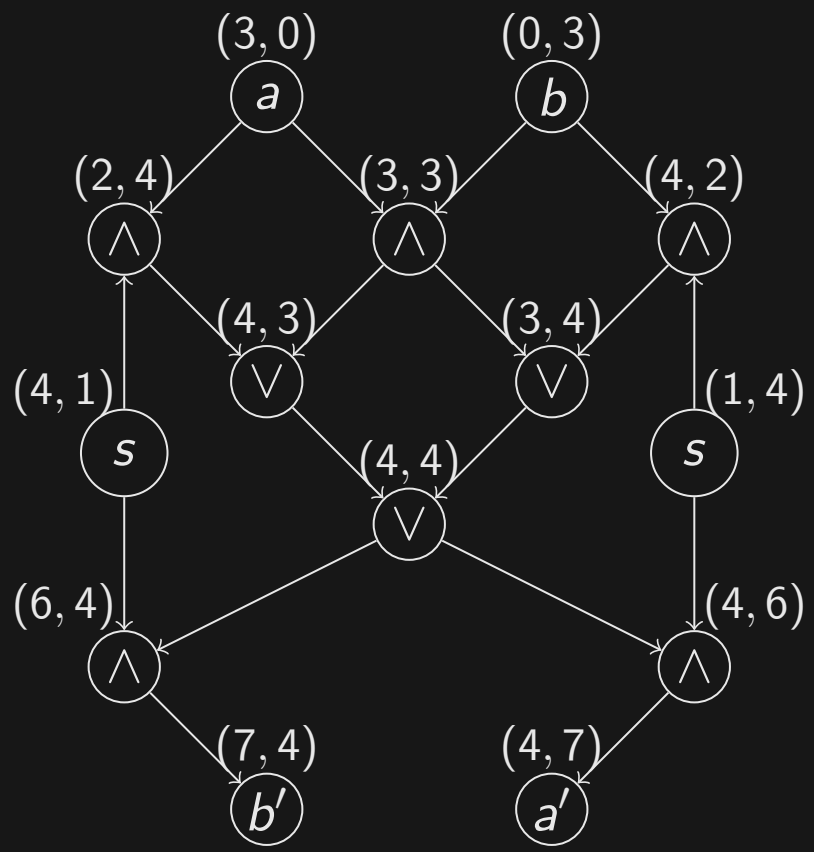
South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



a	b	s_1	s_2	a'	b'
0	0	↓	↓	0	0
0	0	↓	→	0	0
0	0	→	↓	0	0
0	0	→	→	0	0
0	1	↓	↓	0	1
0	1	↓	→	0	0
0	1	→	↓	0	0
0	1	→	→	0	0
1	0	↓	↓	0	0
1	0	↓	→	0	0
1	0	→	↓	0	0
1	0	→	→	1	0
1	1	↓	↓	0	1
1	1	↓	→	1	1
1	1	→	↓	0	0
1	1	→	→	1	0

Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



	0	1	2	3	4	5	6	7
0	0	0	0	$\vee$	0	0	0	0
1	0	0	0	$\vee$	S	$\vee$	$\vee$	0
2	0	0	0	$\vee$	$\wedge$	0	$\vee$	0
3	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$	0	$\vee$	0
4	0	S	$\wedge$	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$
5	0	$\vee$	0	0	$\vee$	0	0	0
6	0	$\vee$	$\vee$	$\vee$	$\wedge$	0	0	0
7	0	0	0	0	$\vee$	0	0	0

Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.

	0	1	2	3	4	5	6	7
0	0	0	0	$\vee$	0	0	0	0
1	0	0	0	$\vee$	S	$\vee$	$\vee$	0
2	0	0	0	$\vee$	$\wedge$	0	$\vee$	0
3	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$	0	$\vee$	0
4	0	S	$\wedge$	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$
5	0	$\vee$	0	0	$\vee$	0	0	0
6	0	$\vee$	$\vee$	$\vee$	$\wedge$	0	0	0
7	0	0	0	0	$\vee$	0	0	0

Problem!!!

Lemma II

South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.

	0	1	2	3	4	5	6	7
0	0	0	0	$\vee$	0	0	0	0
1	0	0	0	$\vee$	S	$\vee$	$\vee$	0
2	0	0	0	$\vee$	$\wedge$	0	$\vee$	0
3	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$	0	$\vee$	0
4	0	S	$\wedge$	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$
5	0	$\vee$	0	0	$\vee$	0	0	0
6	0	$\vee$	$\vee$	$\vee$	$\wedge$	0	0	0
7	0	0	0	0	$\vee$	0	0	0

	0	1	2	3	4	5	6	7
0	0	0	0	$\vee$	0	0	0	0
1	0	0	0	$\vee$	S	$\vee$	$\vee$	0
2	0	0	0	$\vee$	$\wedge$	0	$\vee$	0
3	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$	0	$\vee$	0
4	0	S	$\wedge$	$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$
5	0	$\vee$	0	0	$\vee$	0	0	0
6	0	$\vee$	$\vee$	$\vee$	$\wedge$	0	0	0
7	0	0	0	0	$\vee$	0	0	0

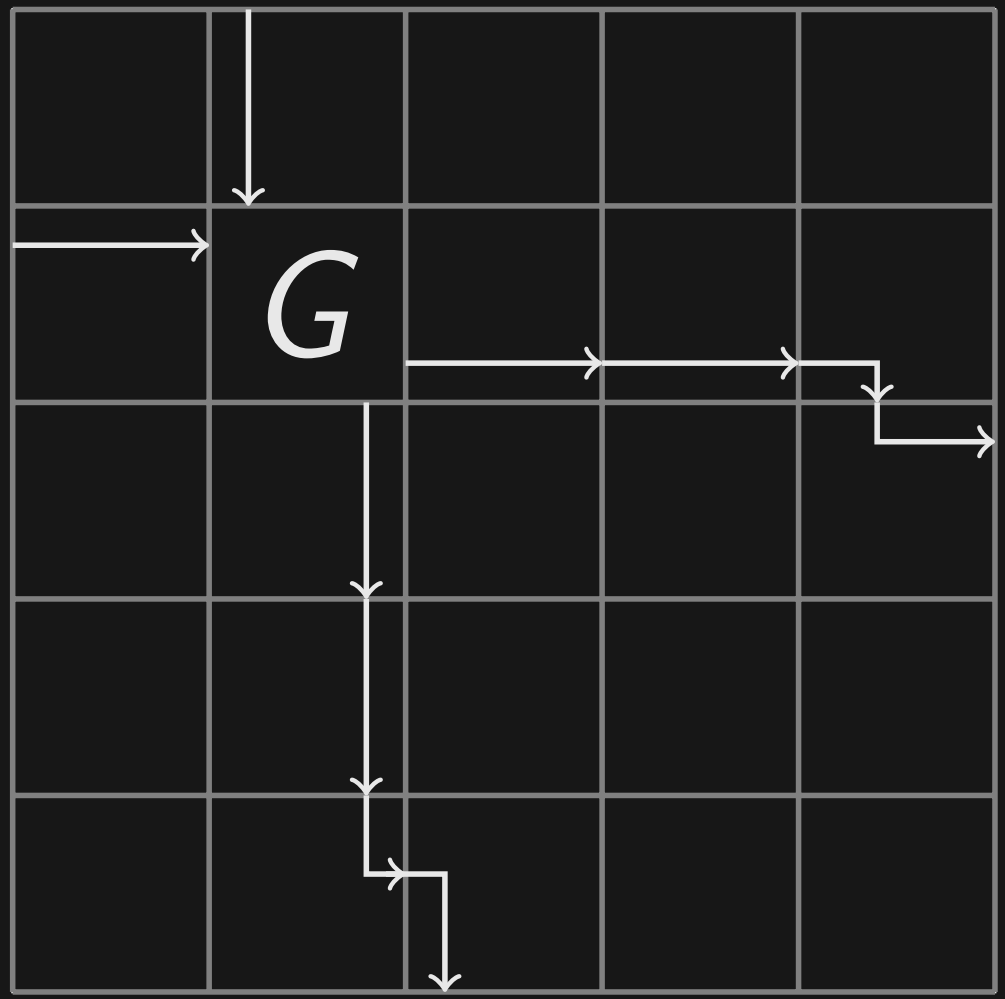
Gates $\vee, \wedge, 0$ and S

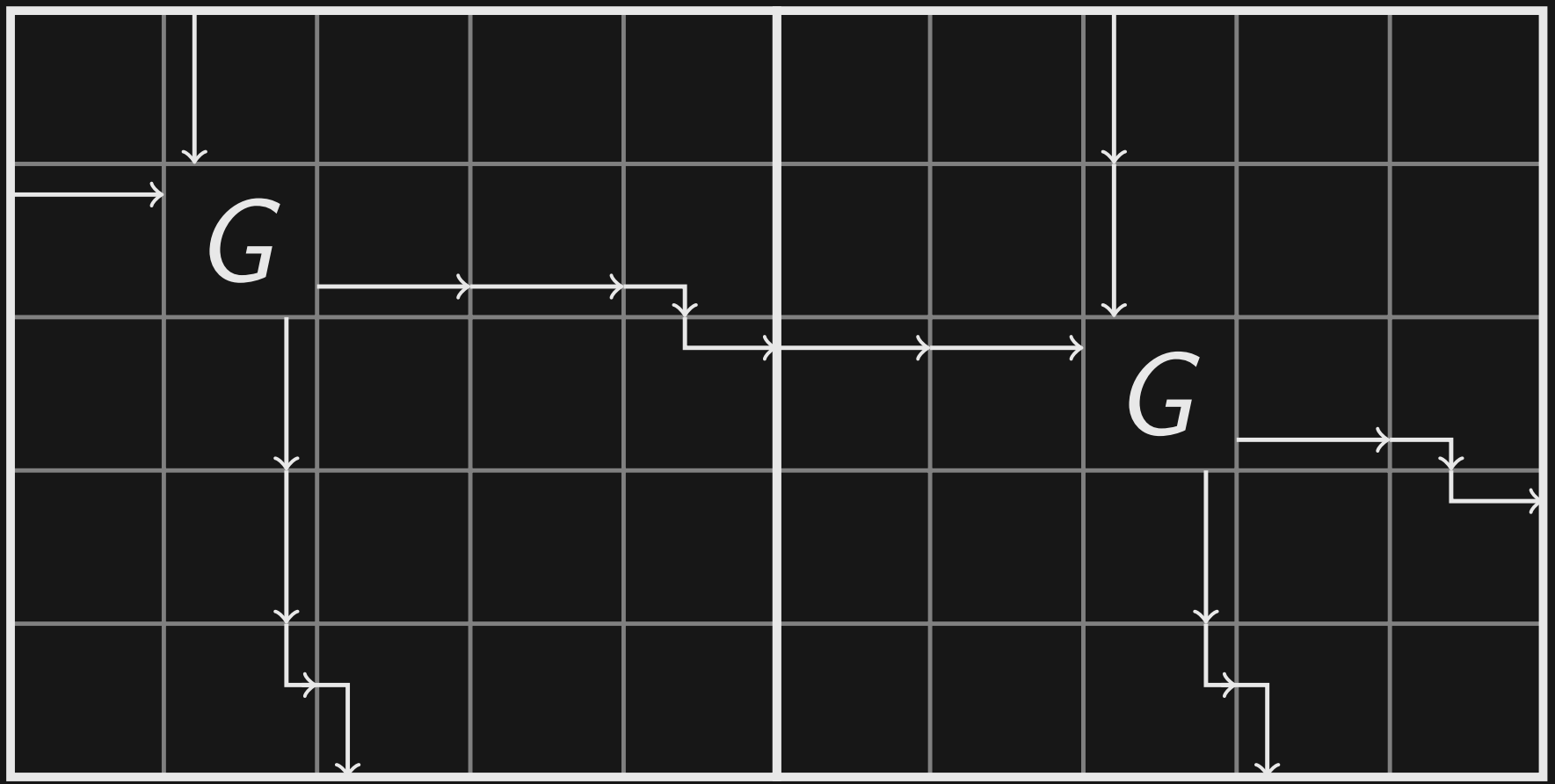
0	0	0	$\vee$	0	0	0	0
0	0	0	$\vee$	0	0	0	0
0	0	0	$\vee$	0	0	0	0
$\vee$	$\vee$	$\vee$	$\vee$	$\vee$	0	0	0
0	0	0	0	$\vee$	$\vee$	$\vee$	$\vee$
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0

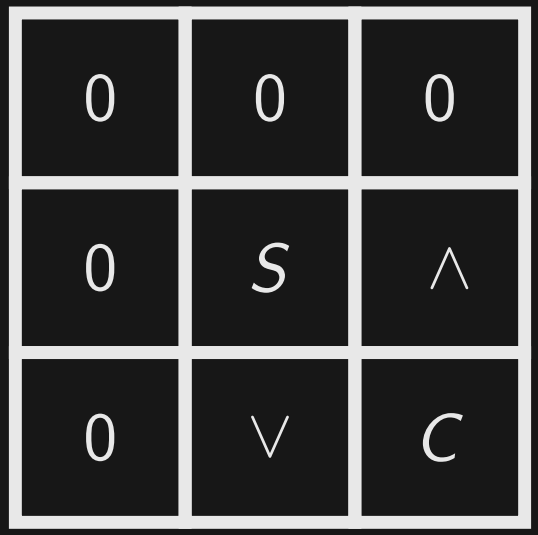
0	0	0	$\vee$	0	0	0	0
0	0	0	$\vee$	0	0	0	0
0	0	0	$\vee$	0	0	0	0
$\vee$	$\vee$	$\vee$	$\wedge$	$\vee$	0	0	0
0	0	0	0	$\vee$	$\vee$	$\vee$	$\vee$
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0

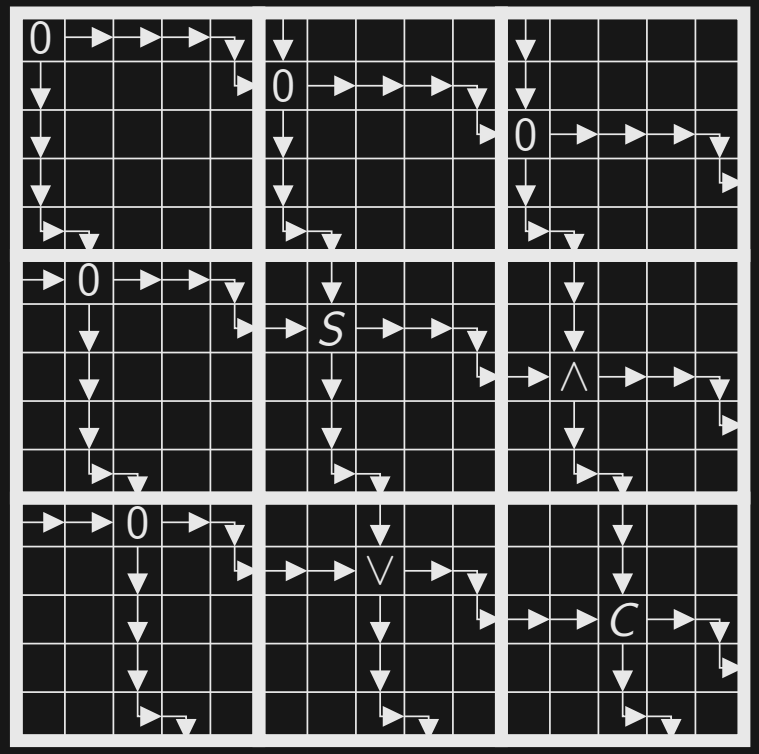
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	S	$\vee$	$\vee$	$\vee$
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	$\vee$	$\vee$	$\vee$	$\vee$
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0
0	0	0	0	$\vee$	0	0	0


G


G
G





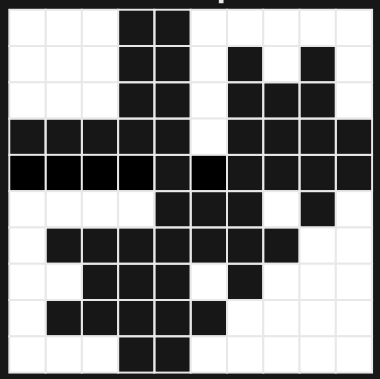
Therefore South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.

Finally...

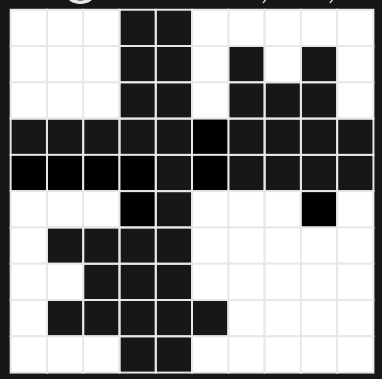
Theorem

AsyncUnstability is NP-complete.

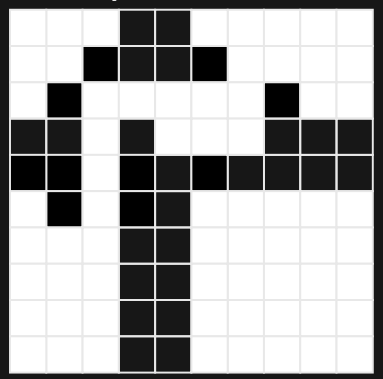
Remember: South-East grid-embedded circuit SAT is NP-complete, with gates $\wedge, \vee, 0$ and S_i.



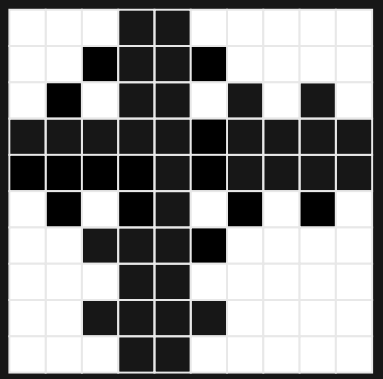
$\vee$



$\wedge$

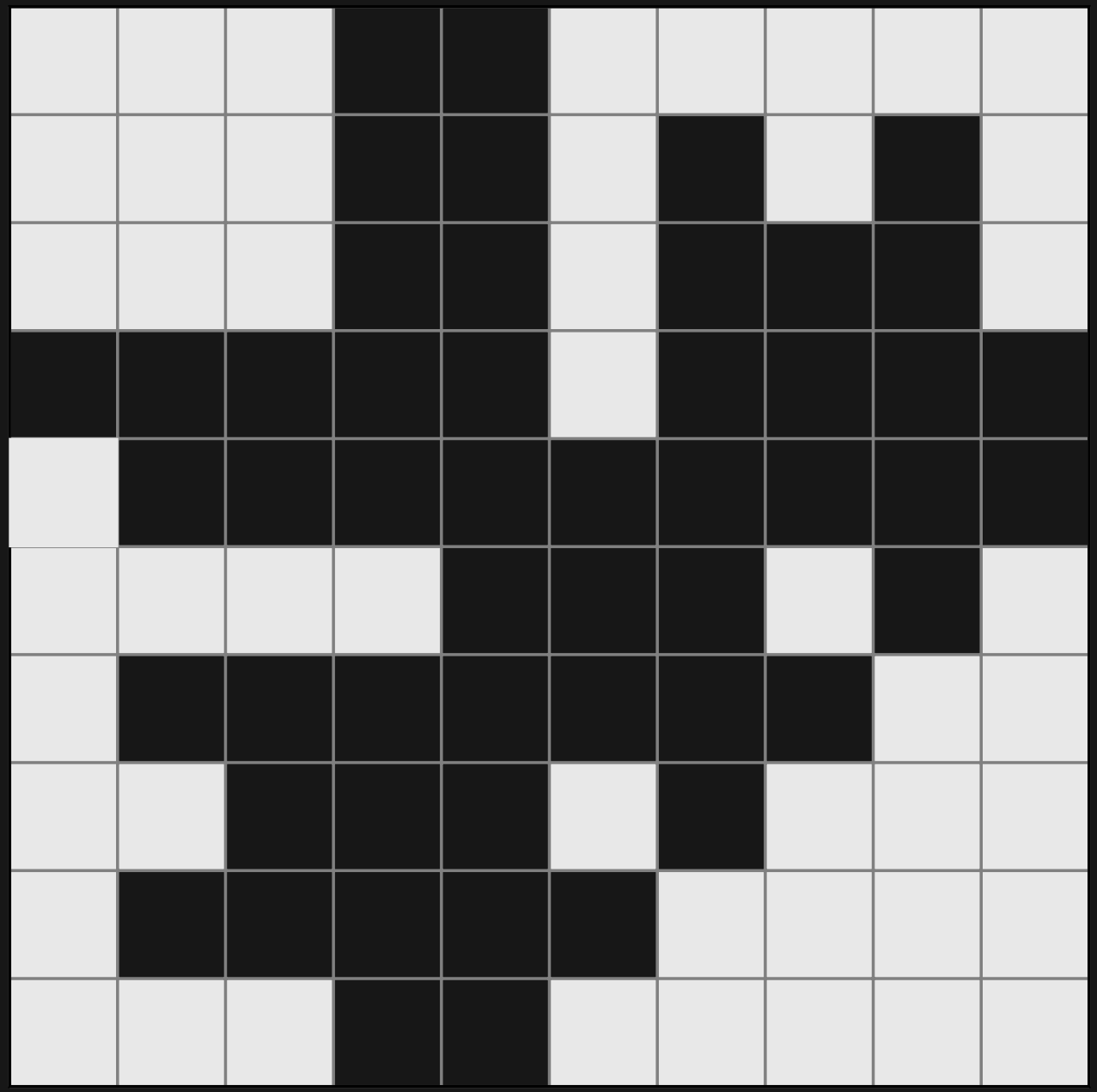


S

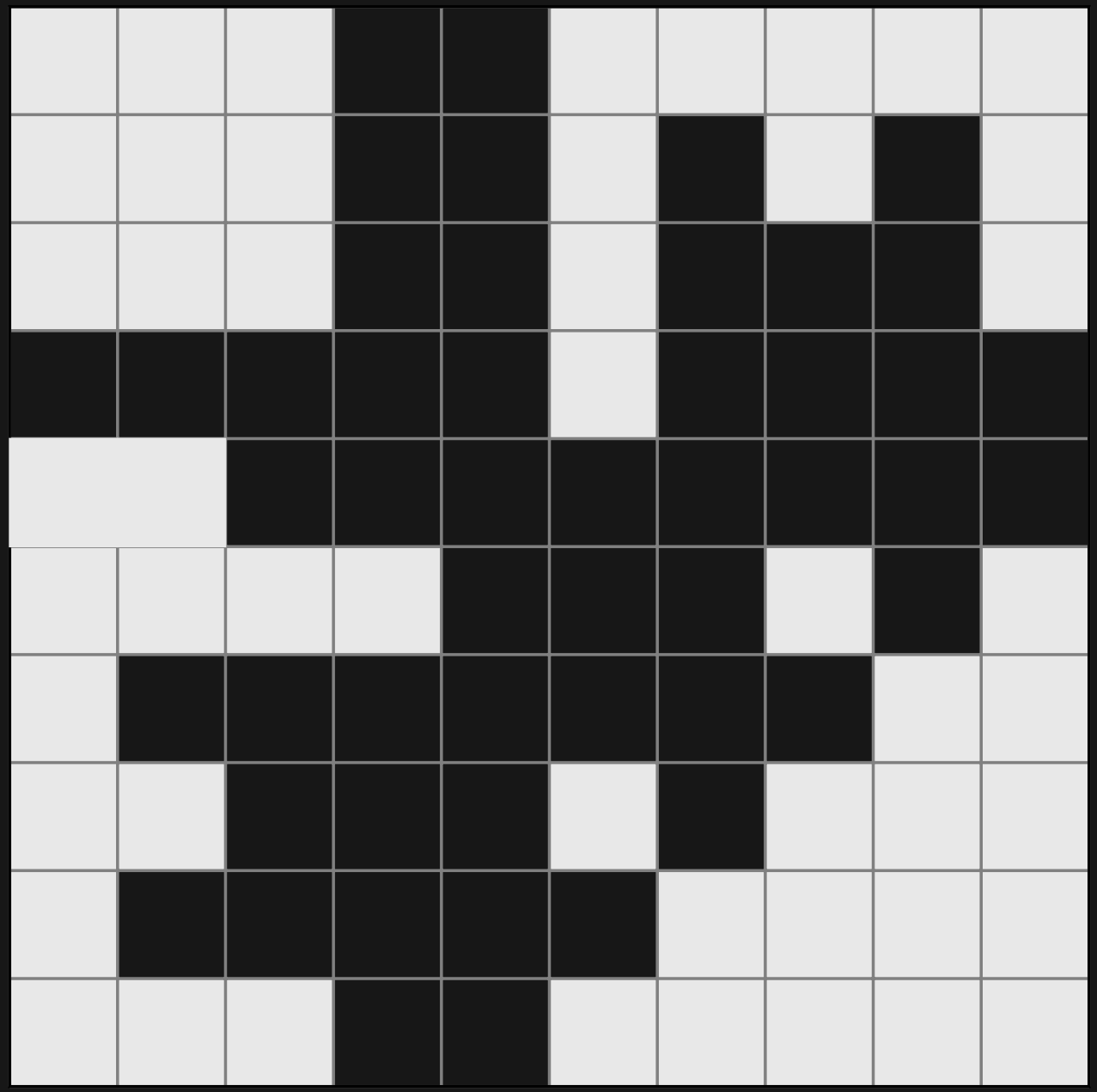


0

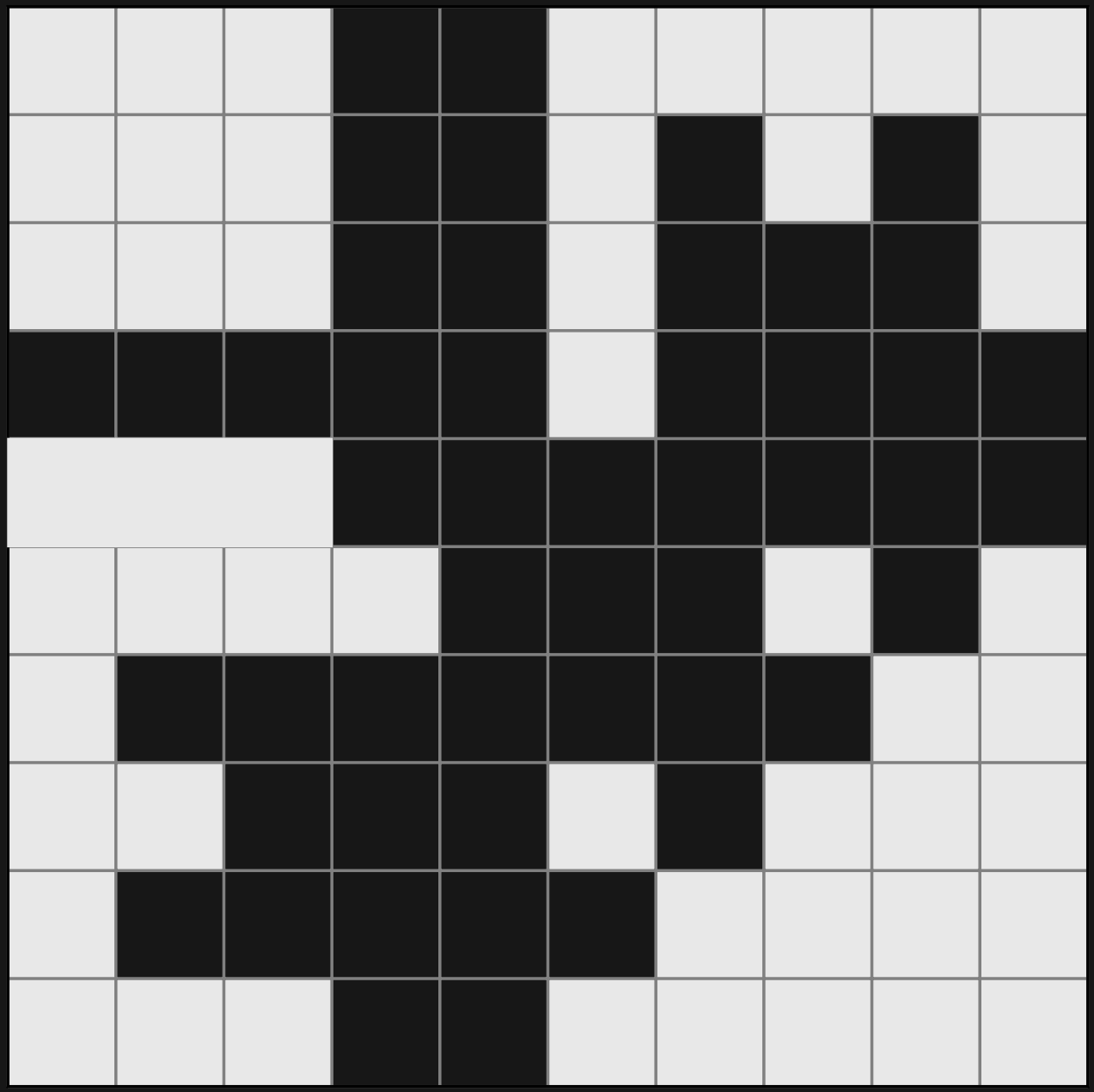
OR gate in f_2



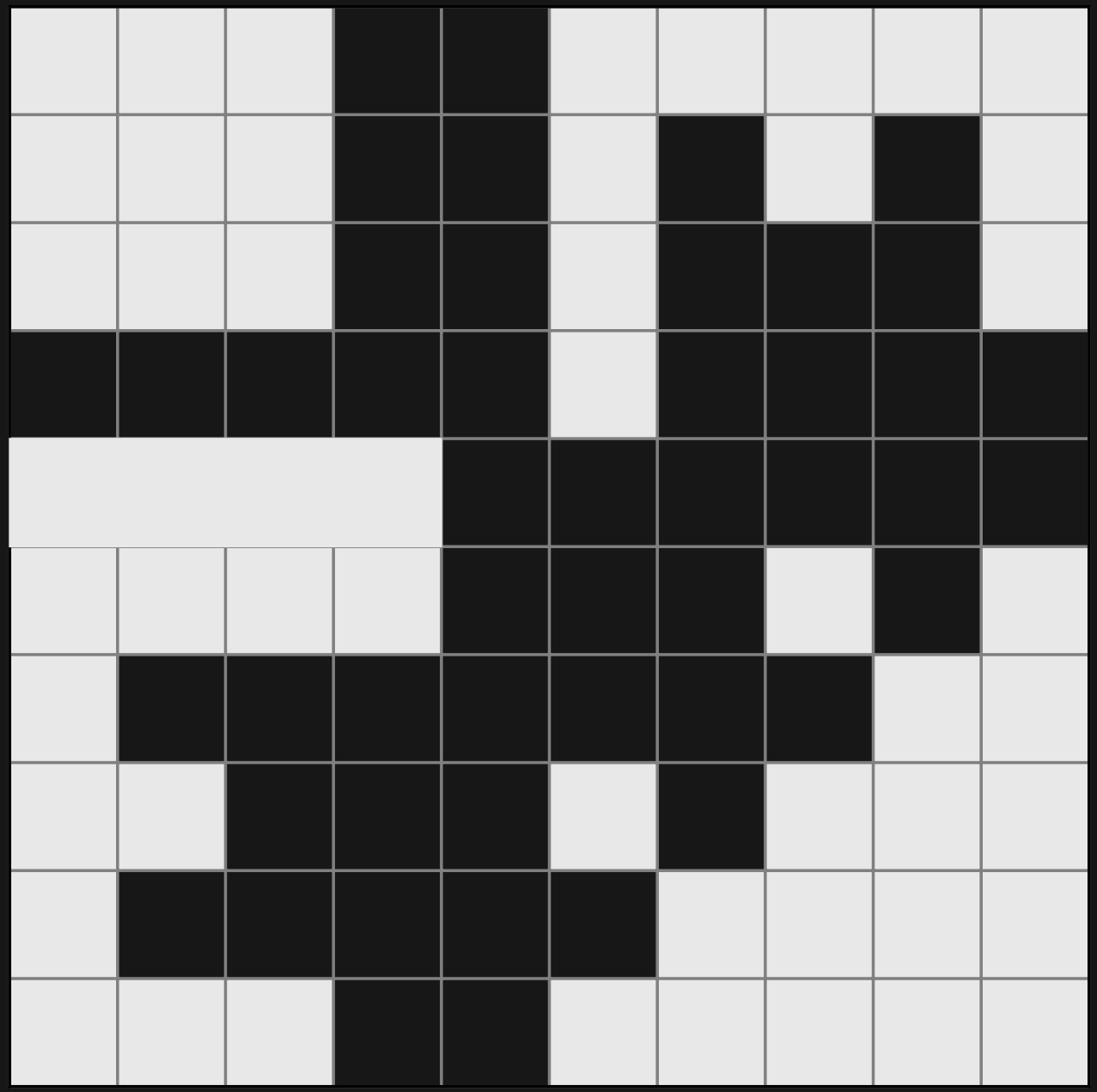
OR gate in f_2



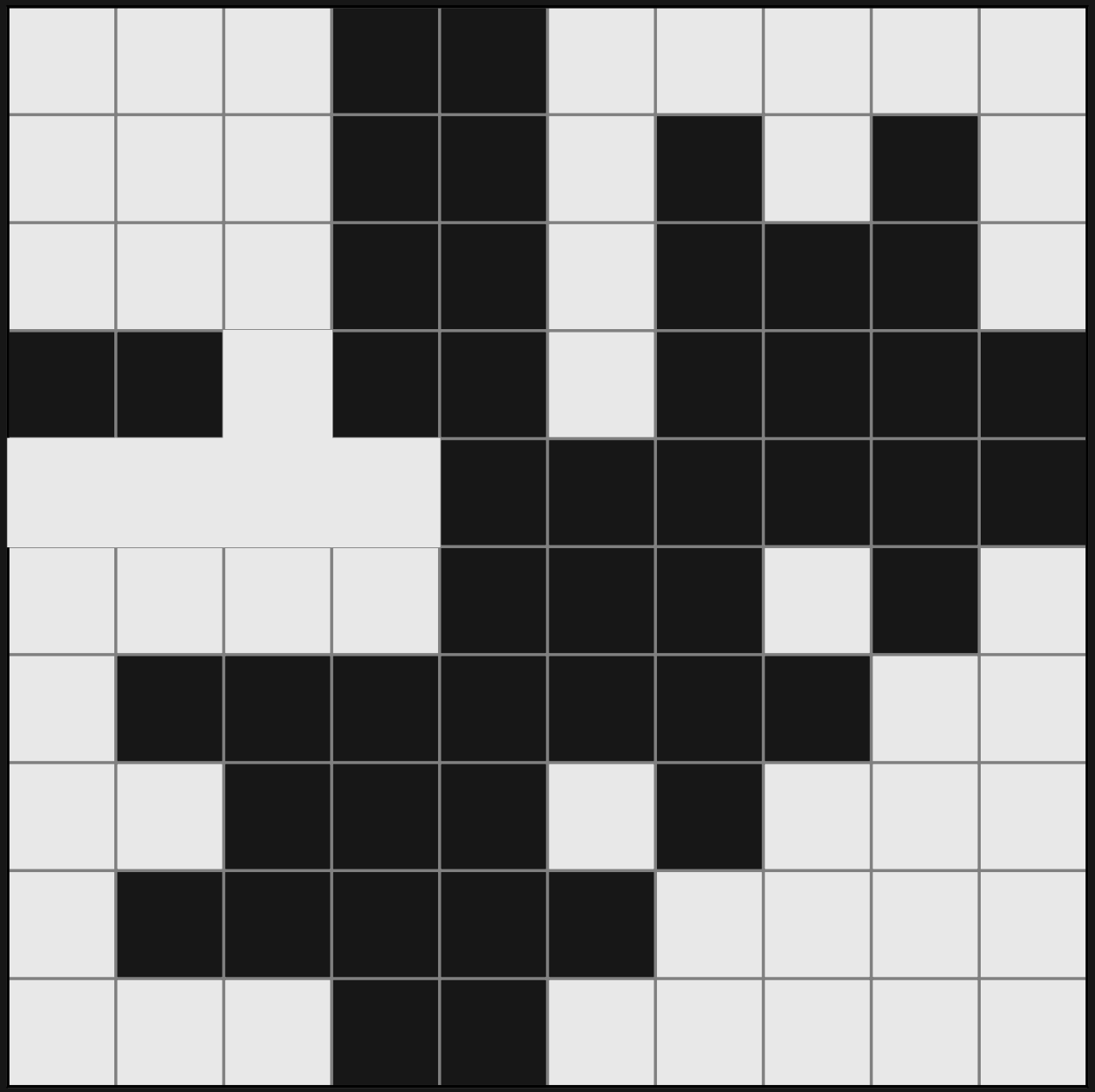
OR gate in f_2



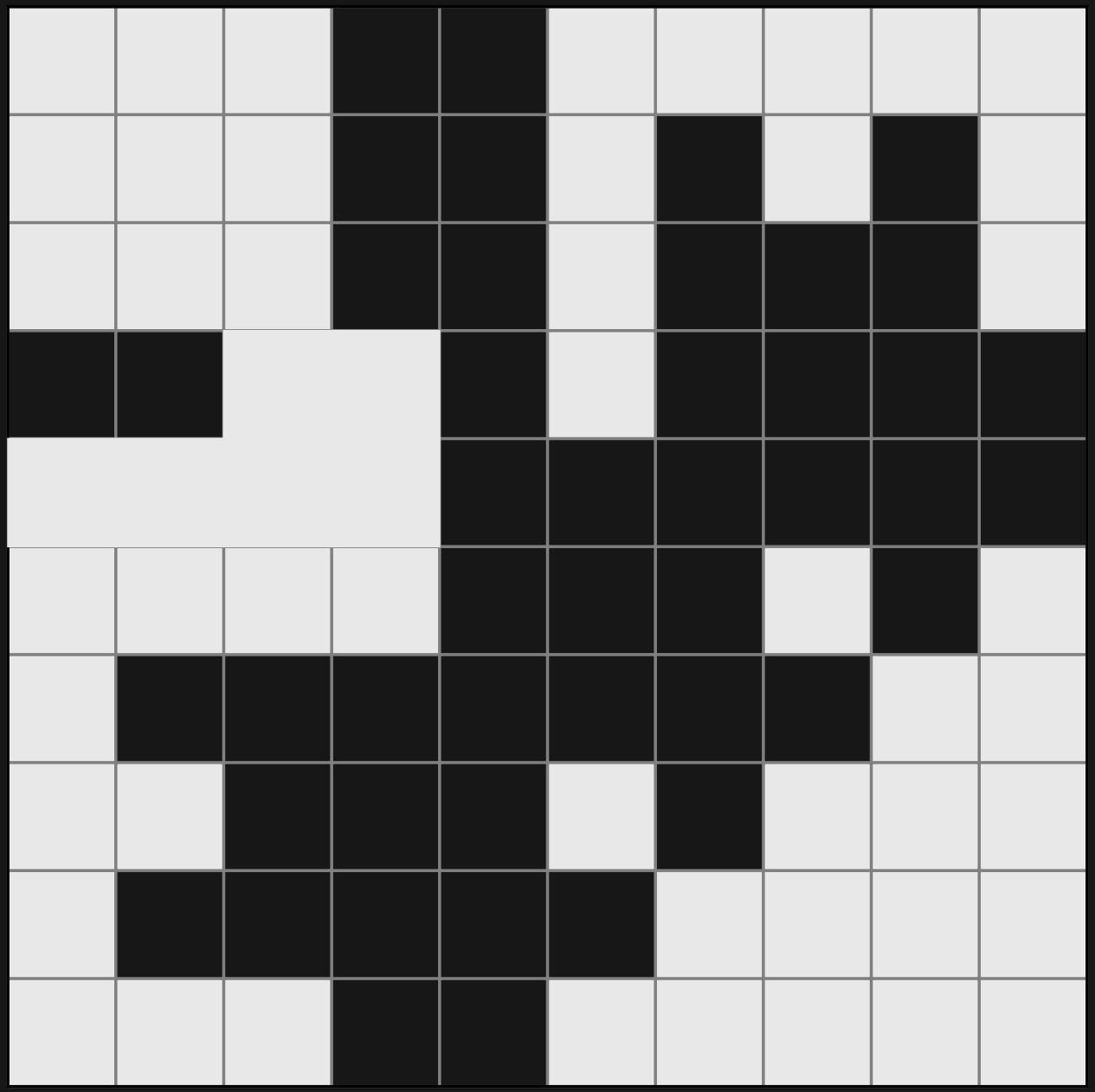
OR gate in f_2



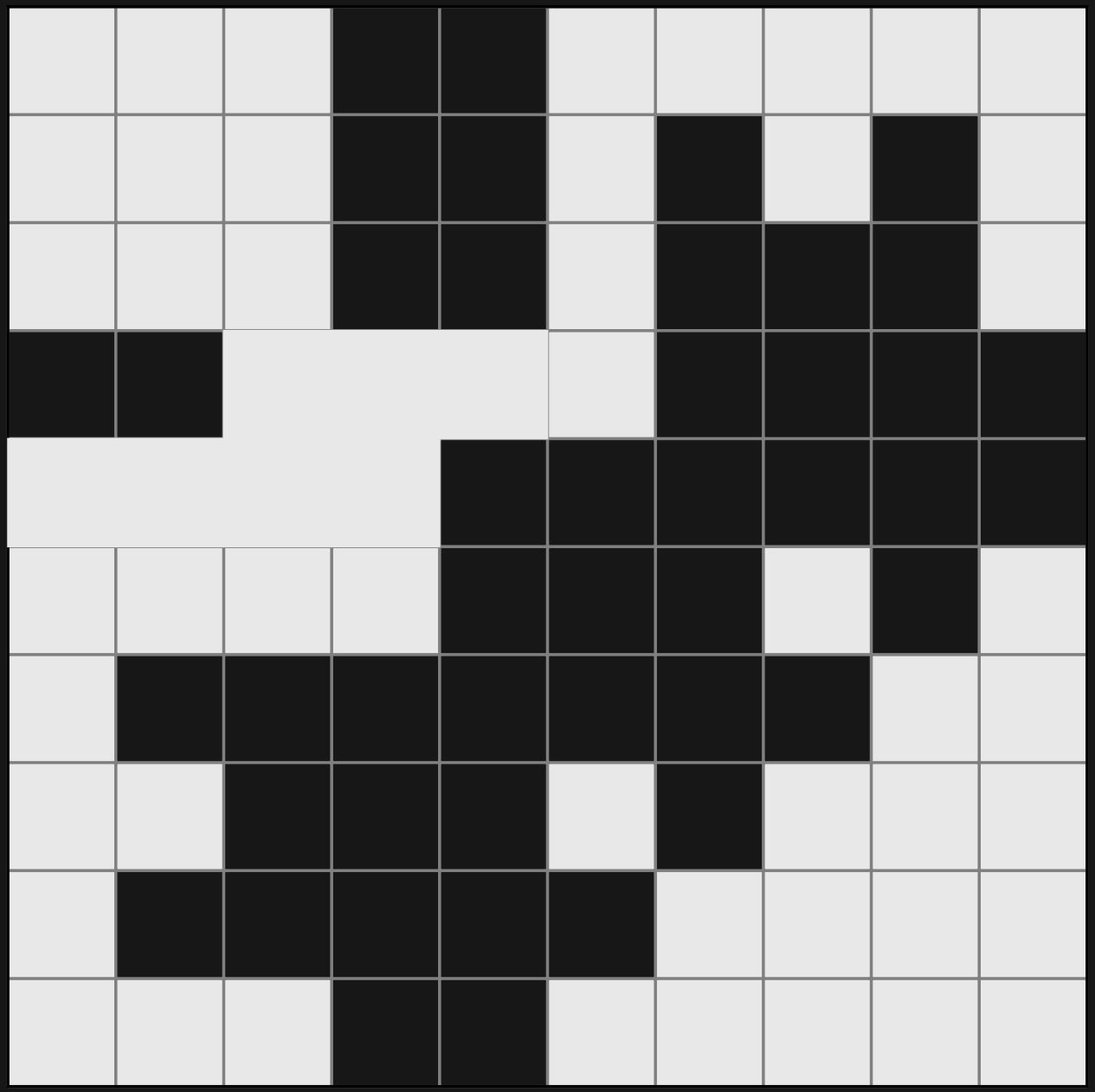
OR gate in f_2



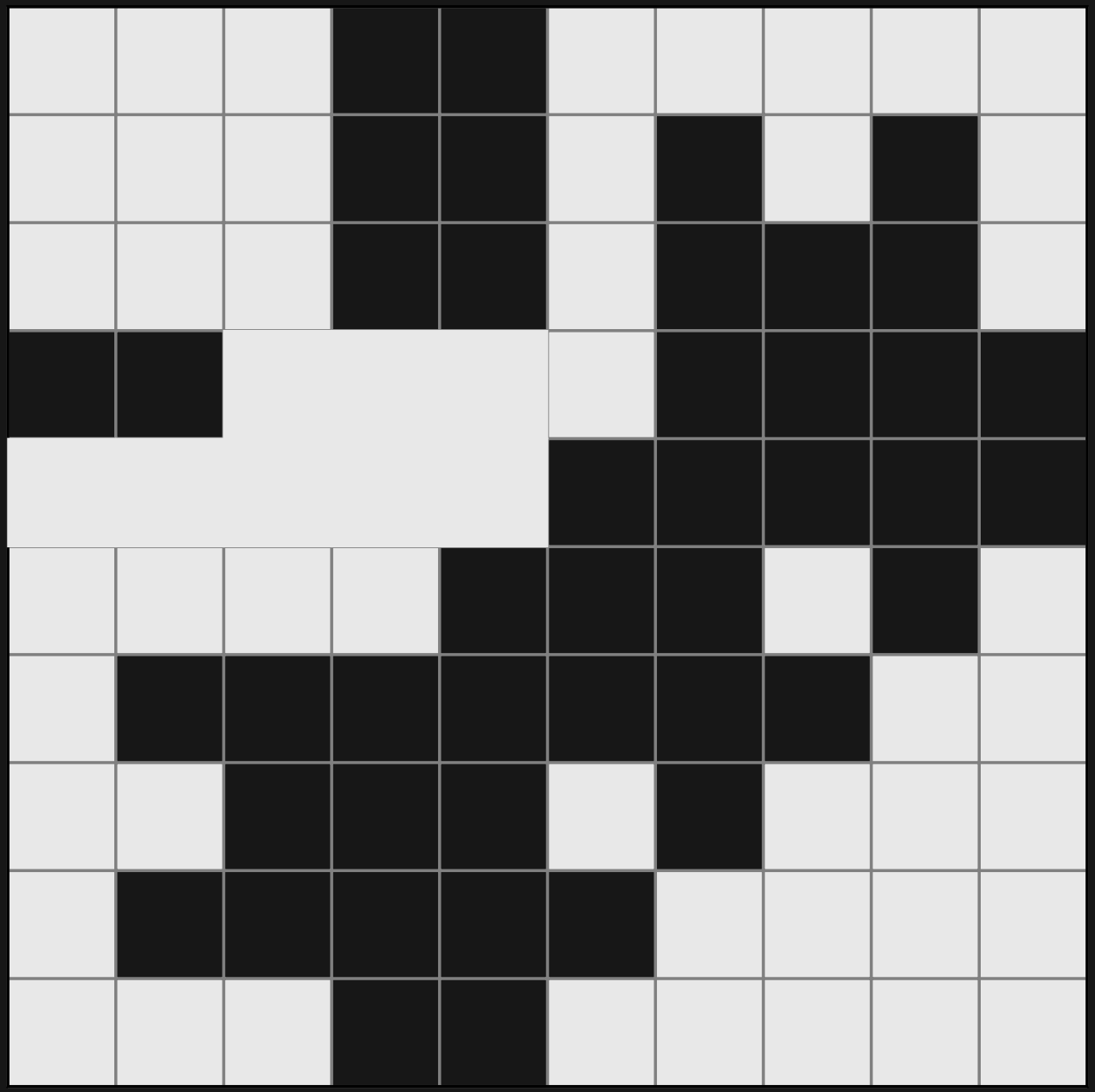
OR gate in f_2



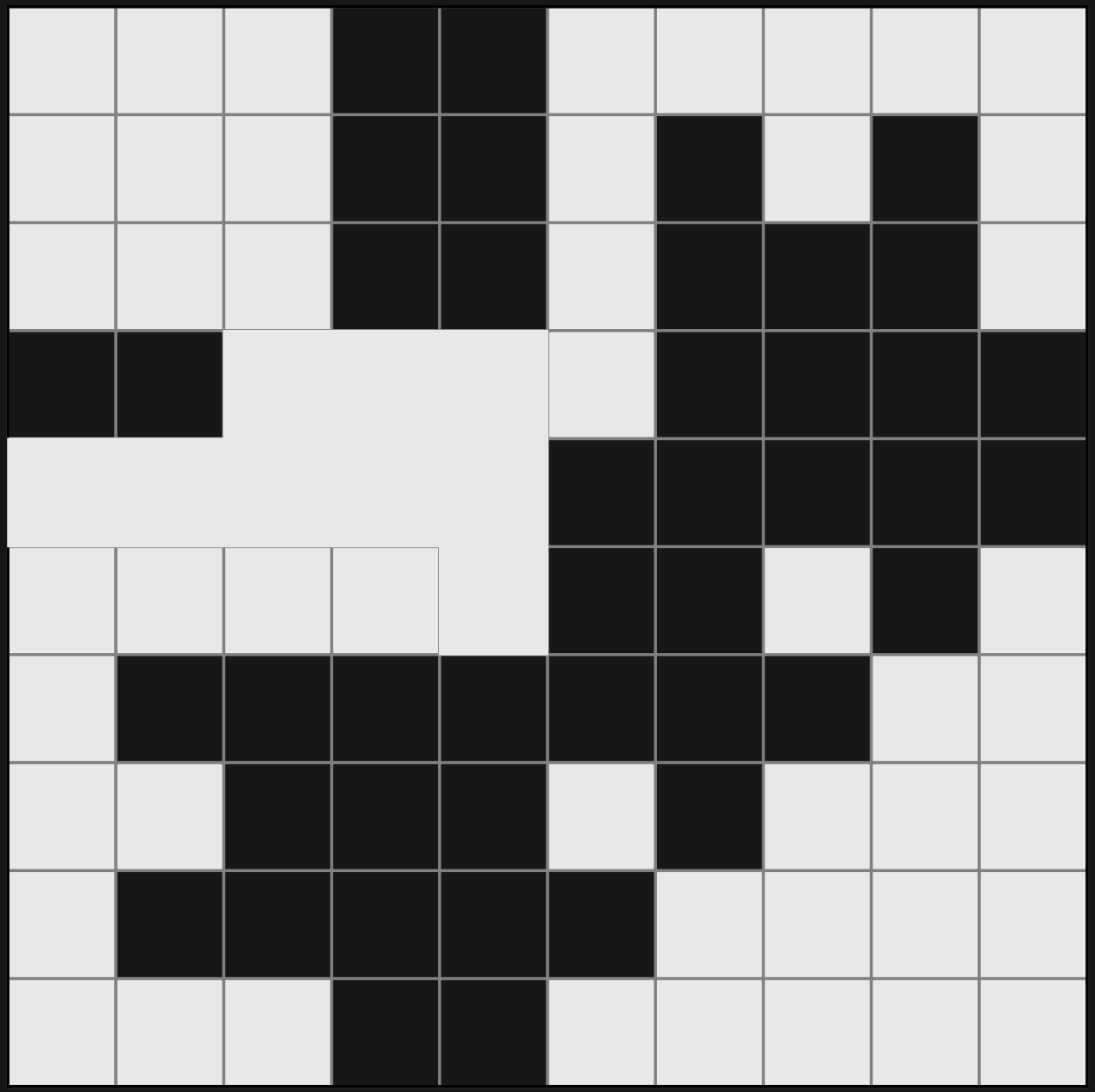
OR gate in f_2



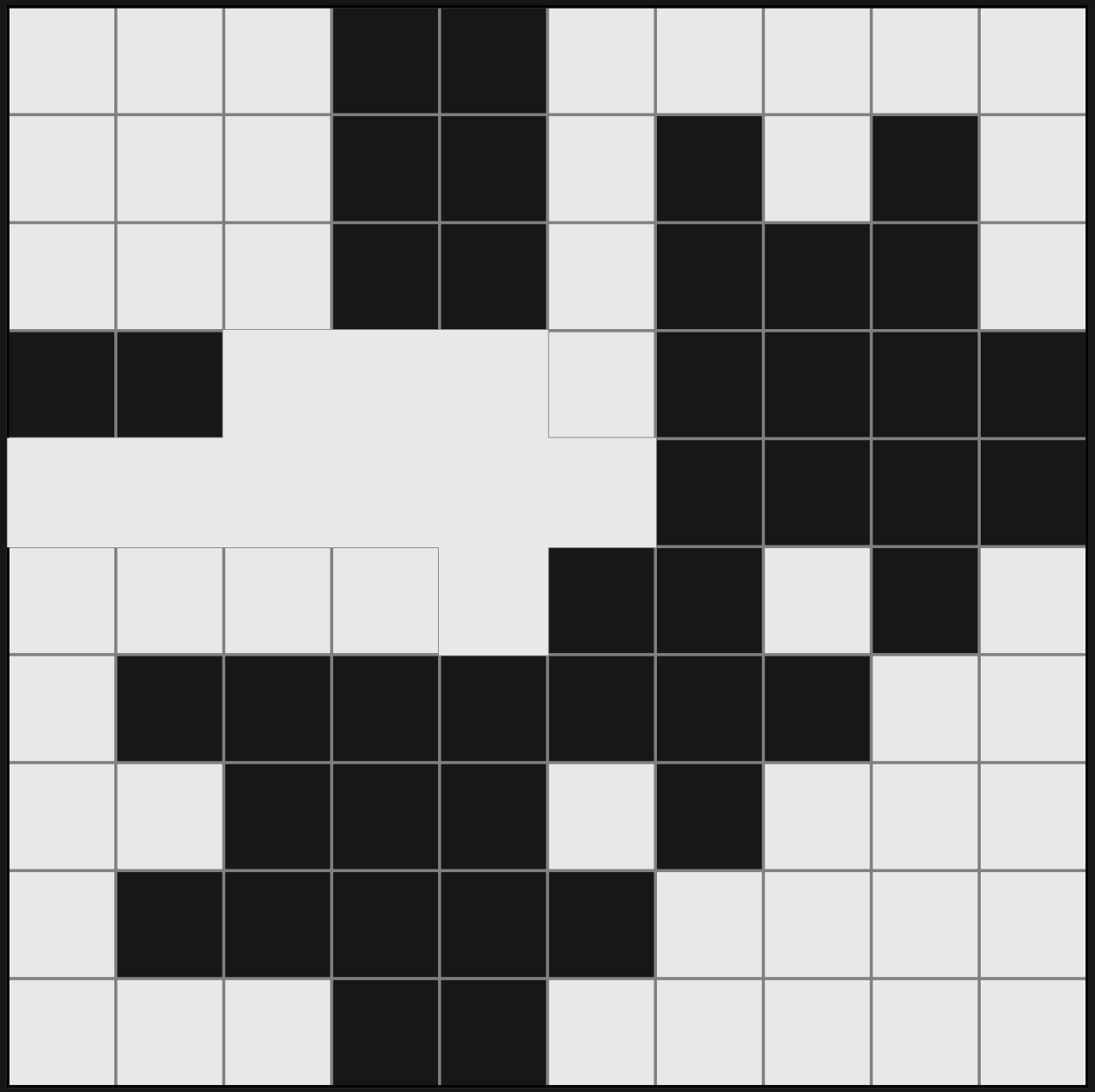
OR gate in f_2



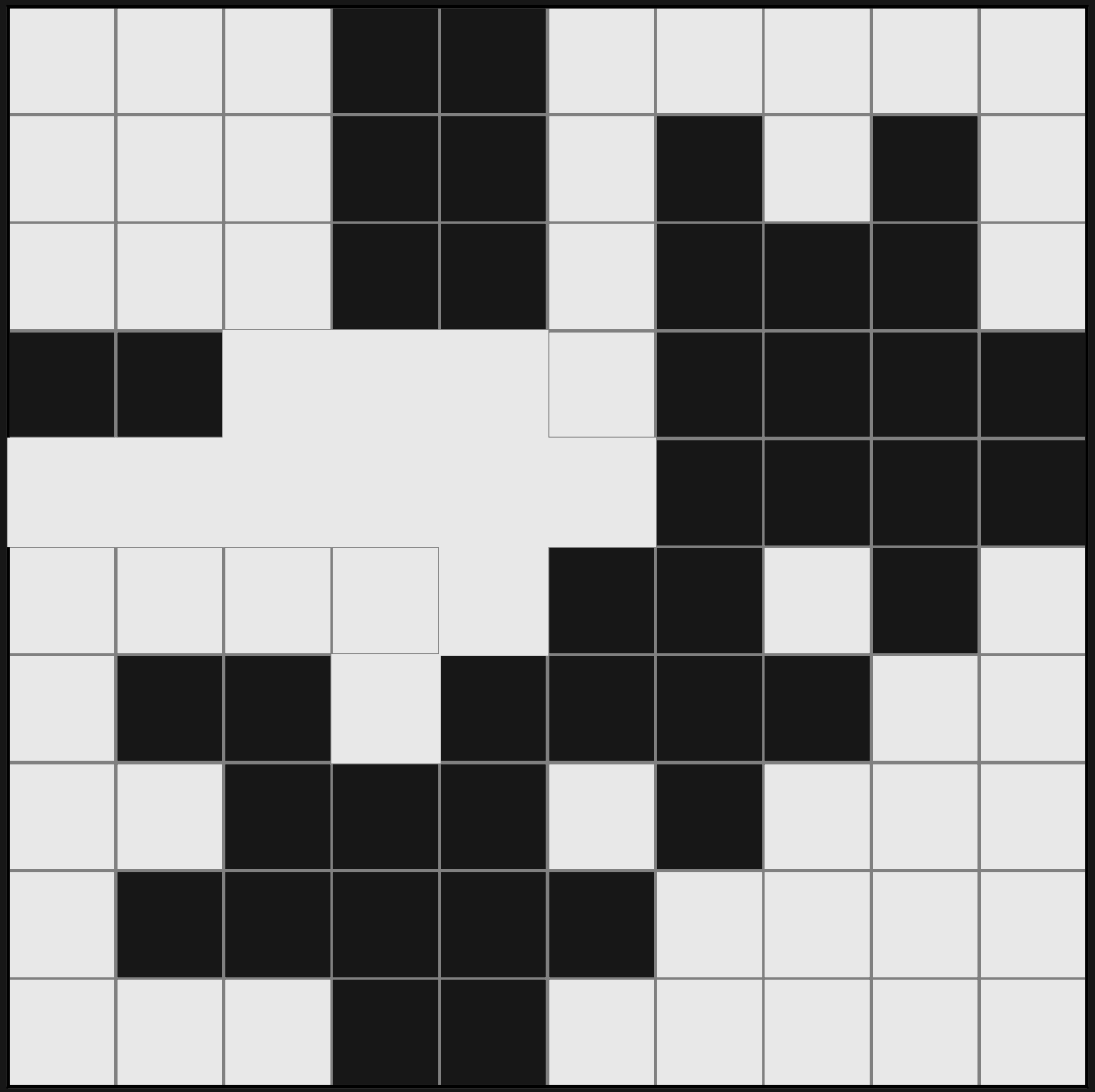
OR gate in f_2



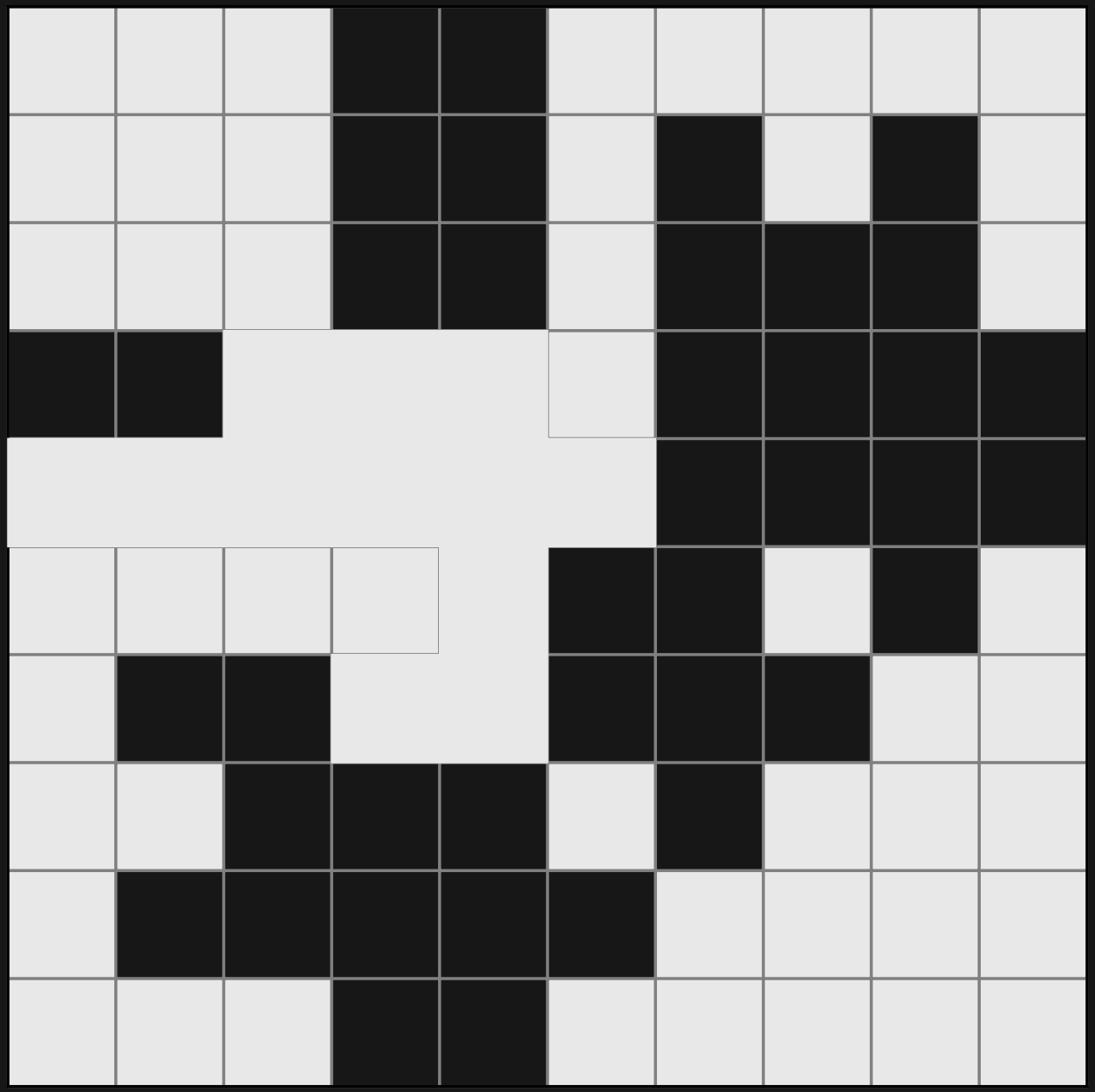
OR gate in f_2



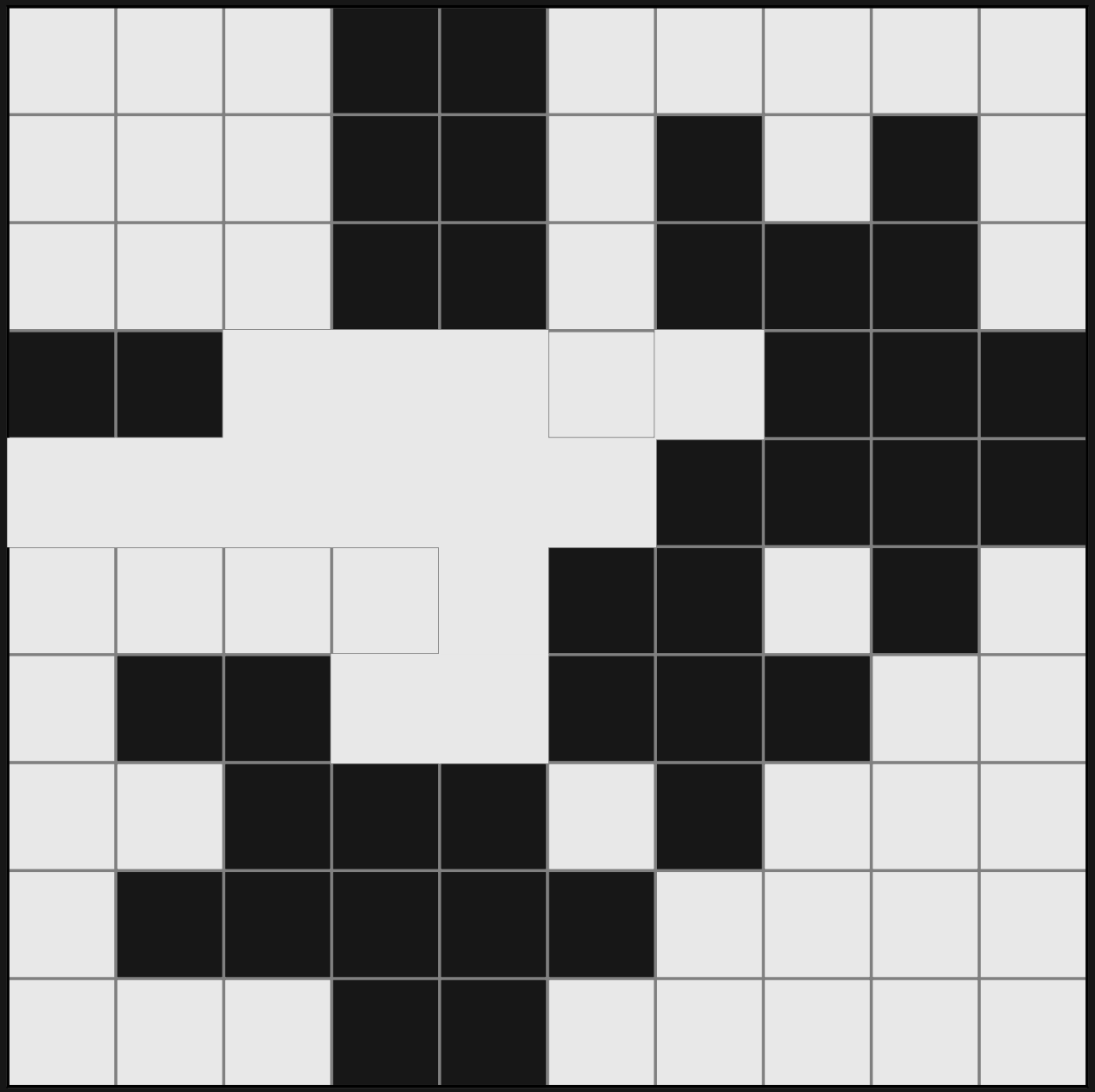
OR gate in f_2



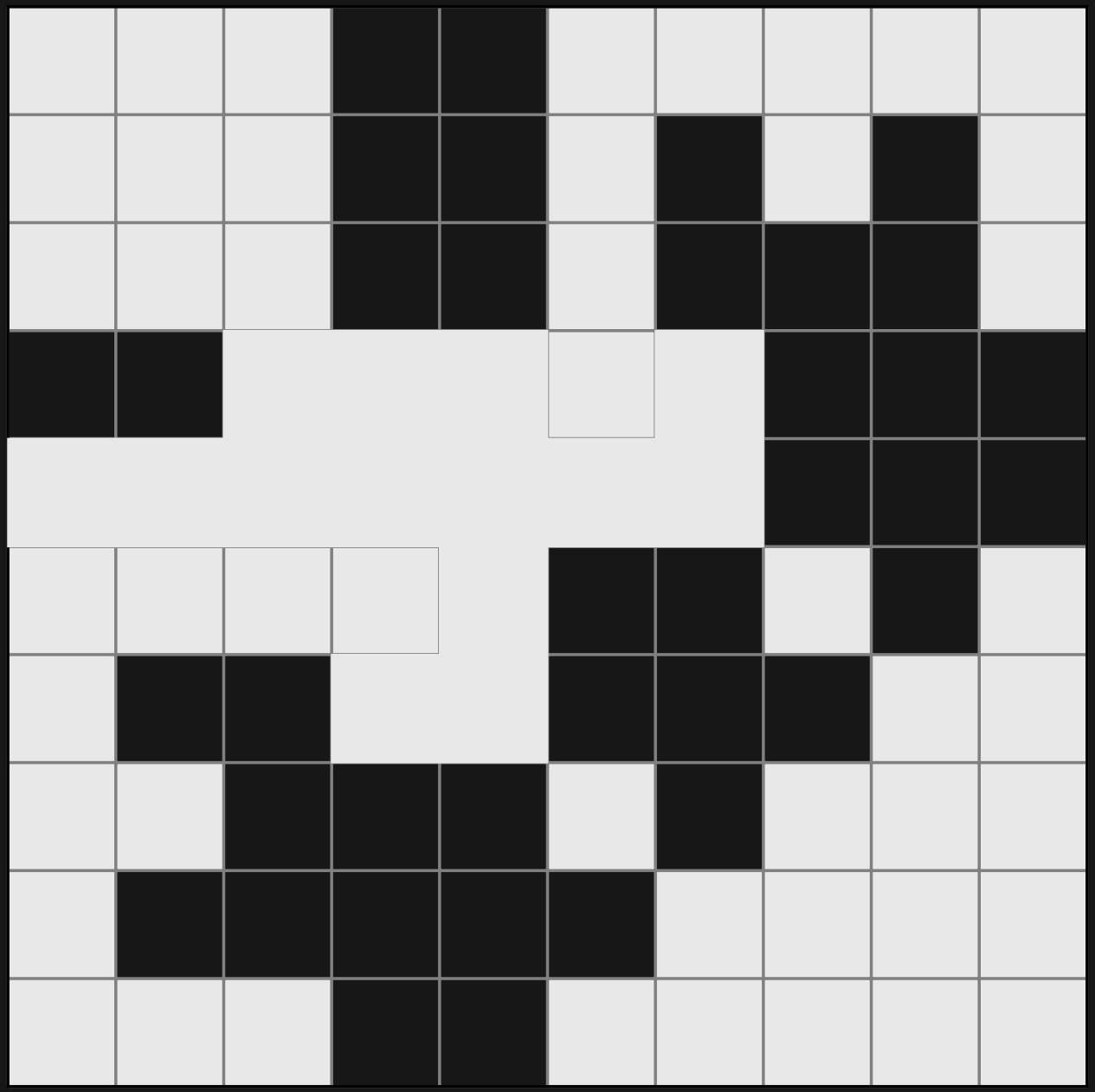
OR gate in f_2



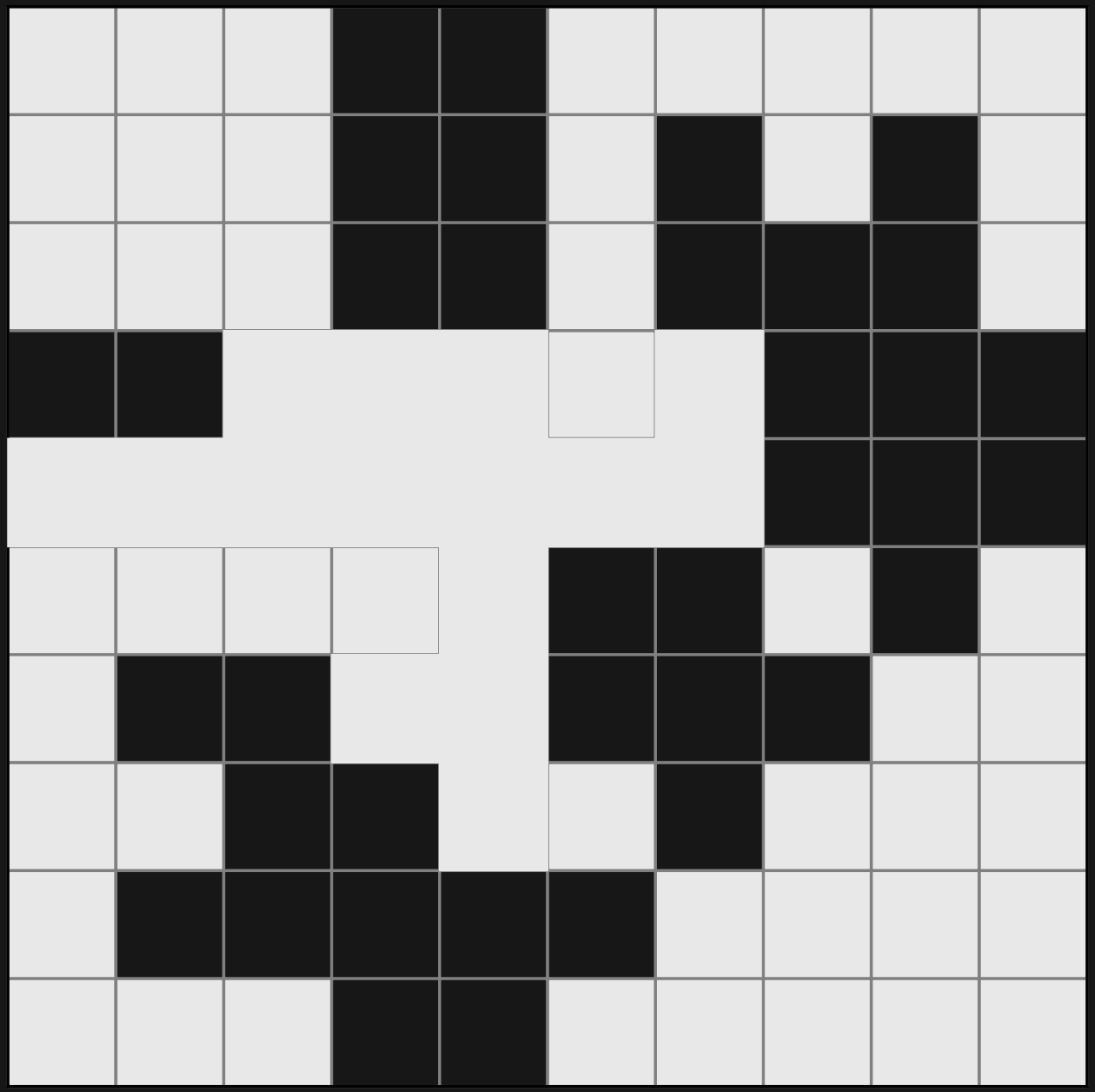
OR gate in f_2



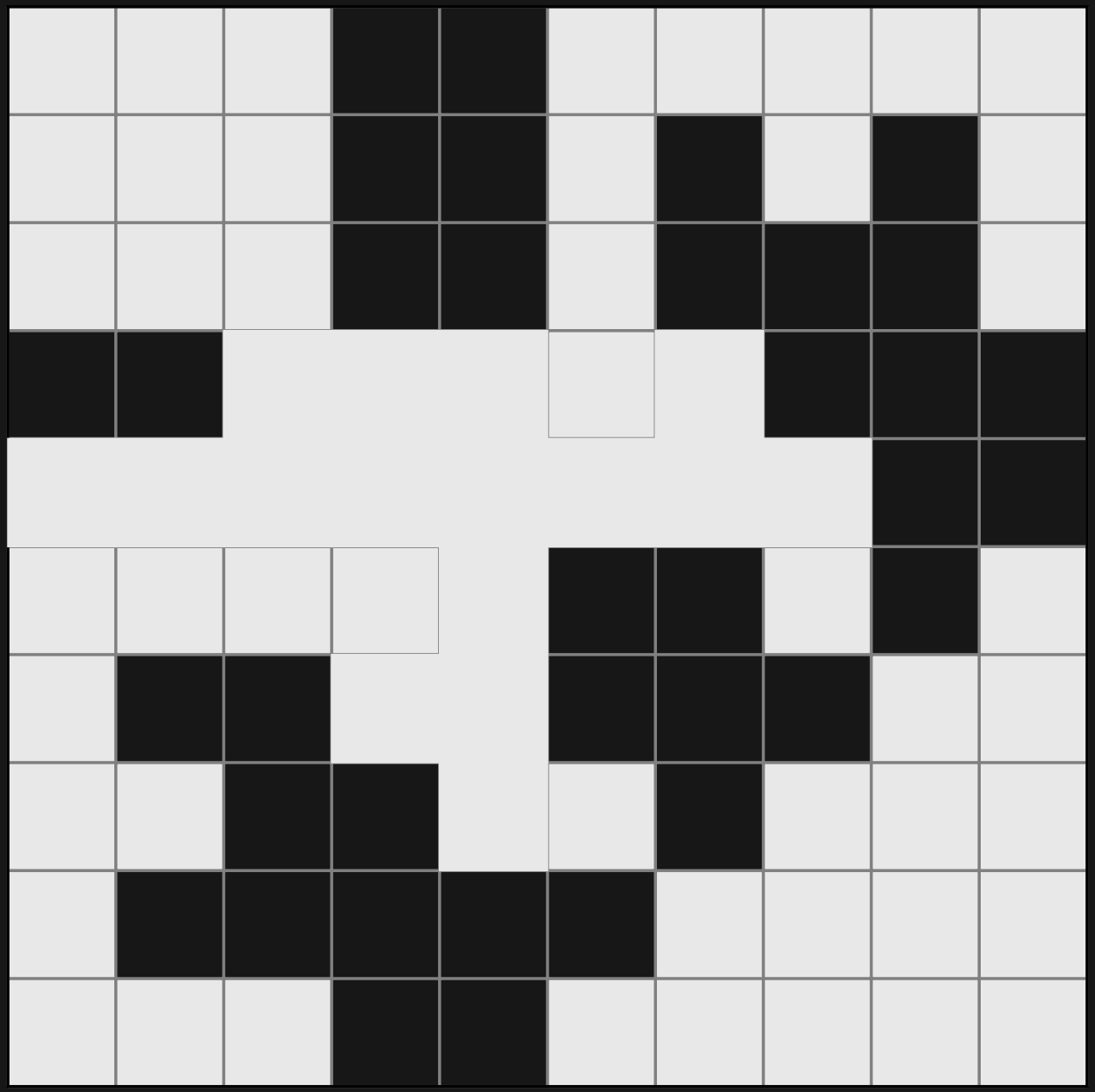
OR gate in f_2



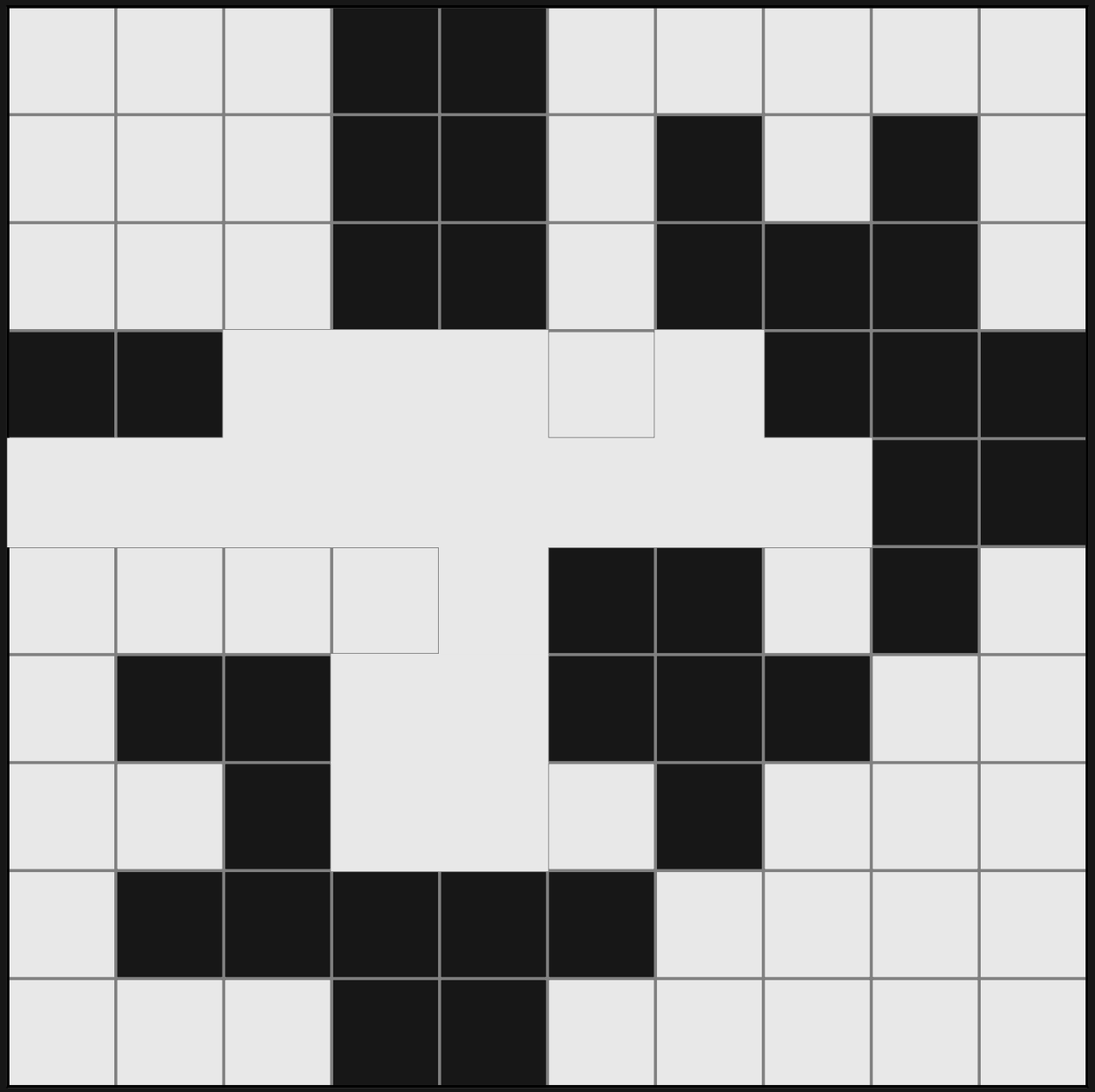
OR gate in f_2



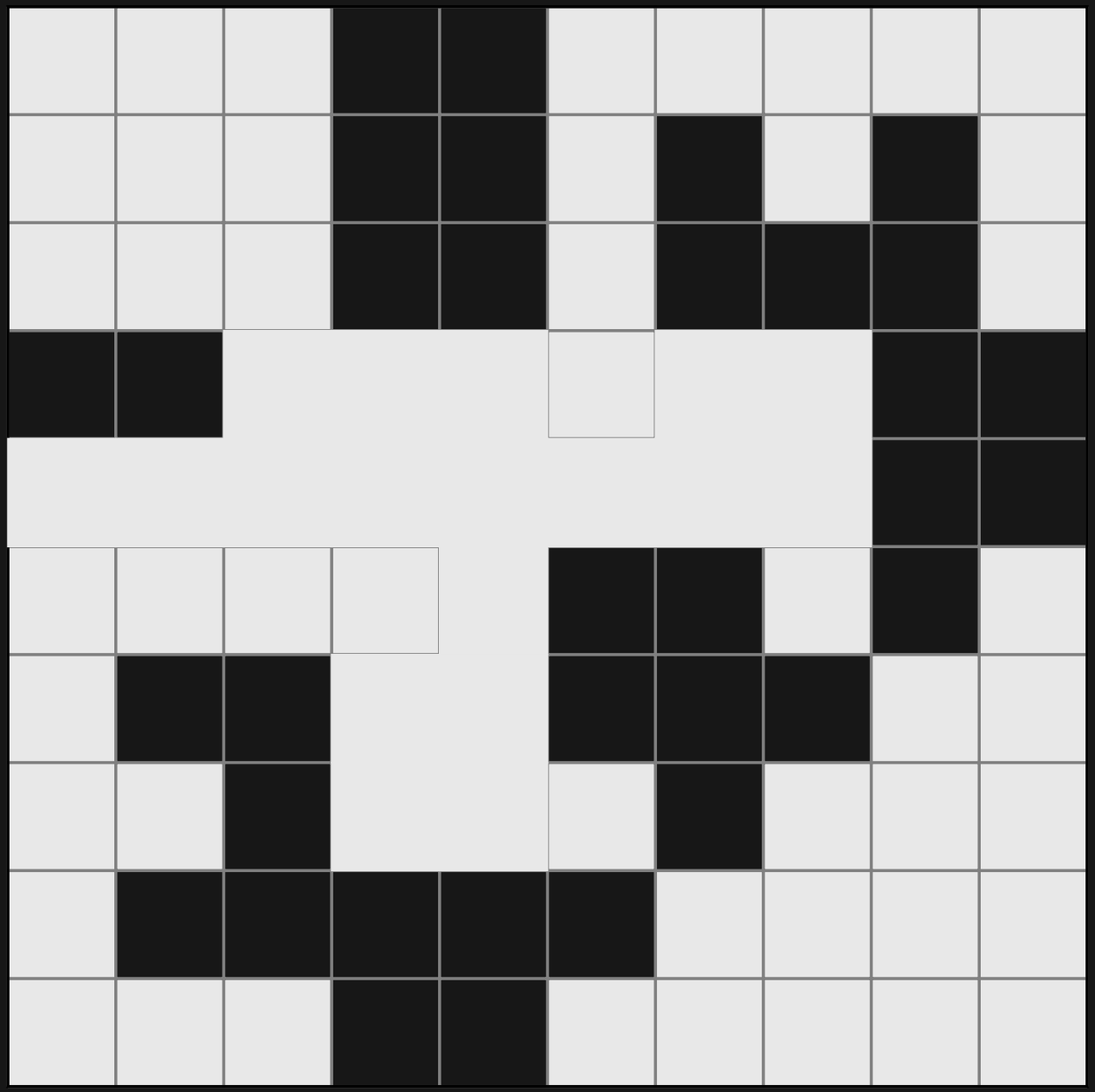
OR gate in f_2



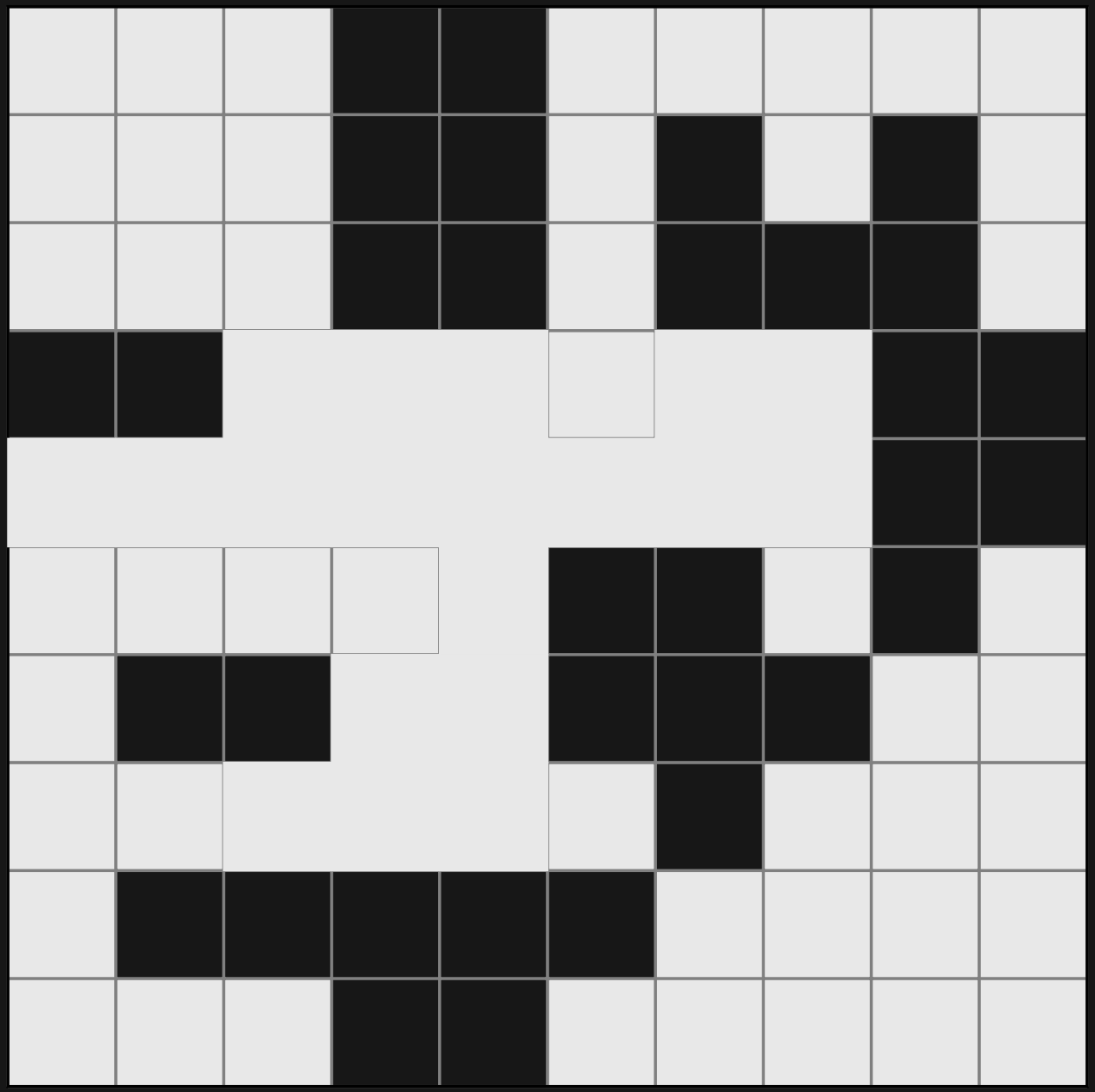
OR gate in f_2



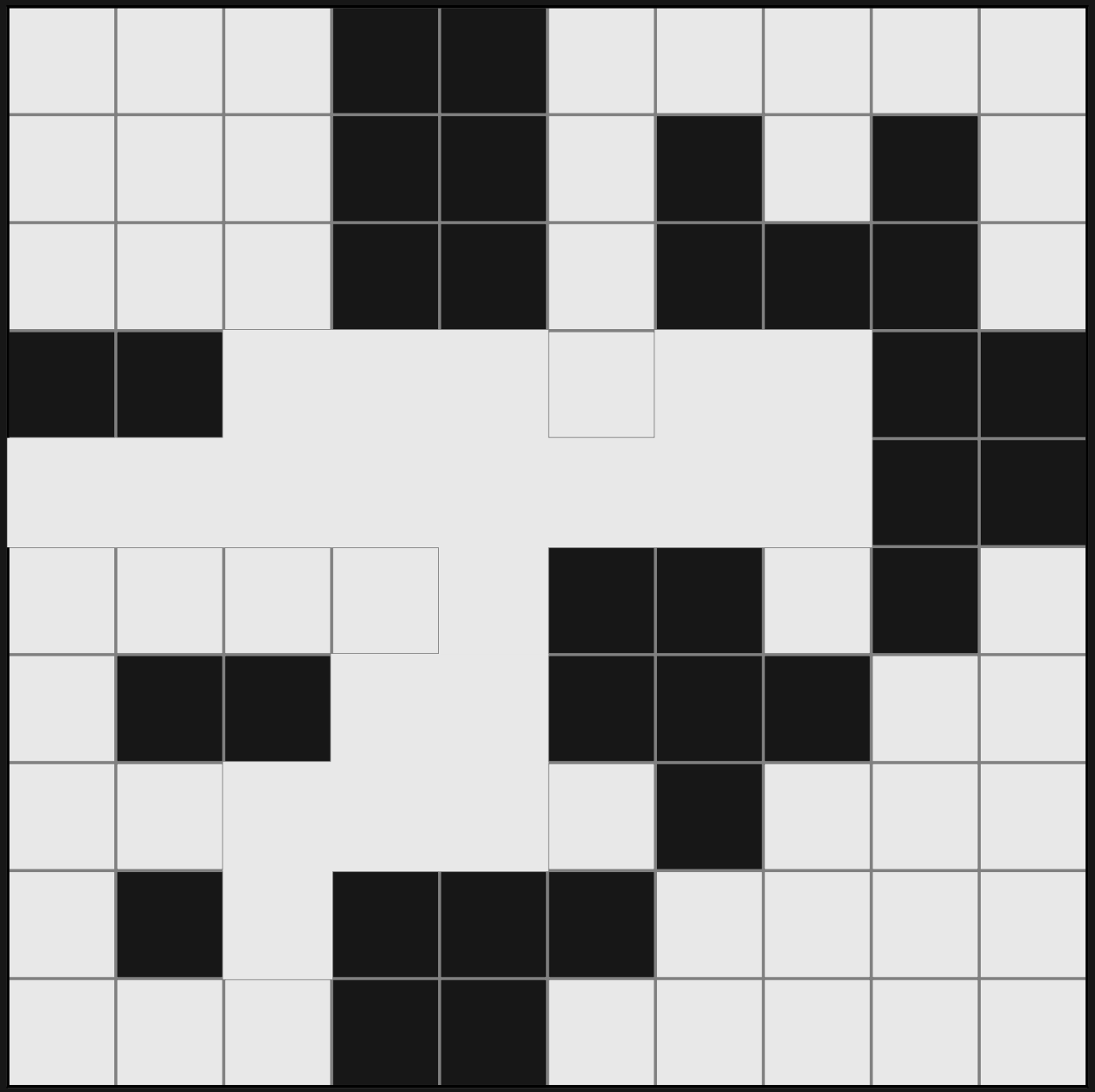
OR gate in f_2



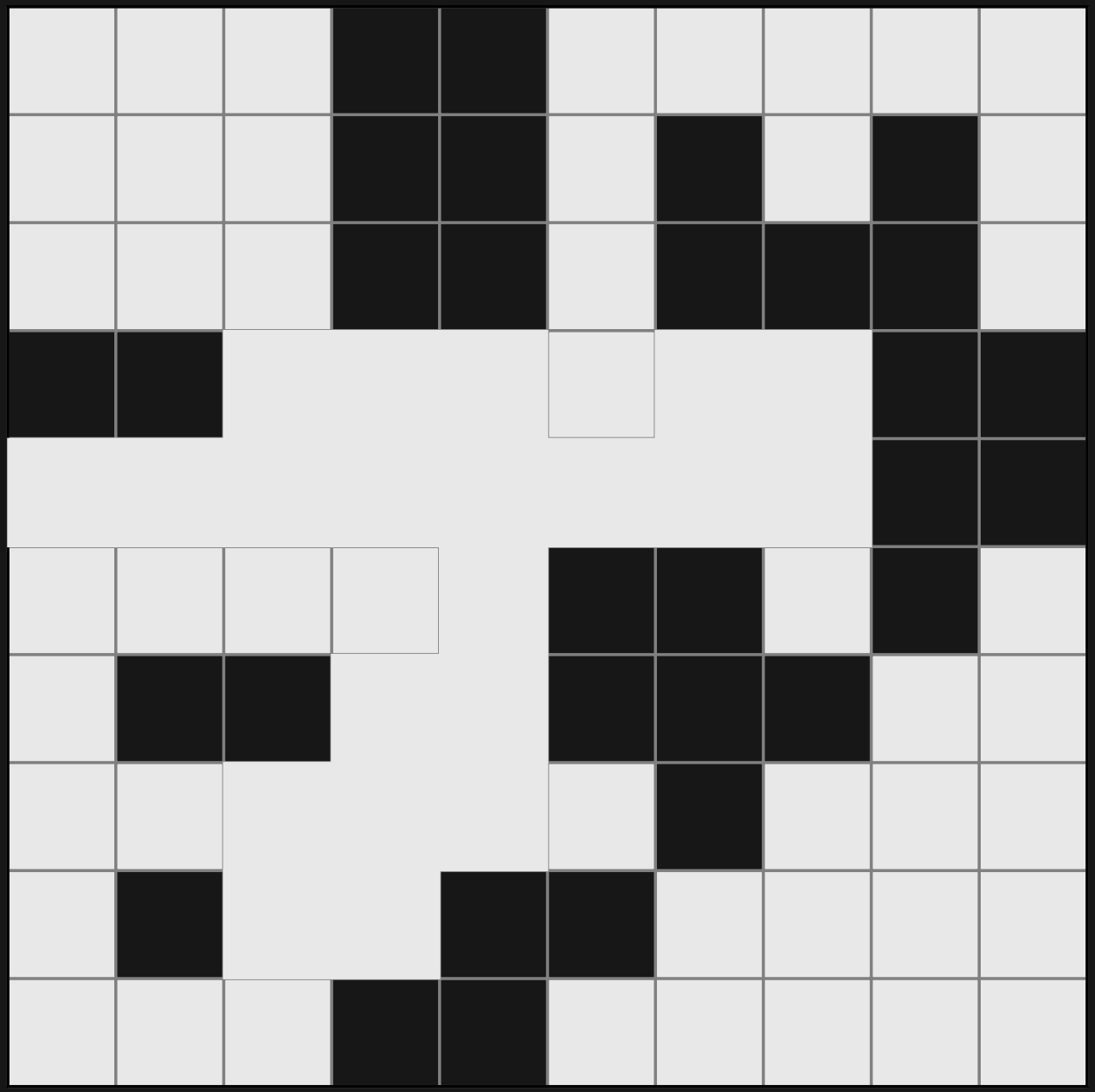
OR gate in f_2



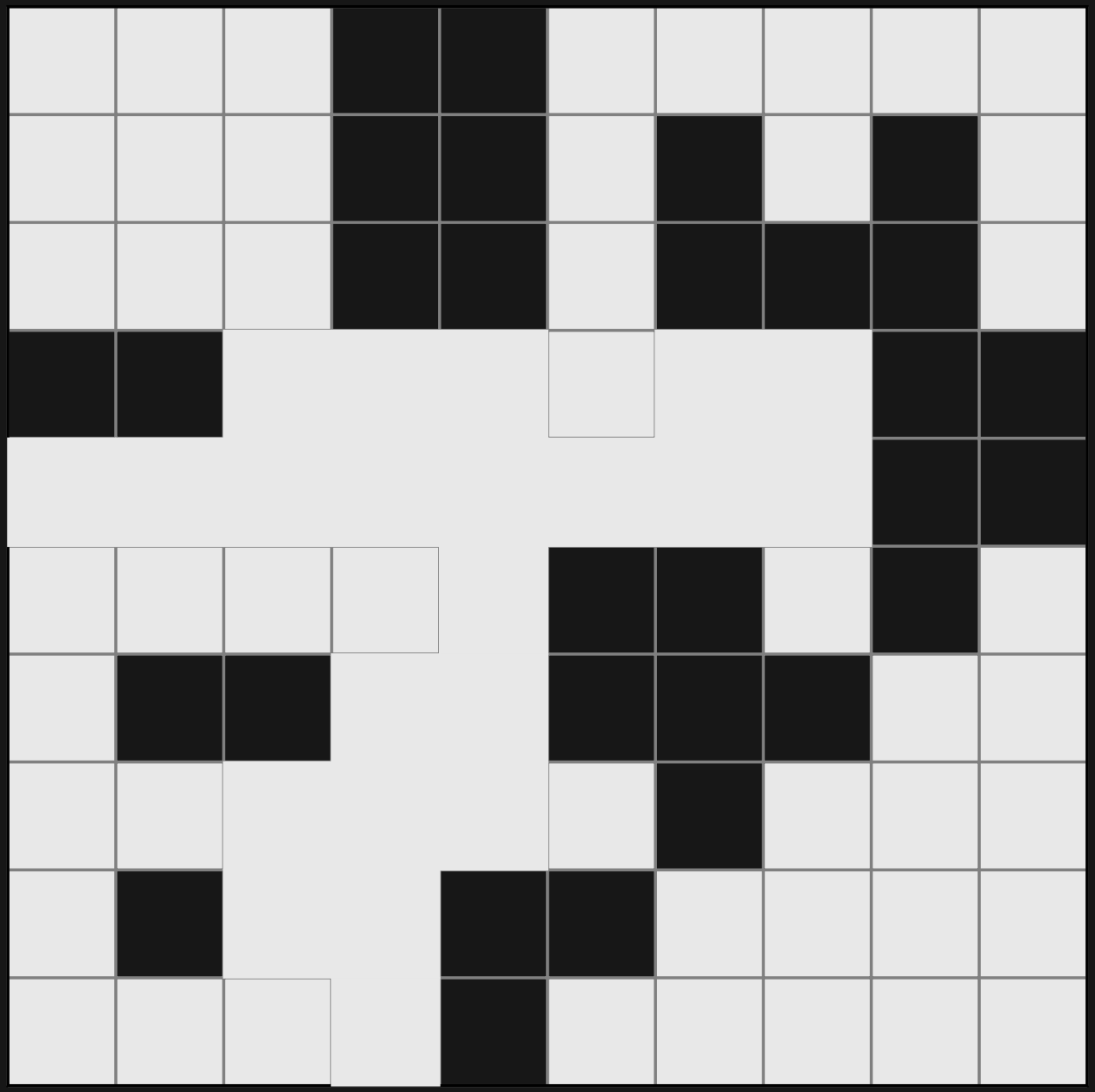
OR gate in f_2



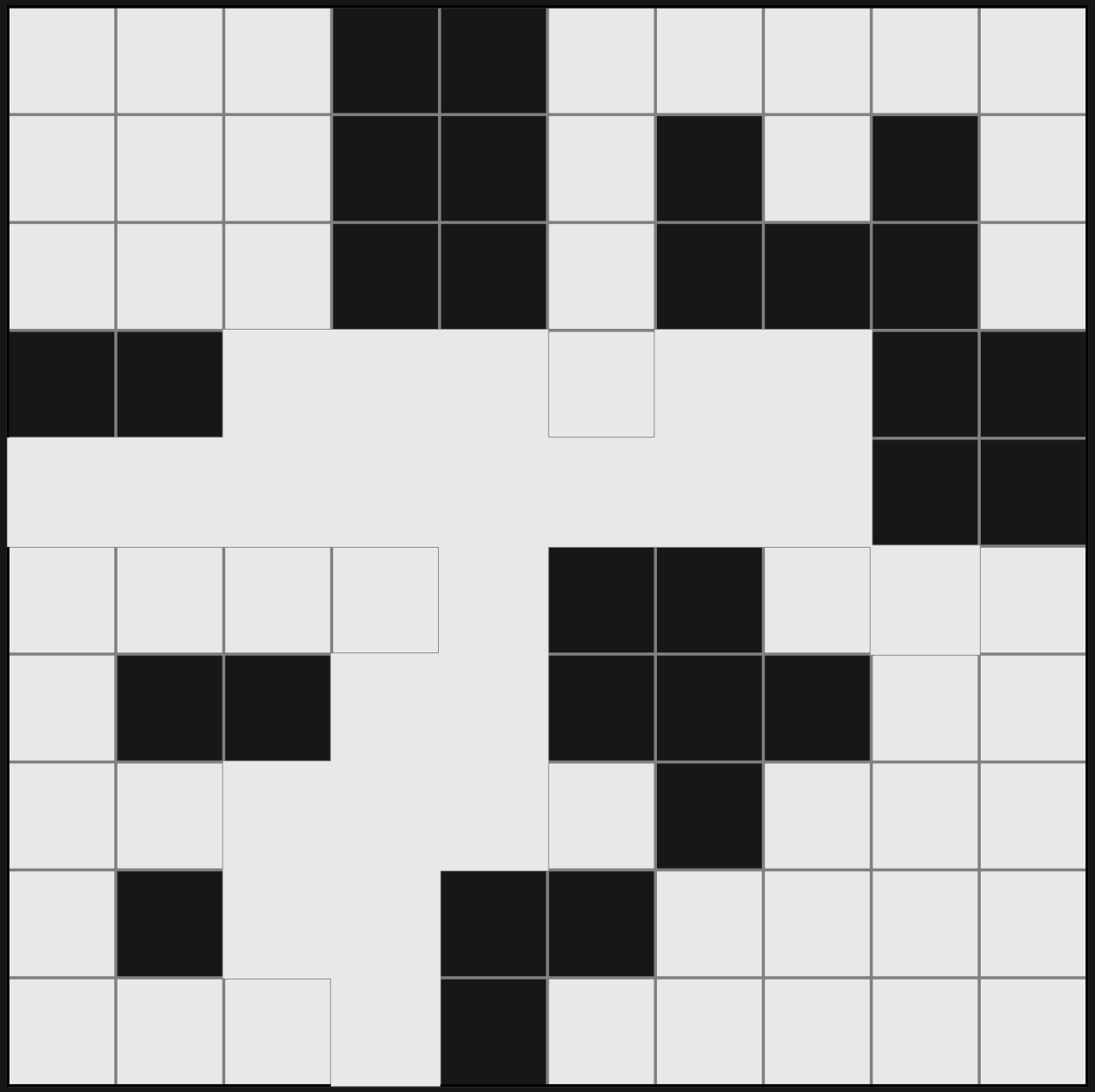
OR gate in f_2



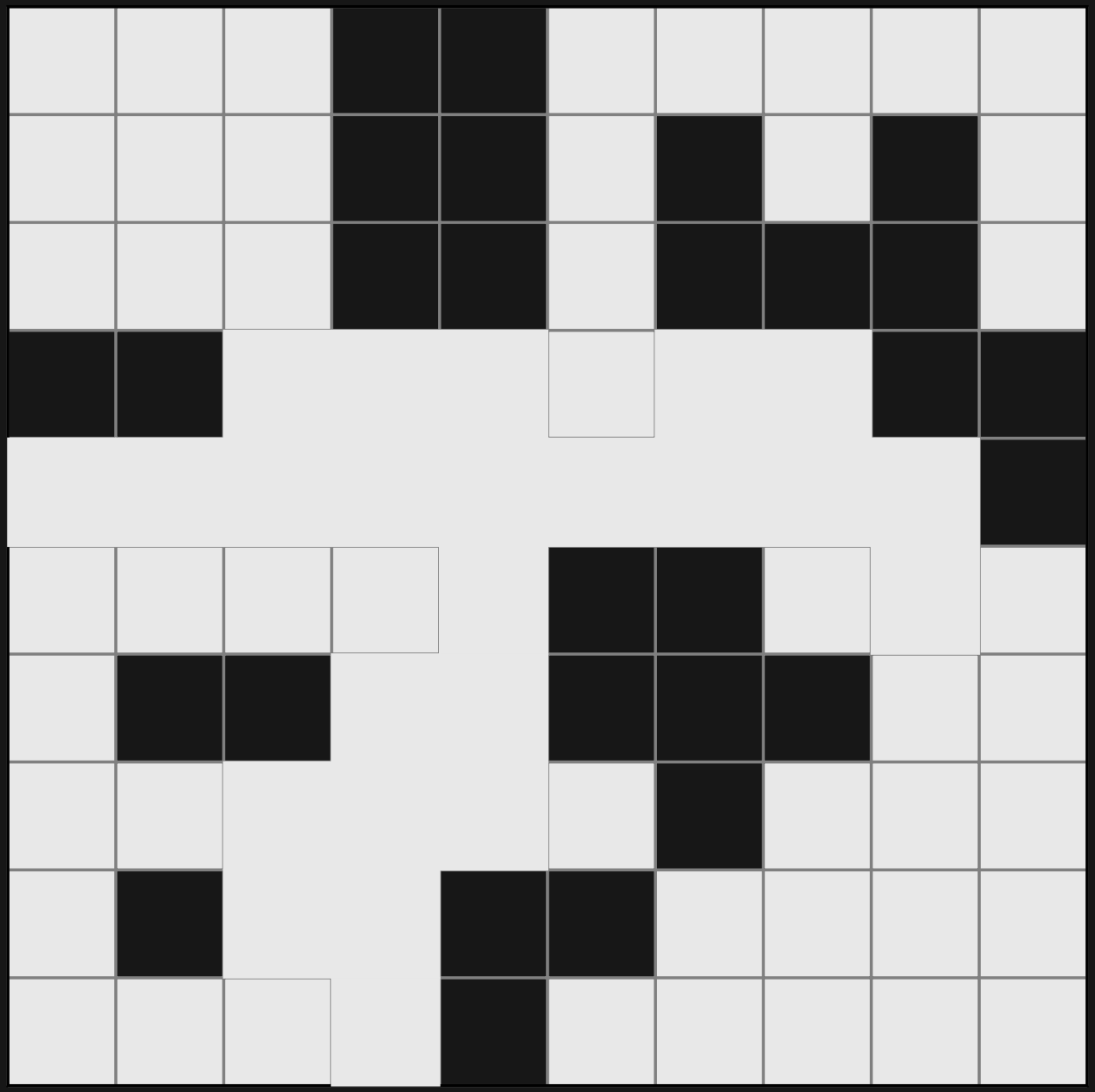
OR gate in f_2



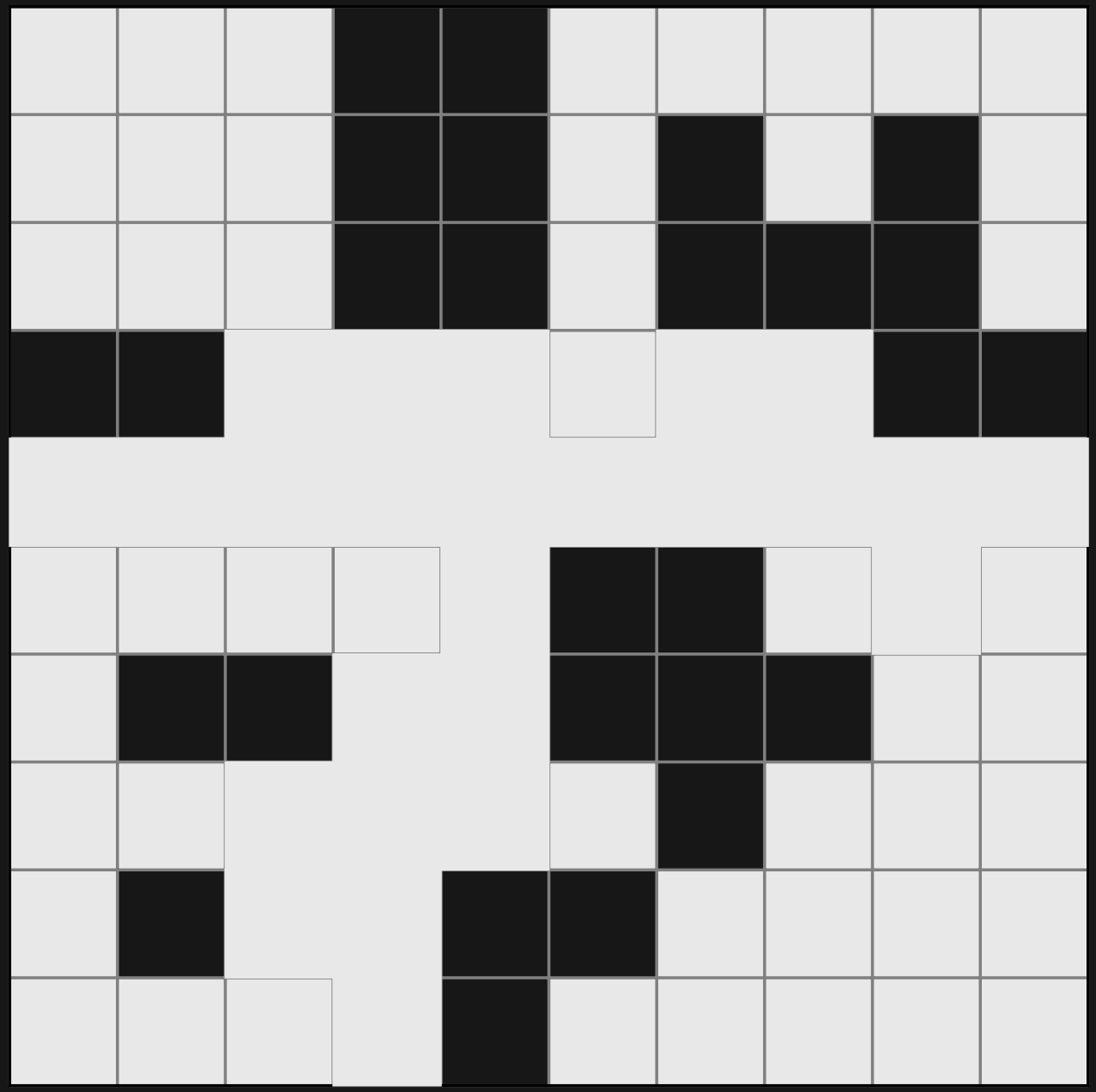
OR gate in f_2



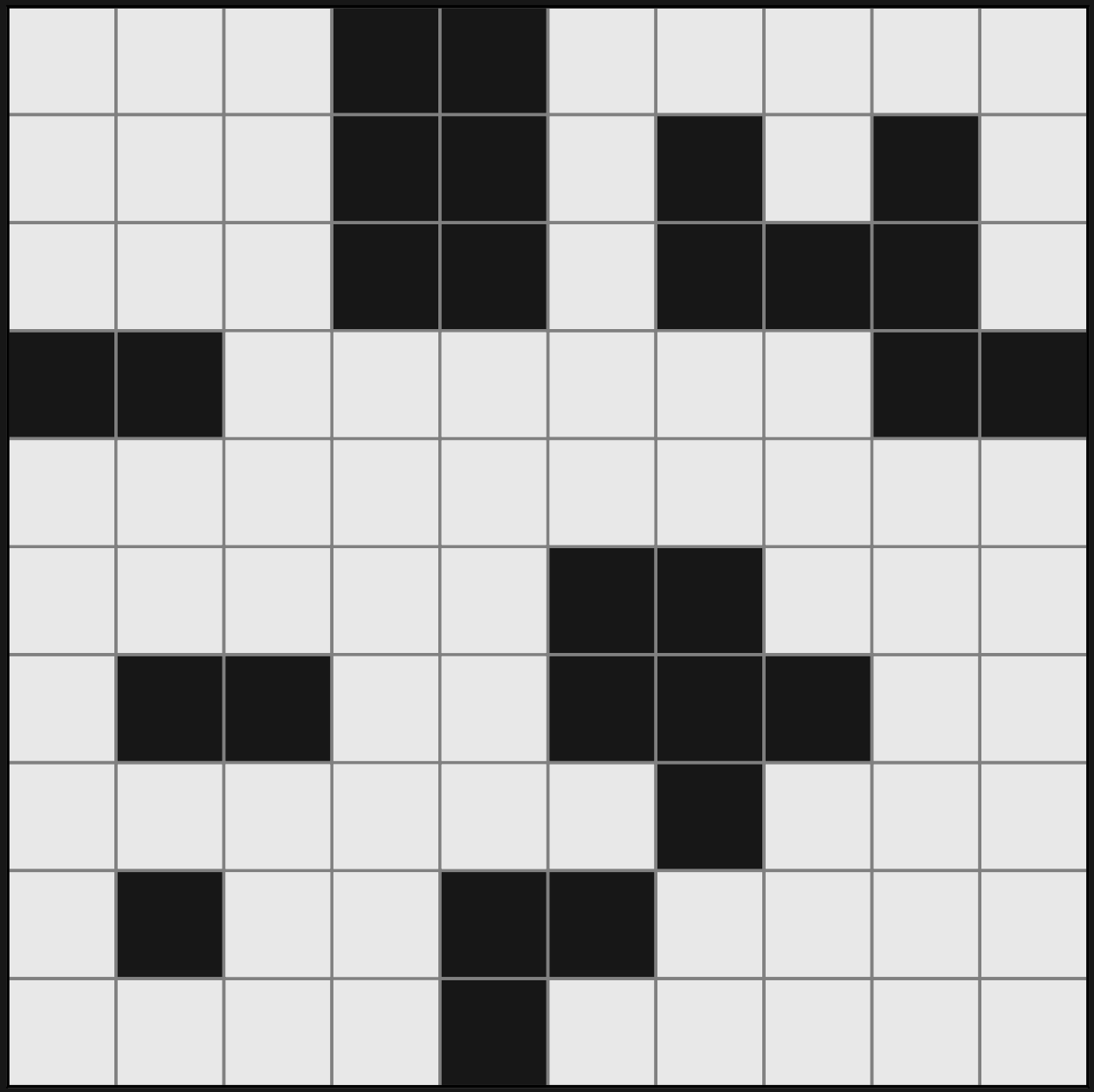
OR gate in f_2



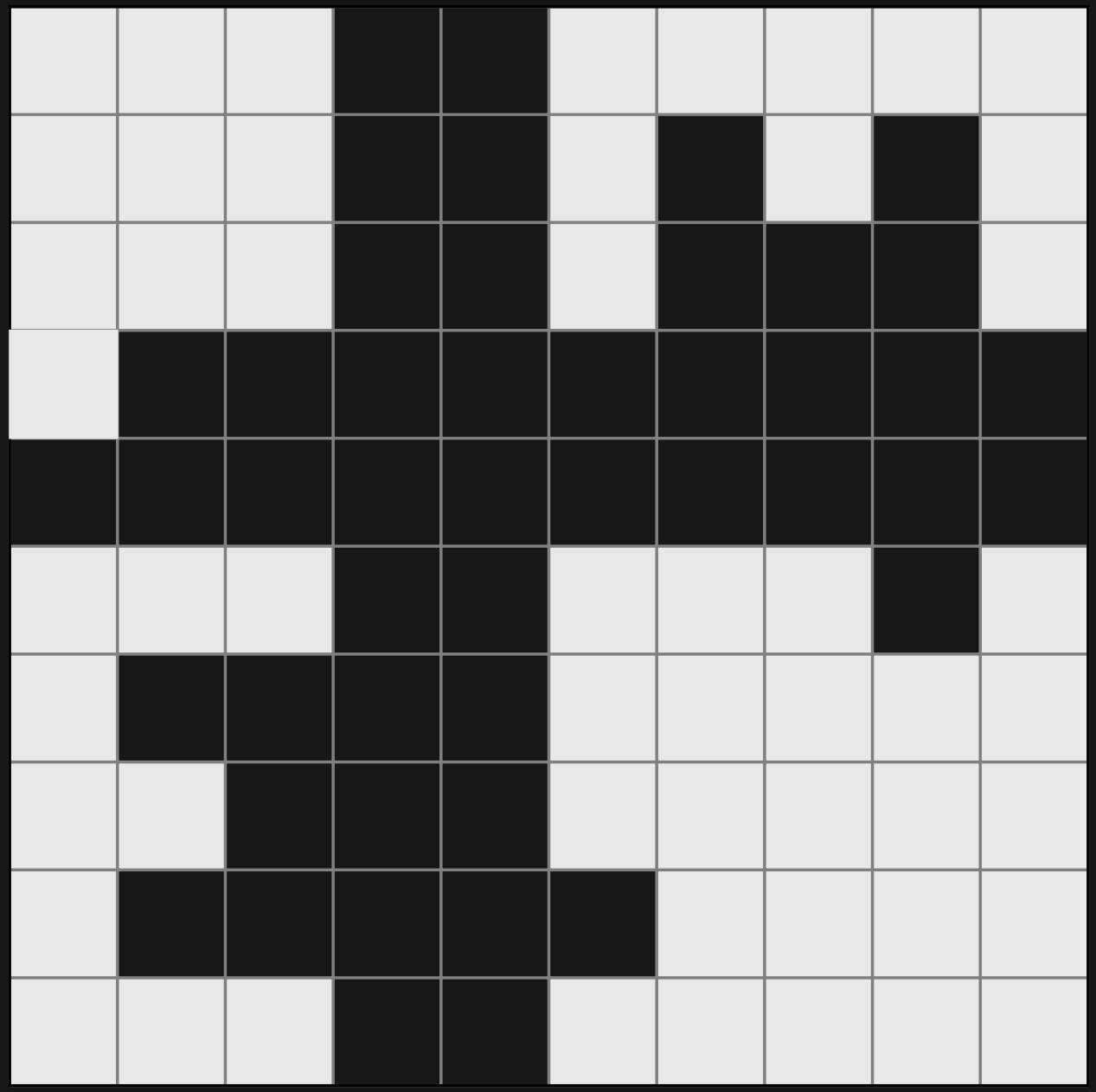
OR gate in f_2



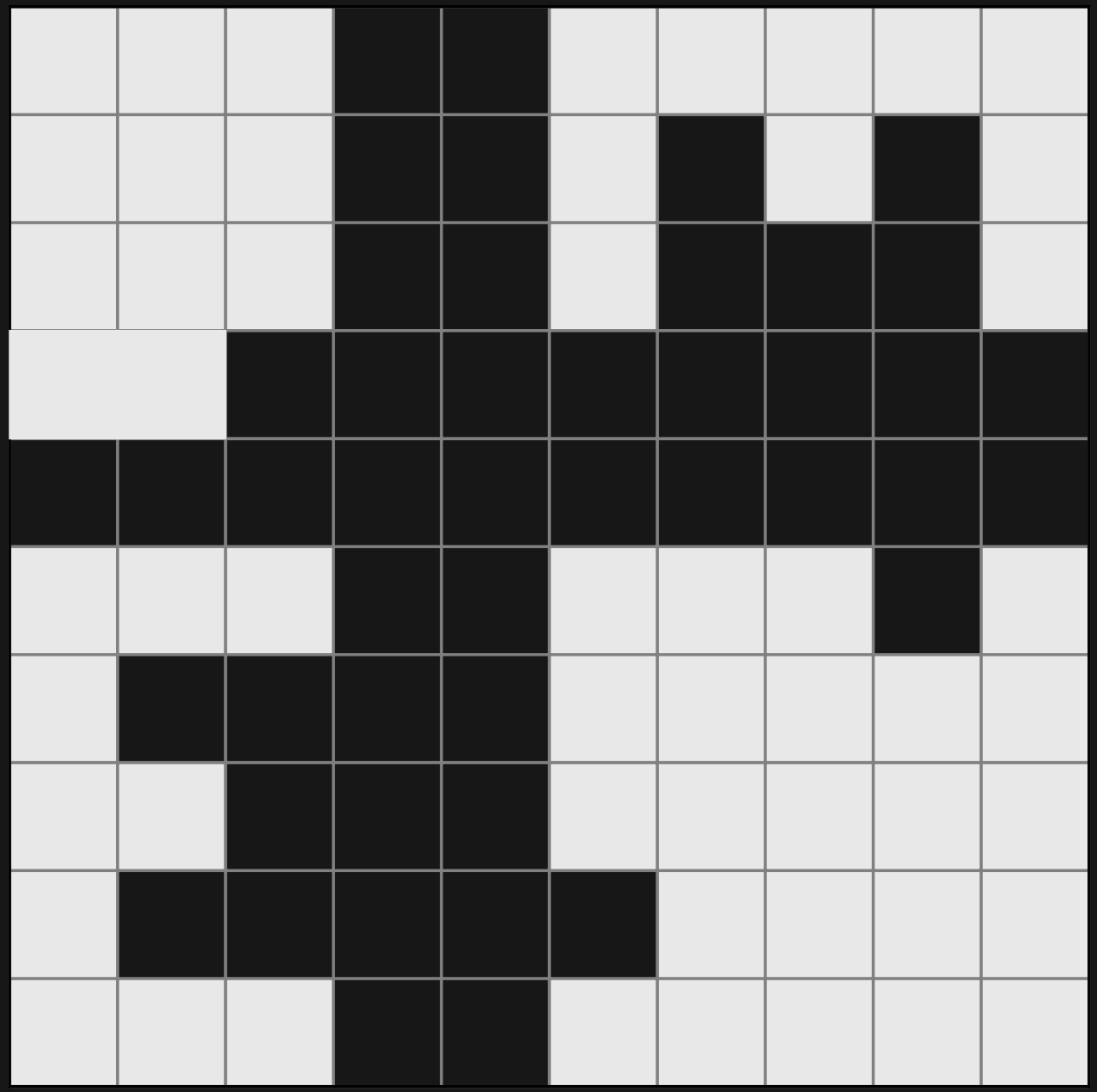
OR gate in f_2



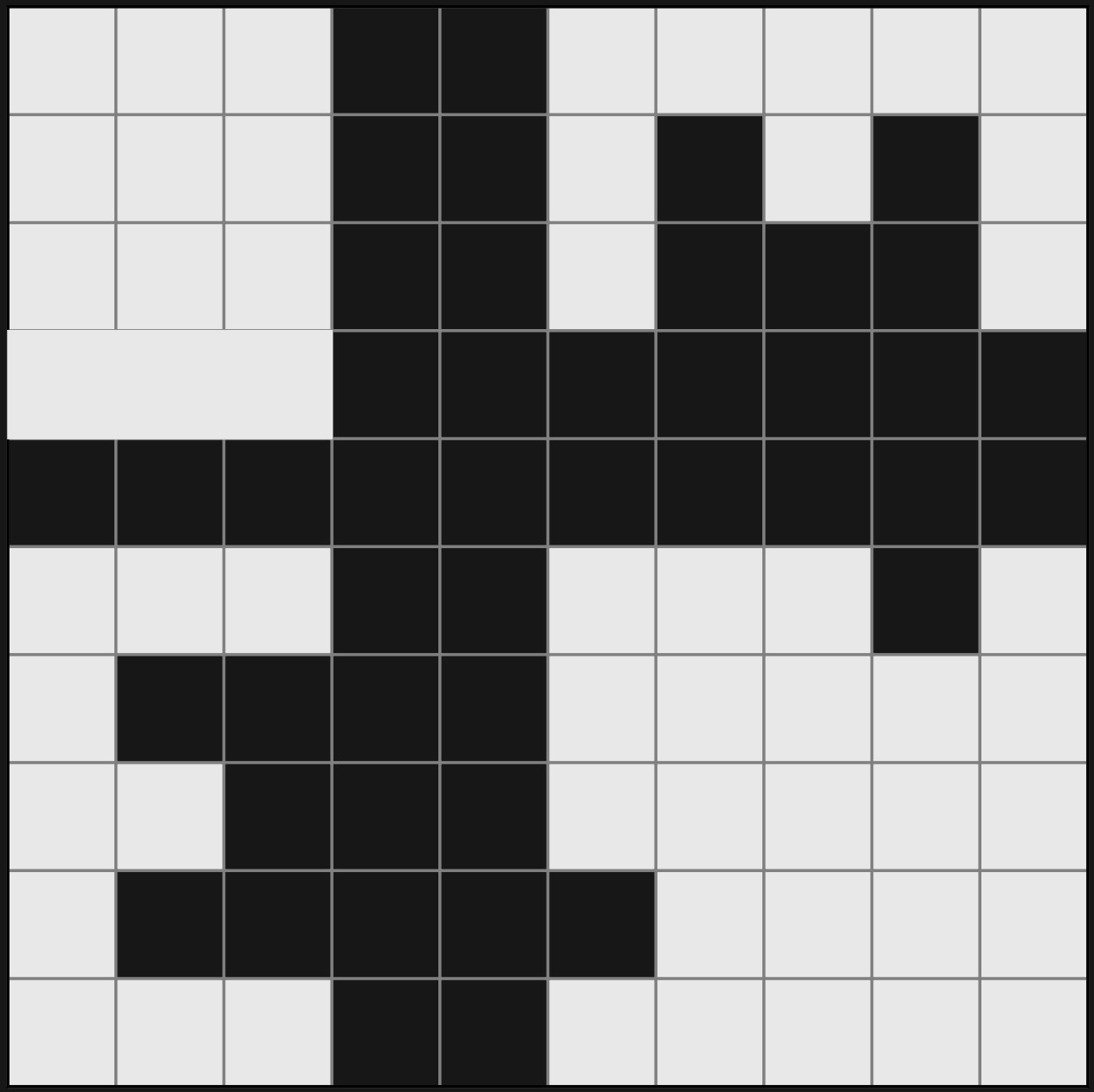
AND gate in f_2



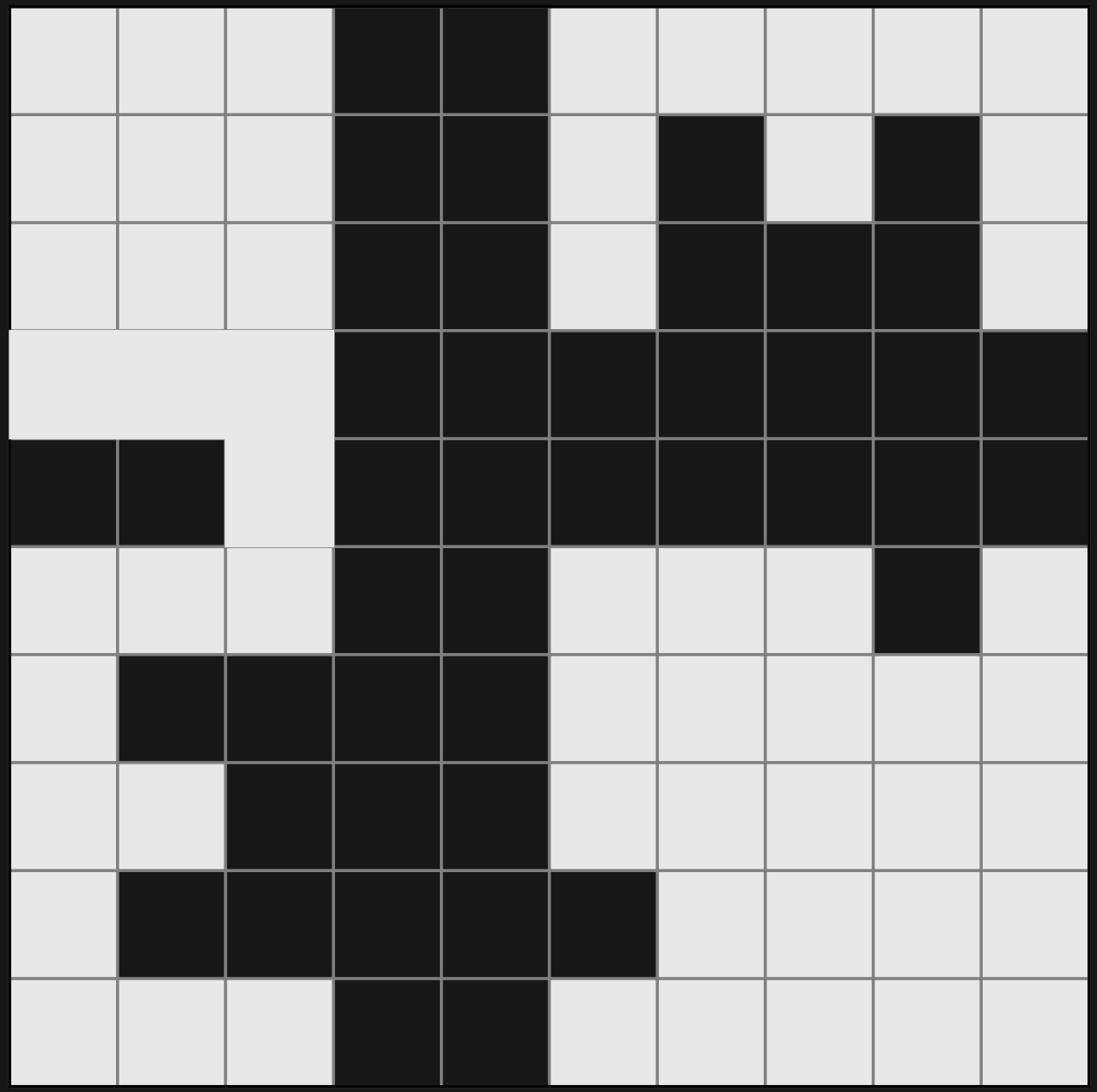
AND gate in f_2



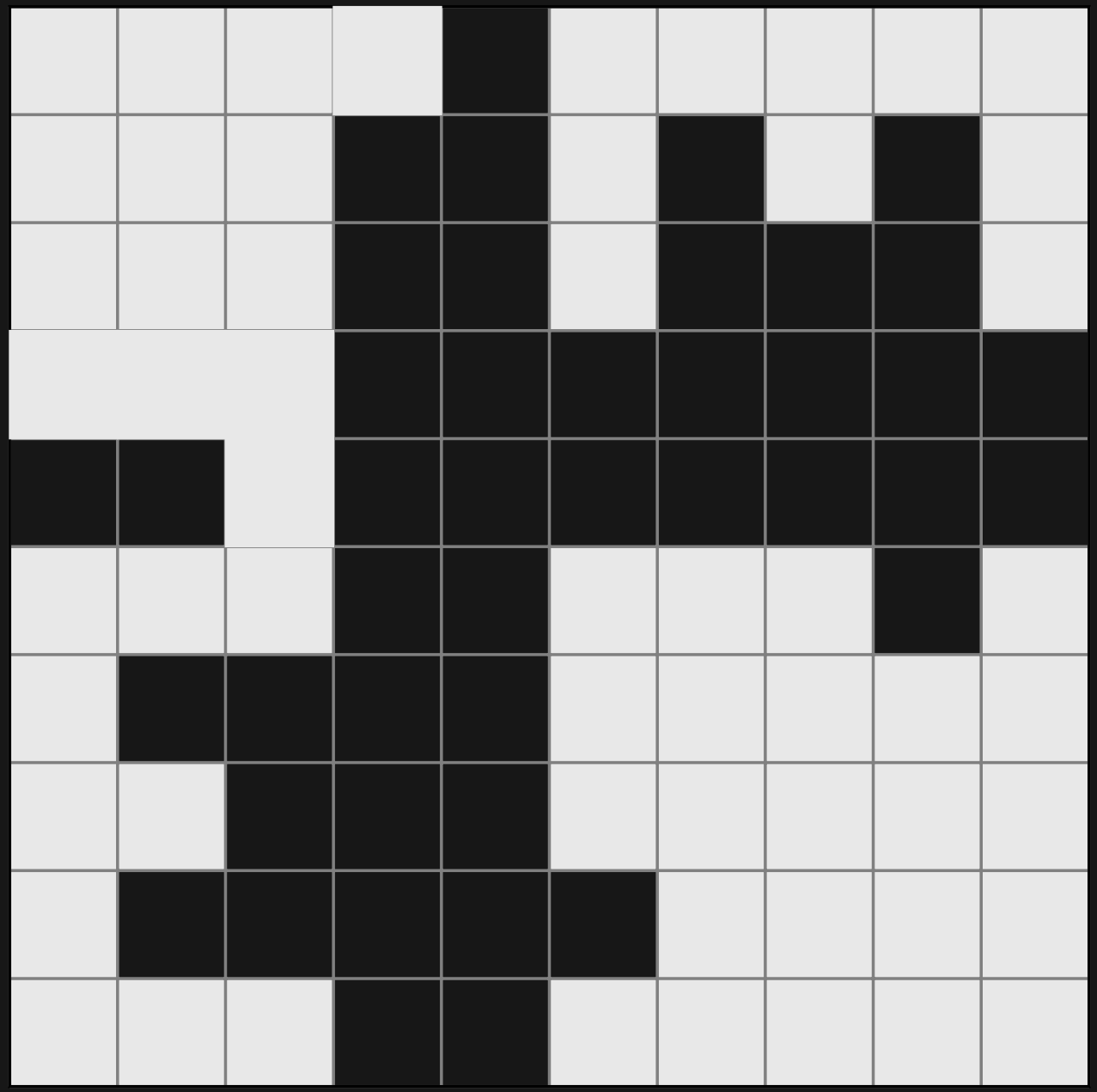
AND gate in f_2



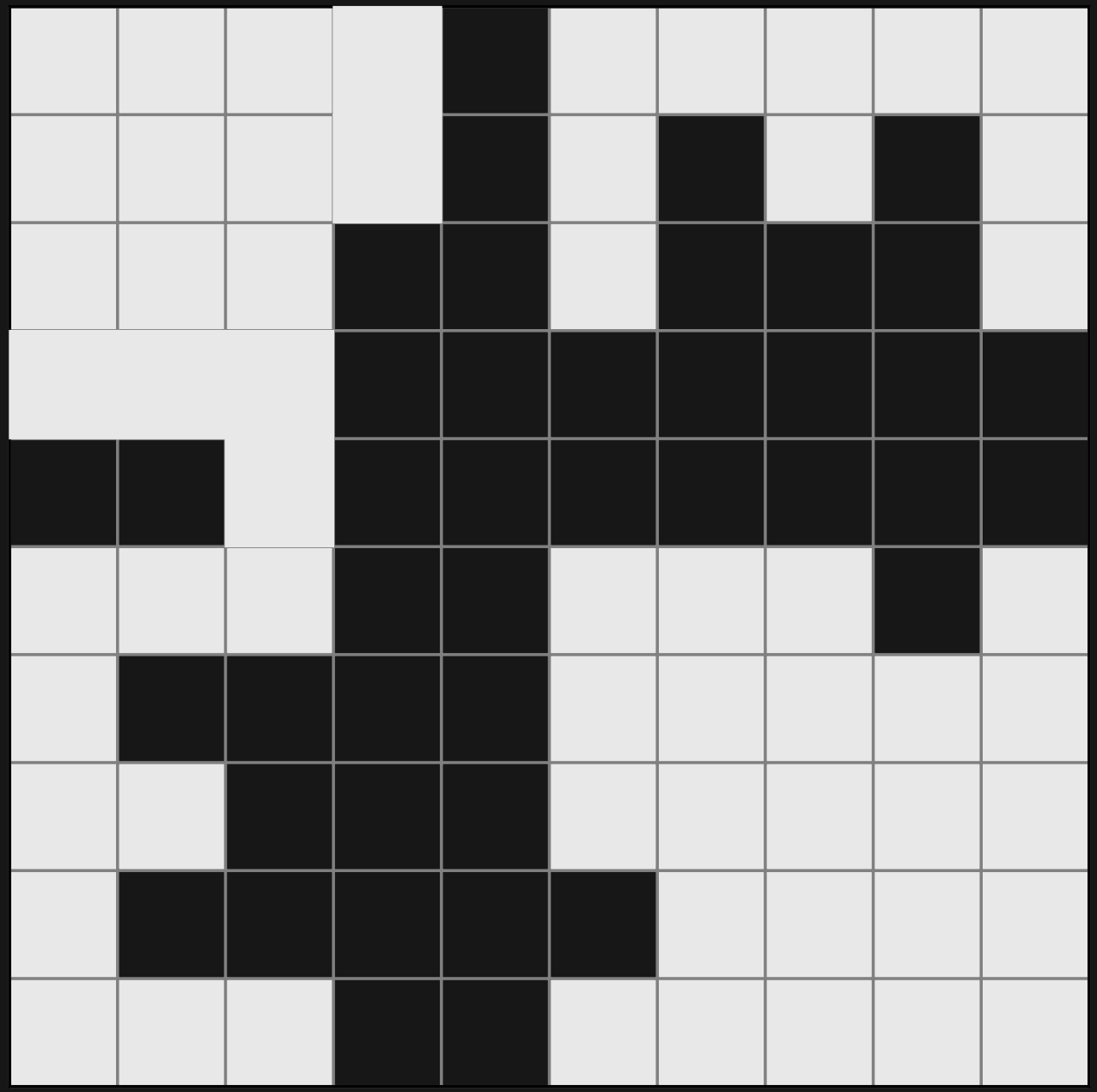
AND gate in f_2



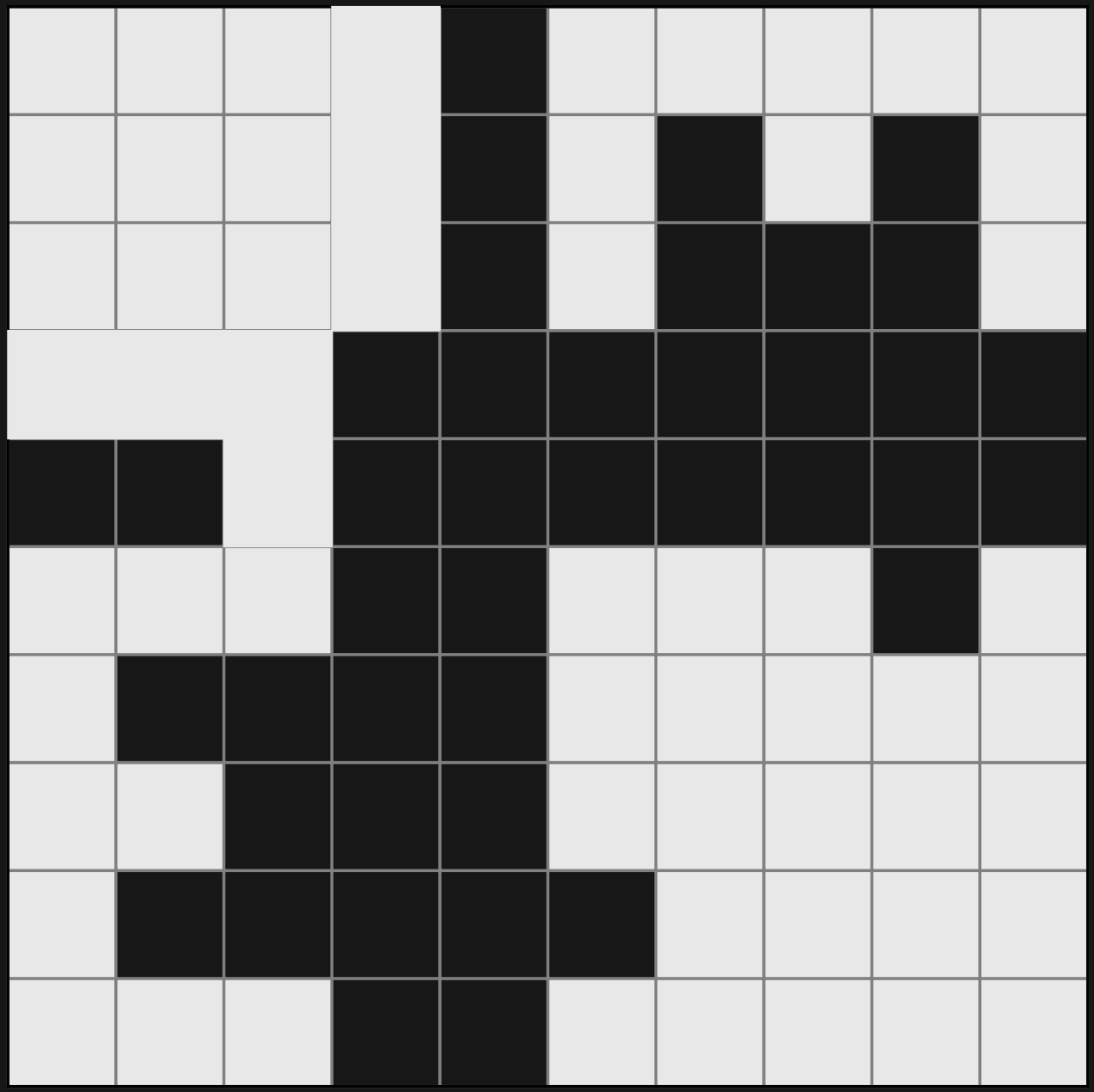
AND gate in f_2



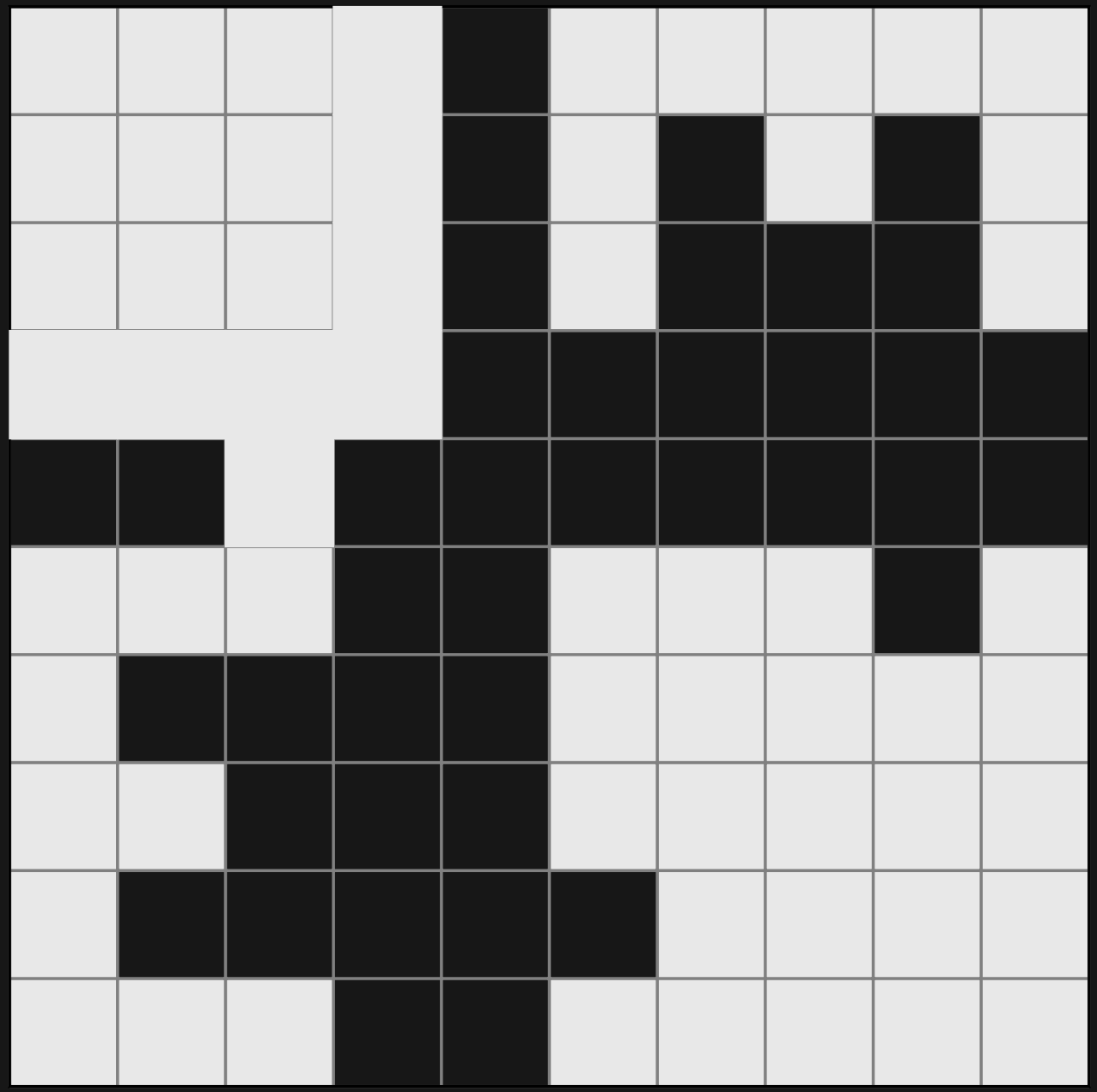
AND gate in f_2



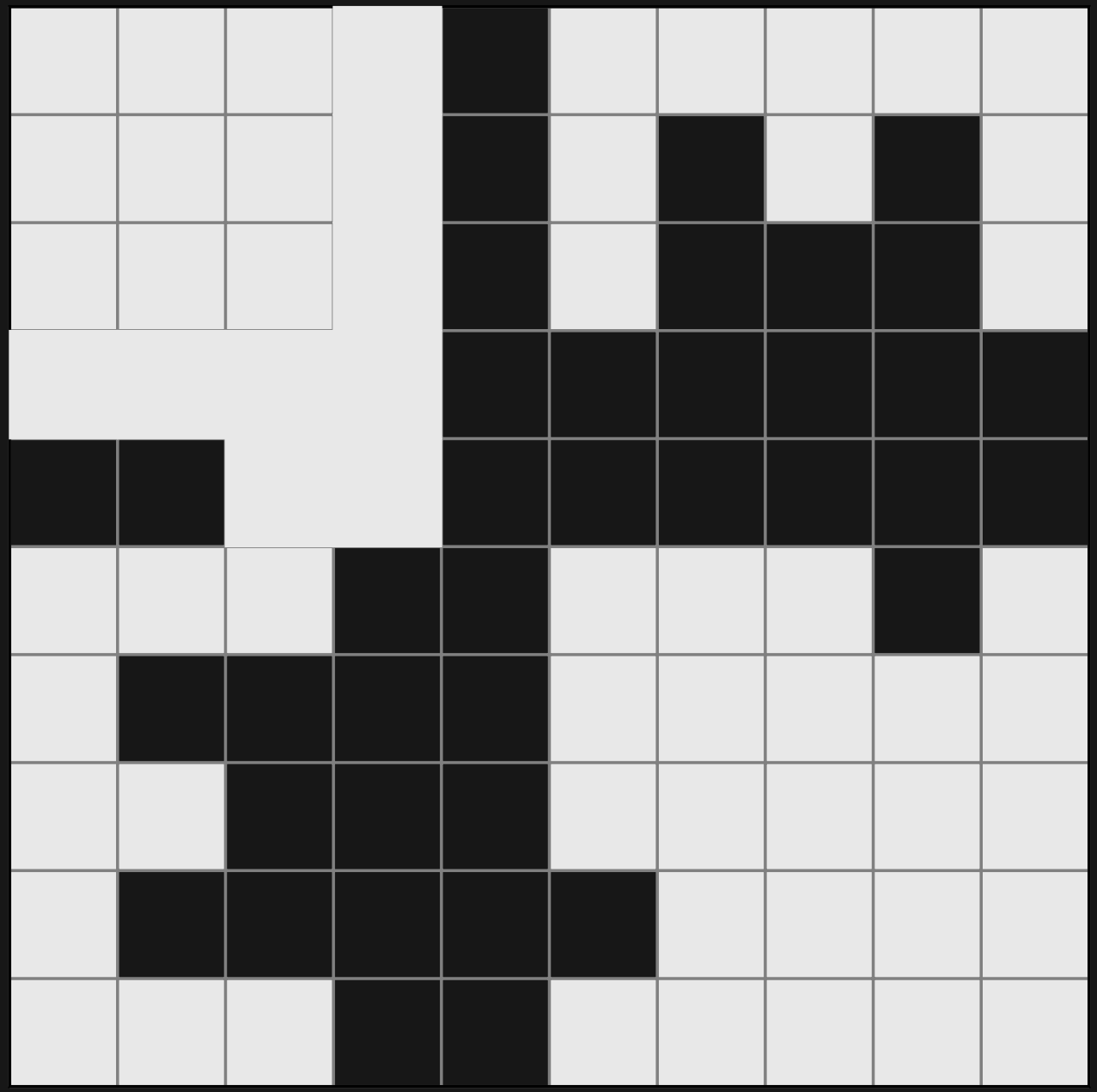
AND gate in f_2



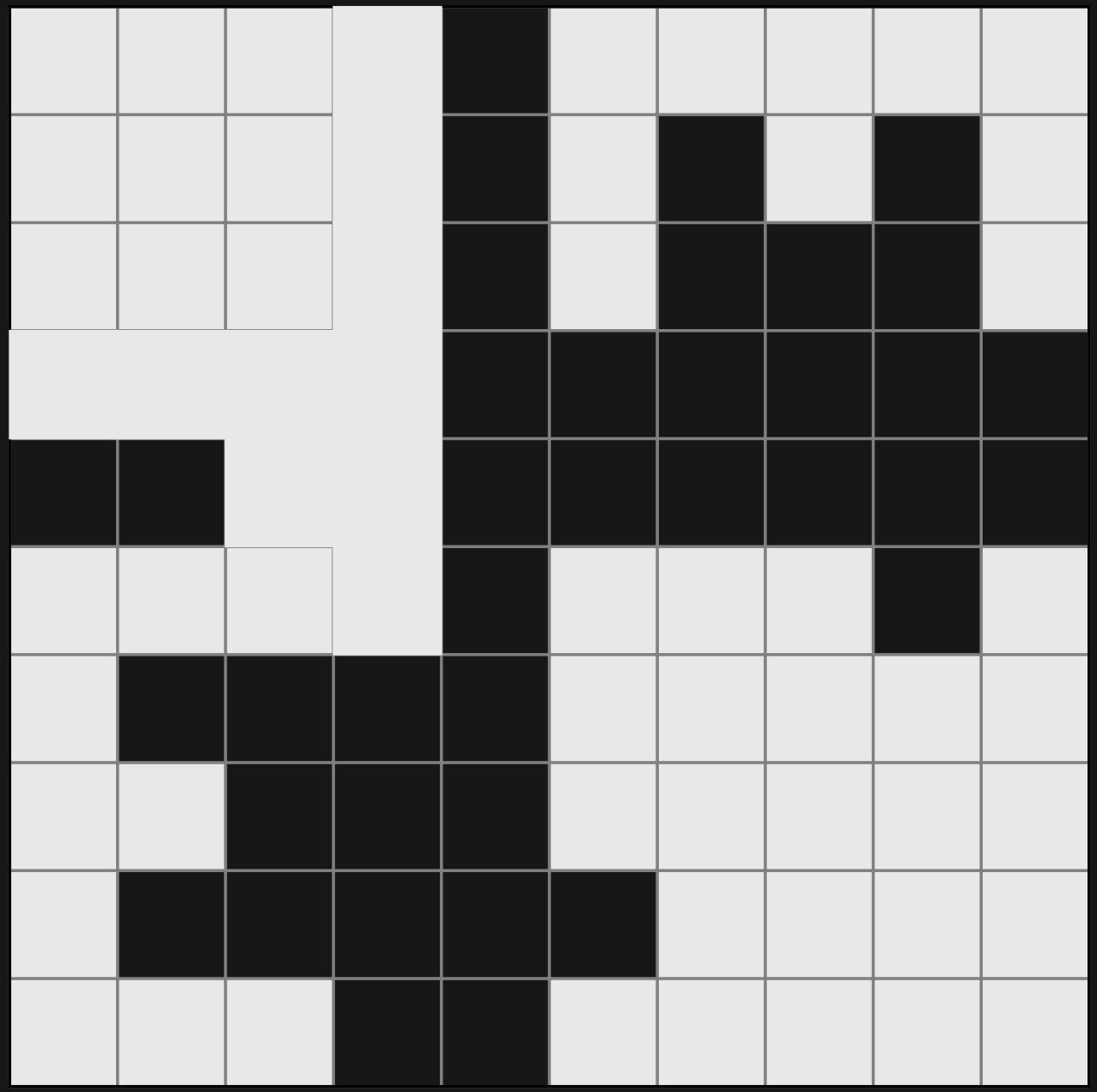
AND gate in f_2



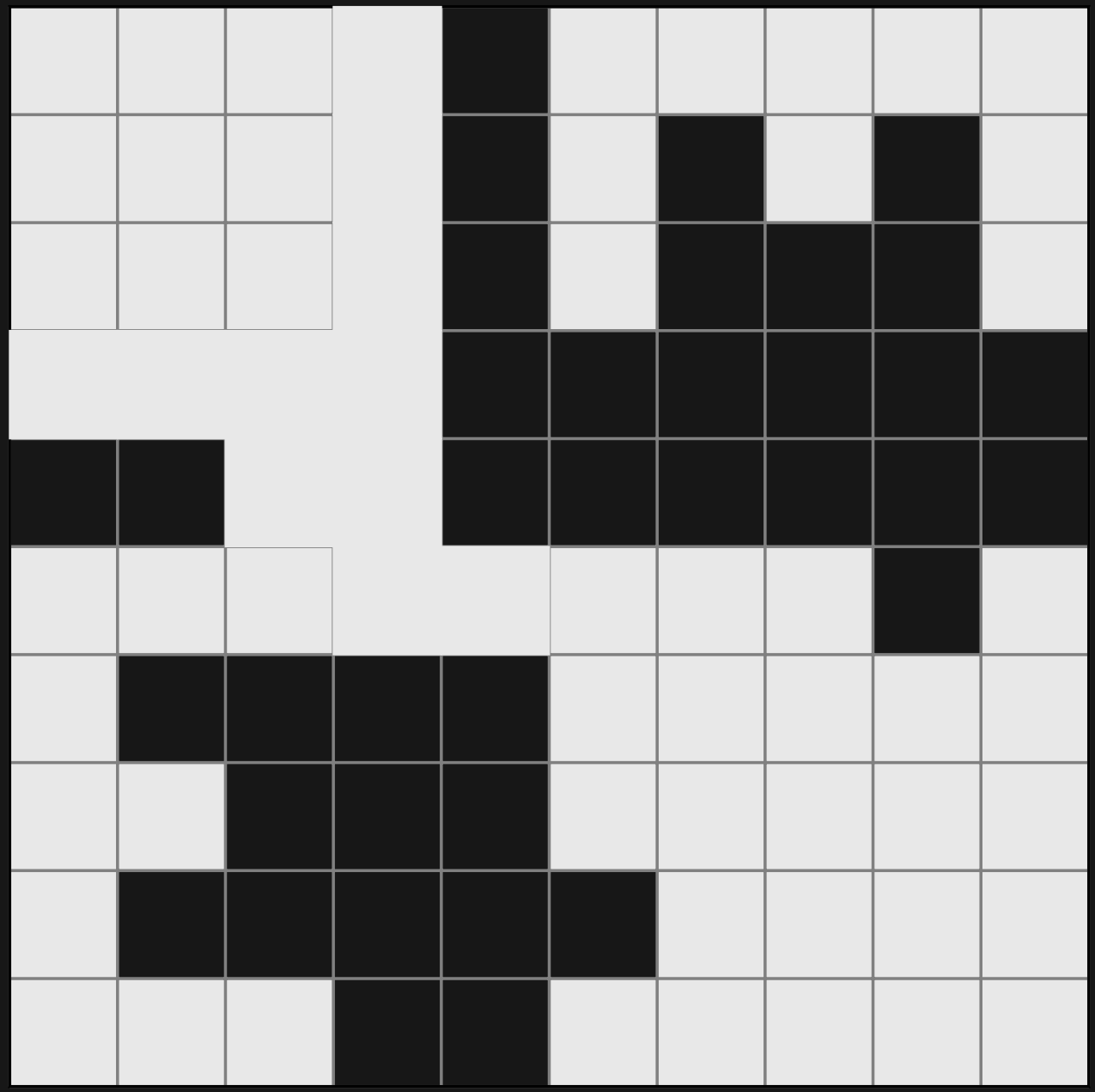
AND gate in f_2



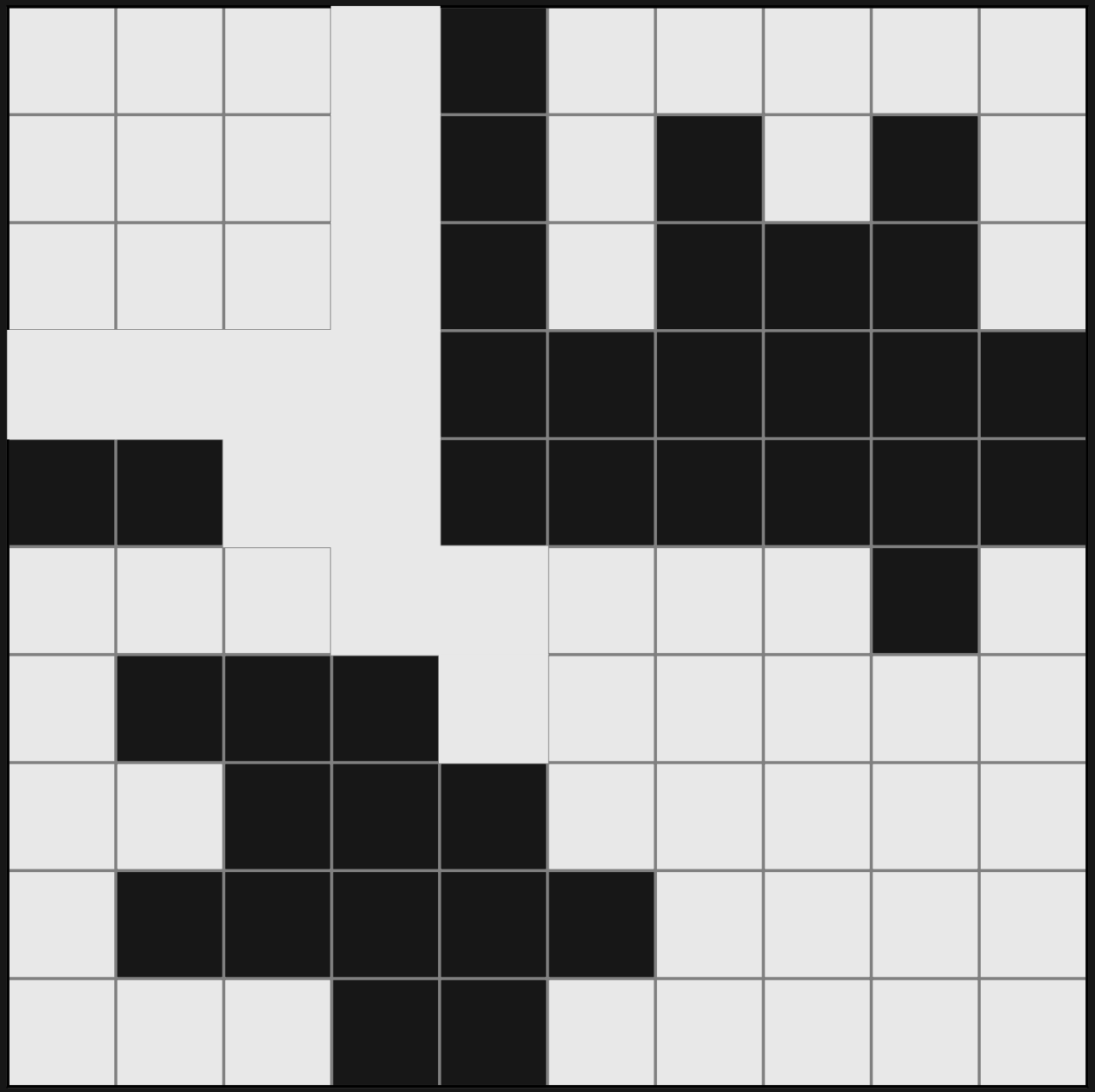
AND gate in f_2



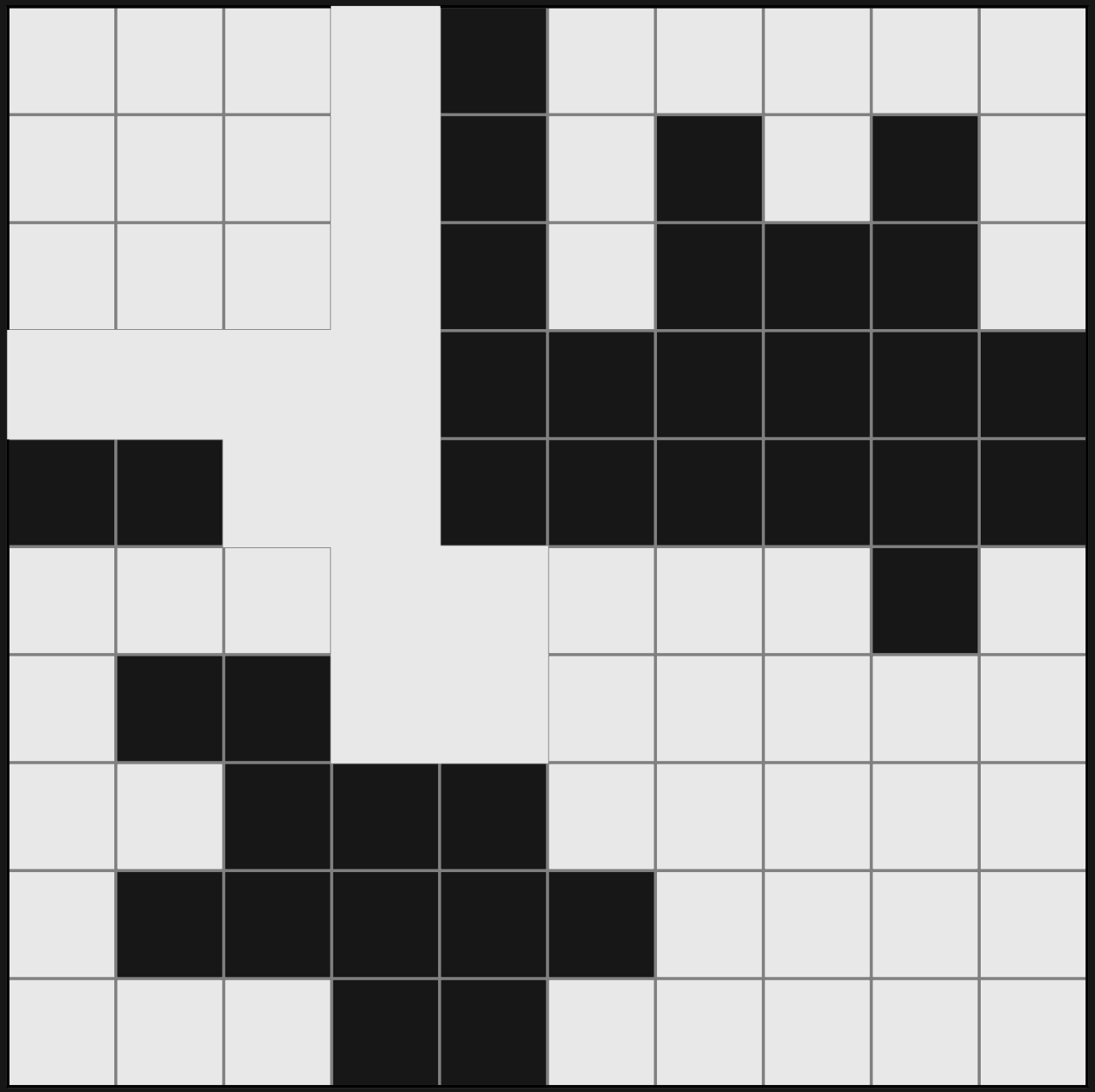
AND gate in f_2



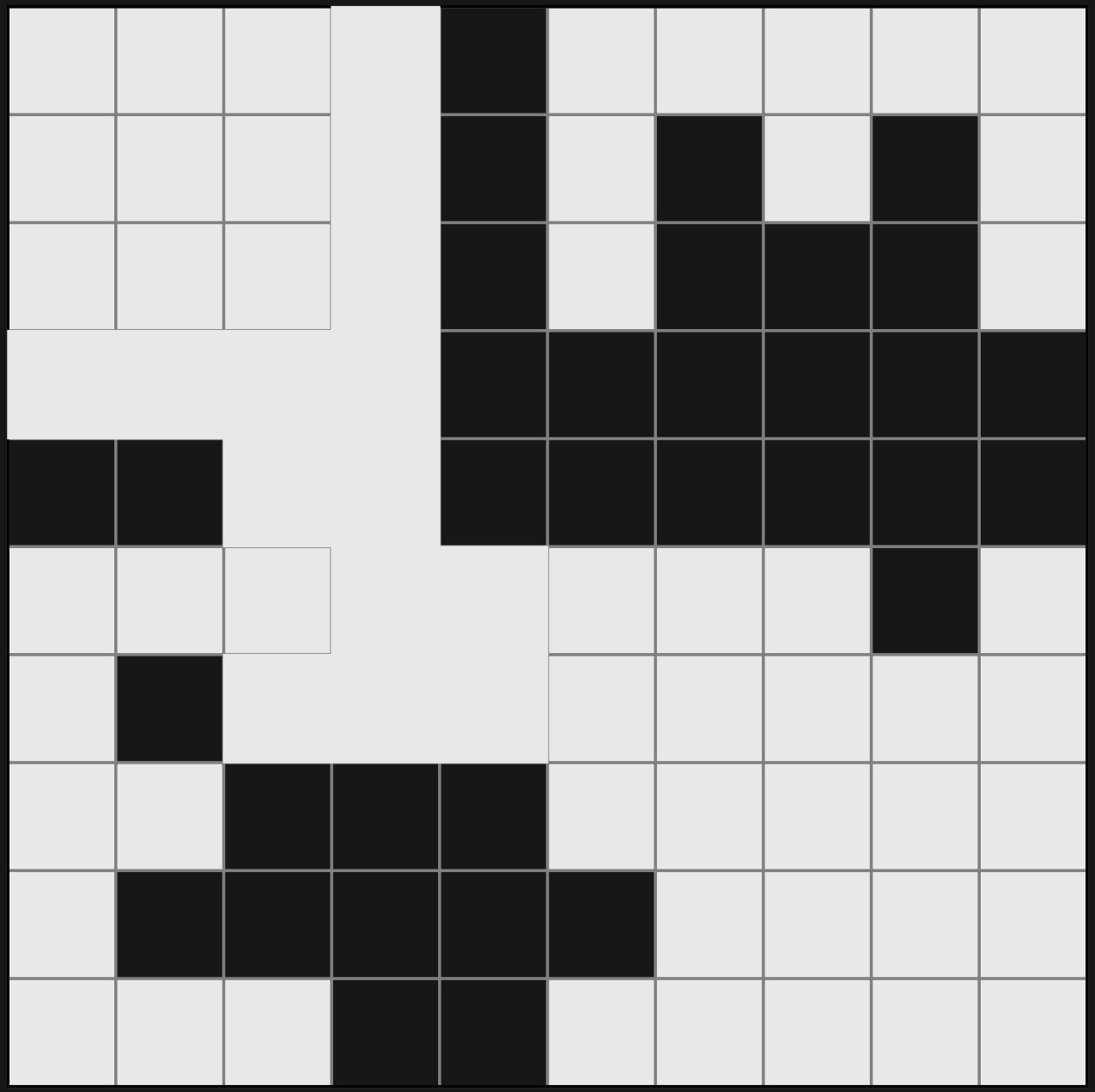
AND gate in f_2



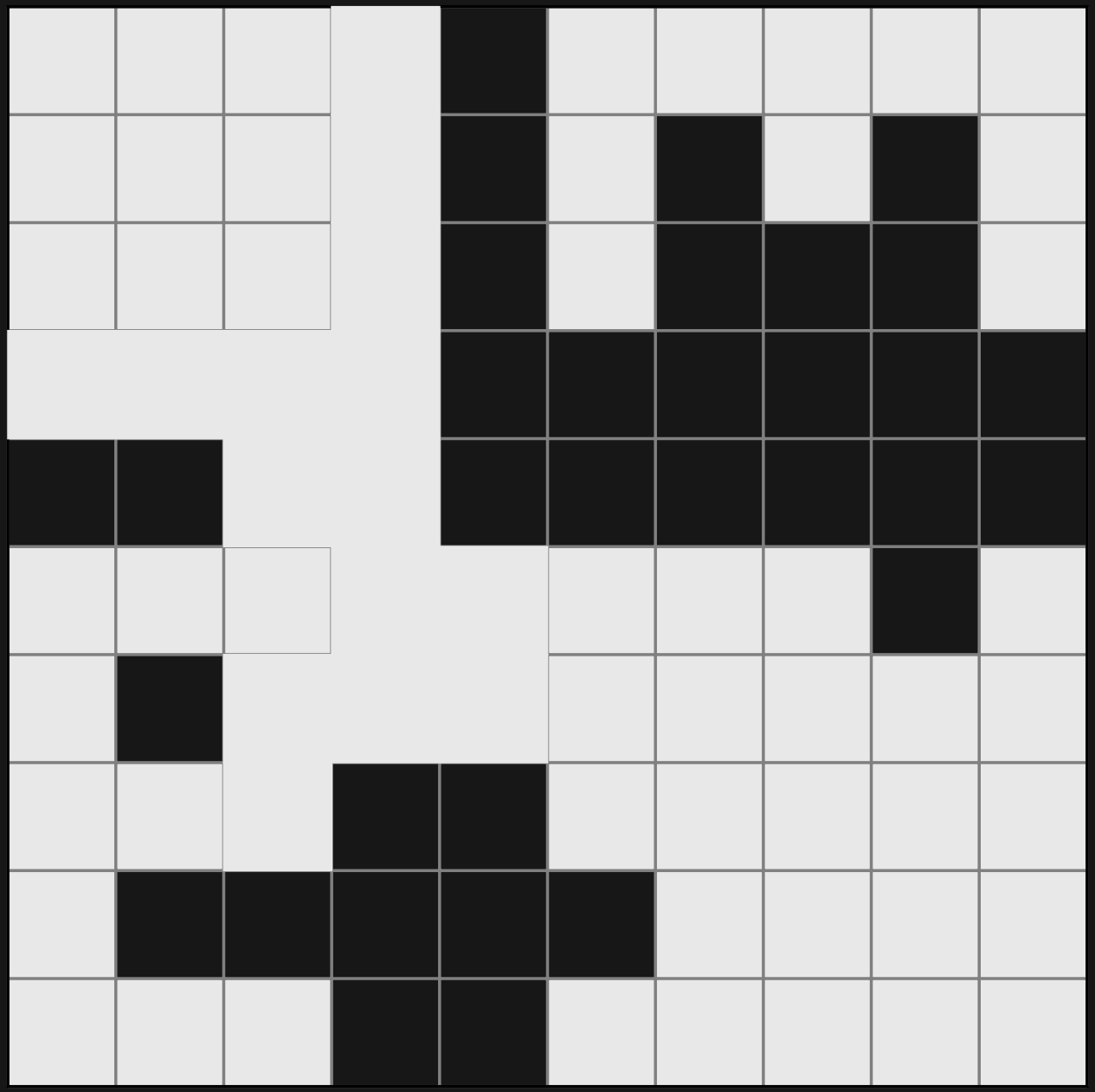
AND gate in f_2



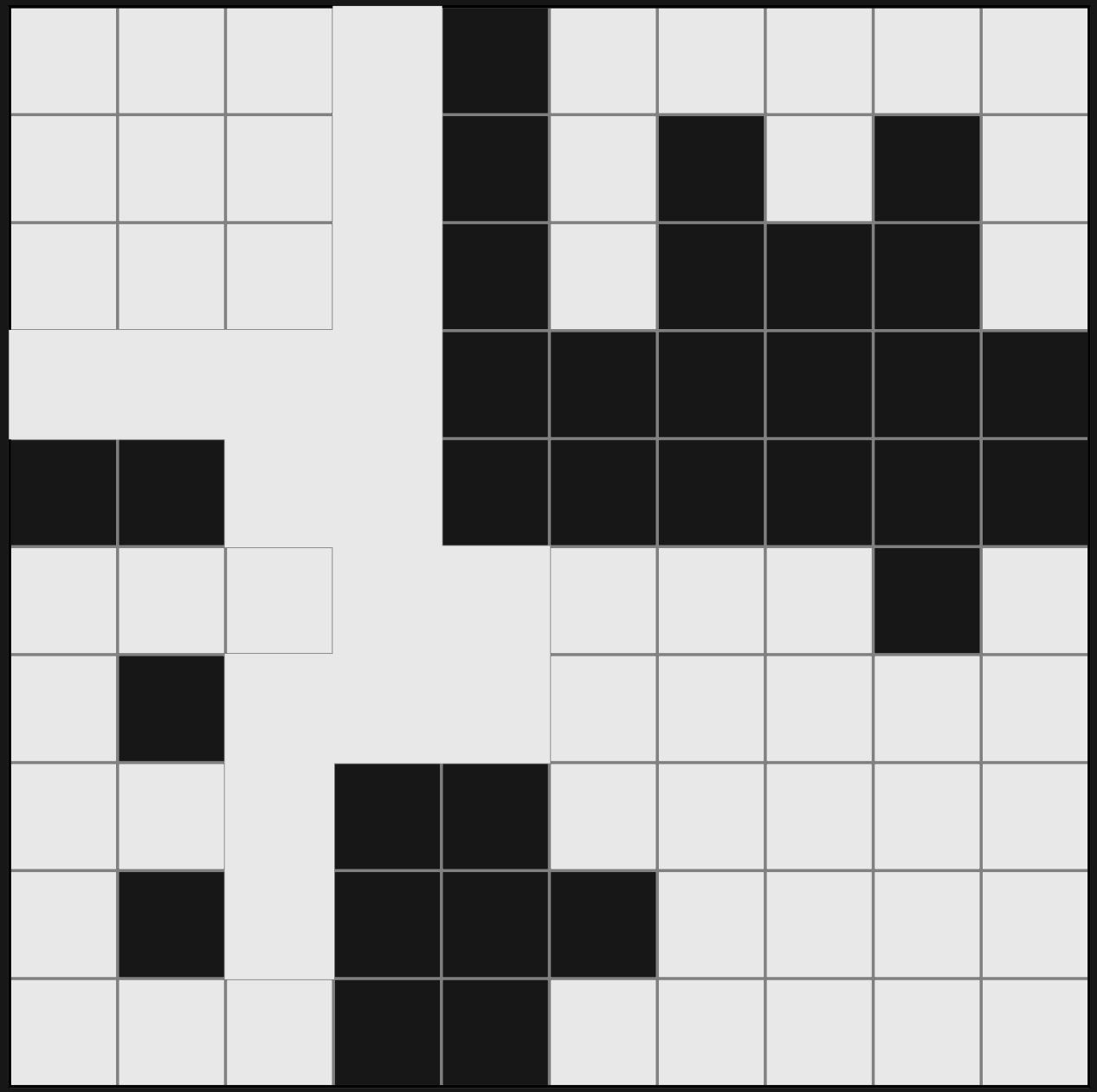
AND gate in f_2



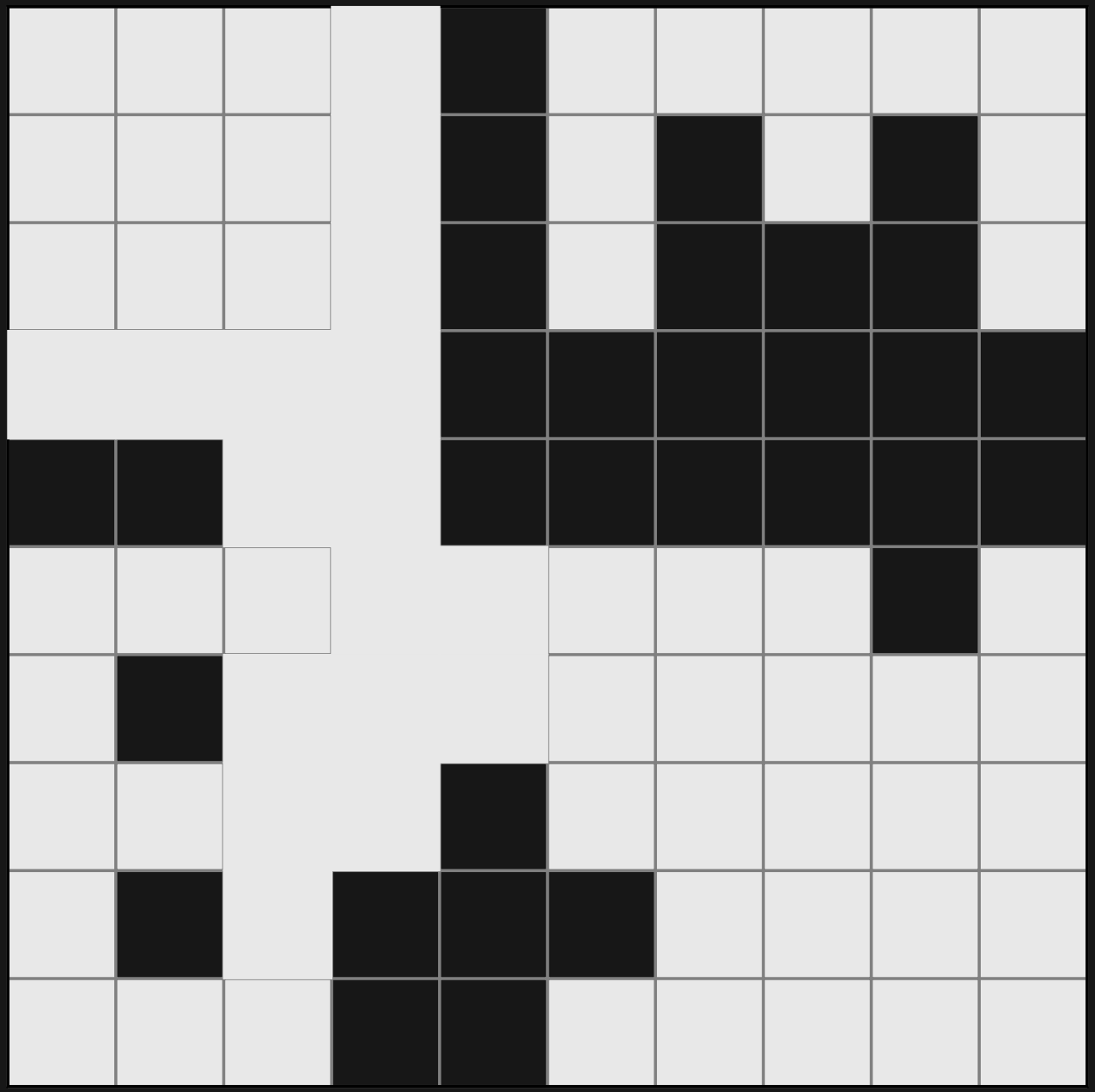
AND gate in f_2



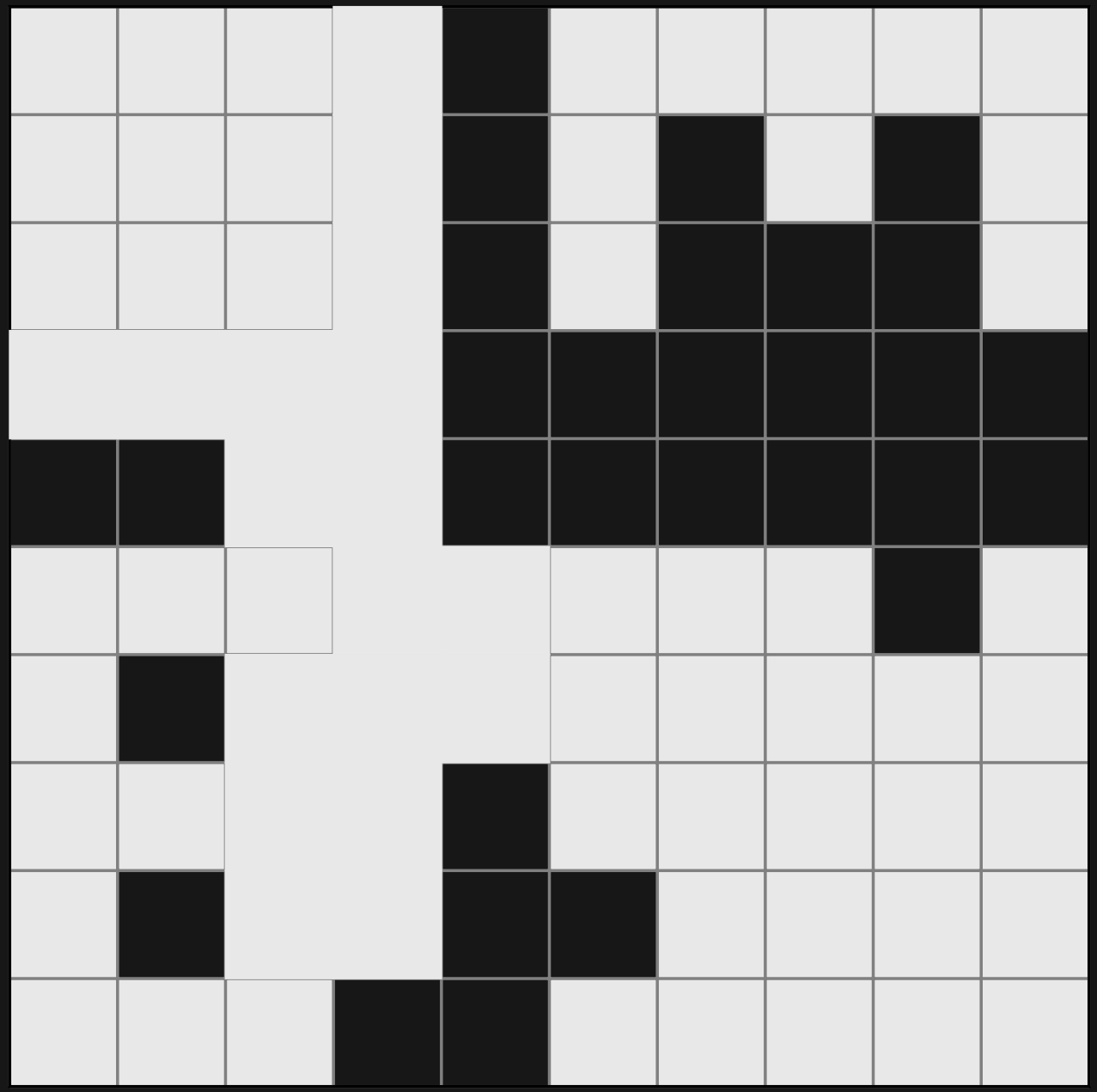
AND gate in f_2



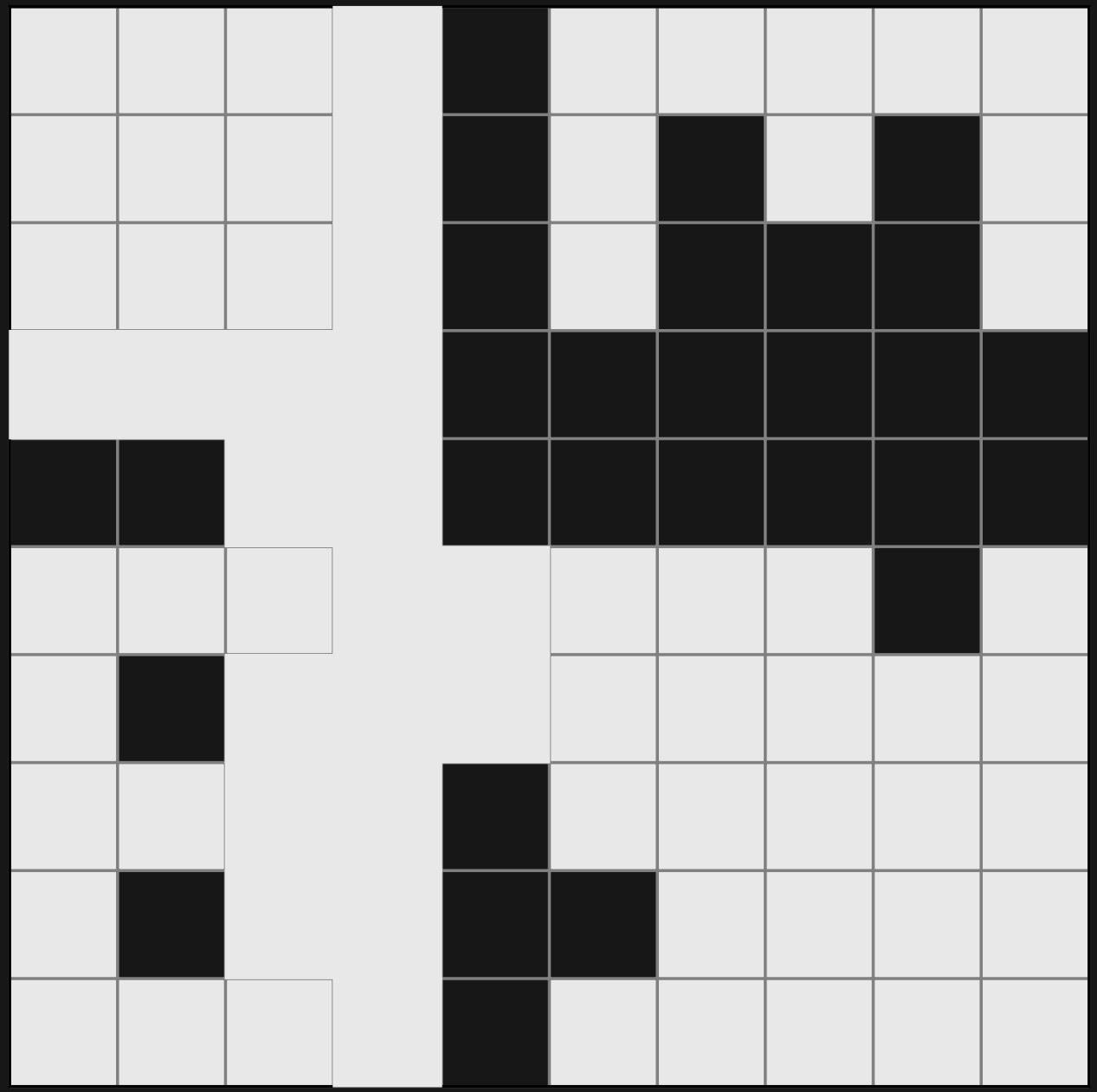
AND gate in f_2



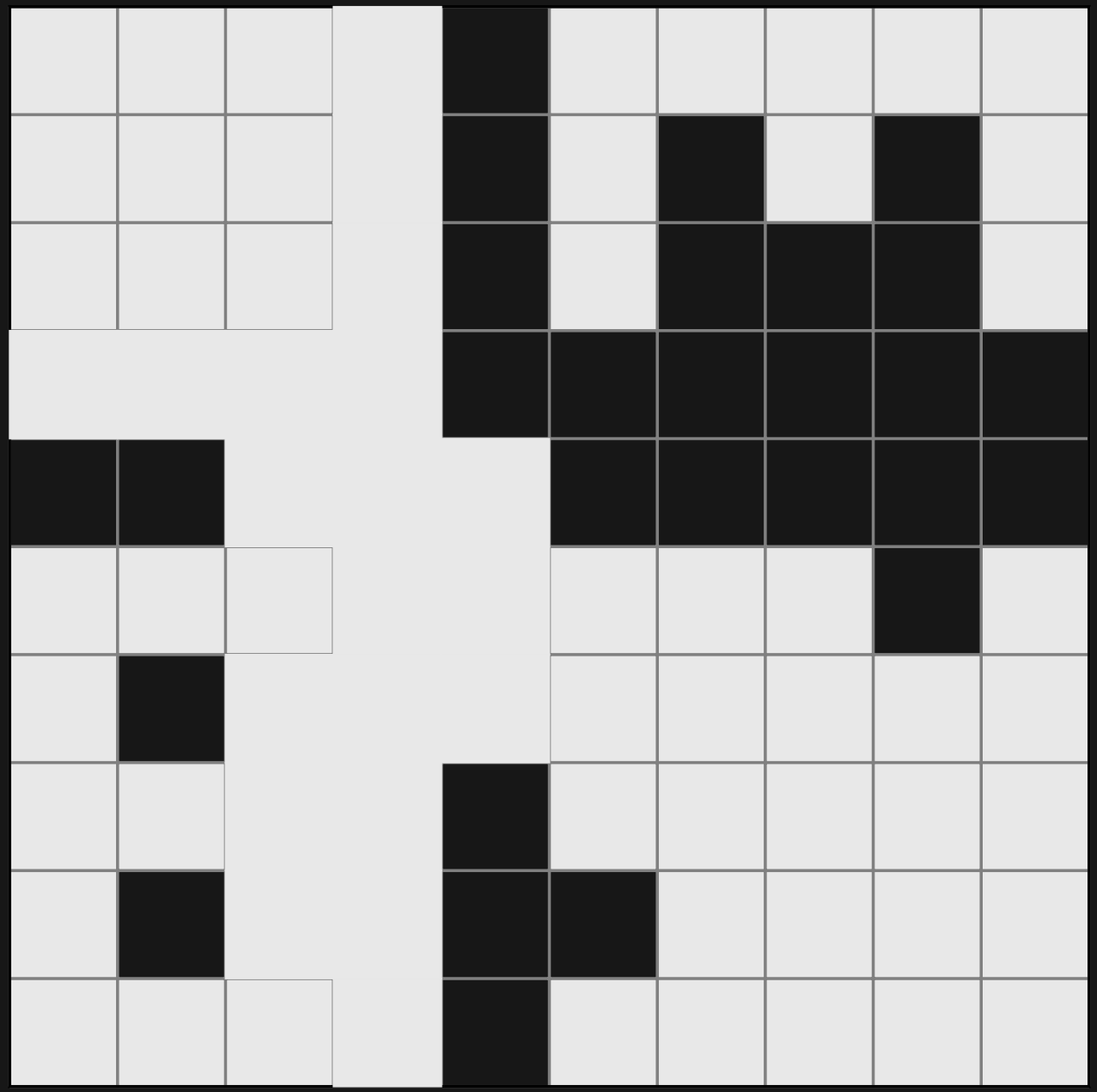
AND gate in f_2



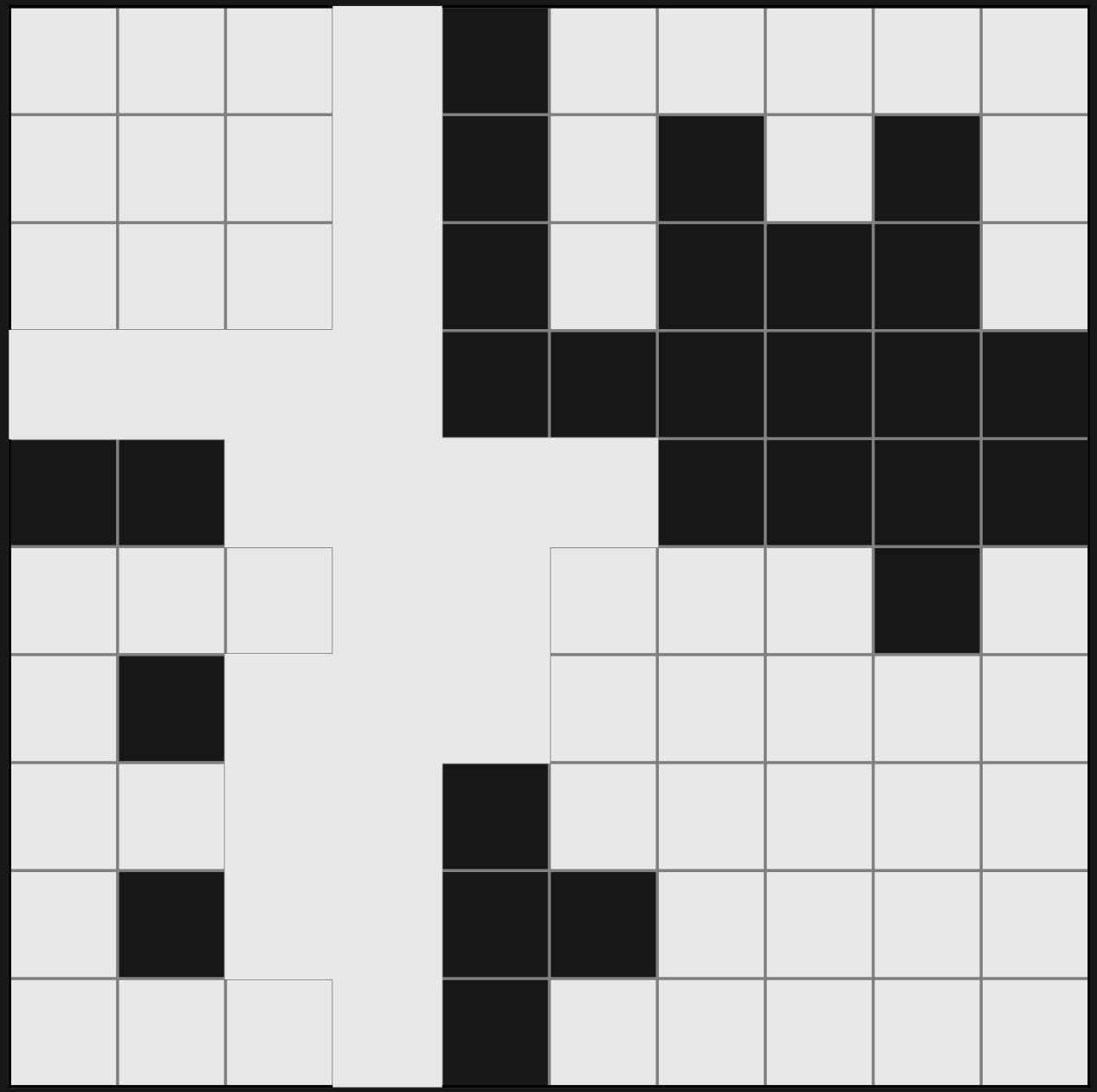
AND gate in f_2



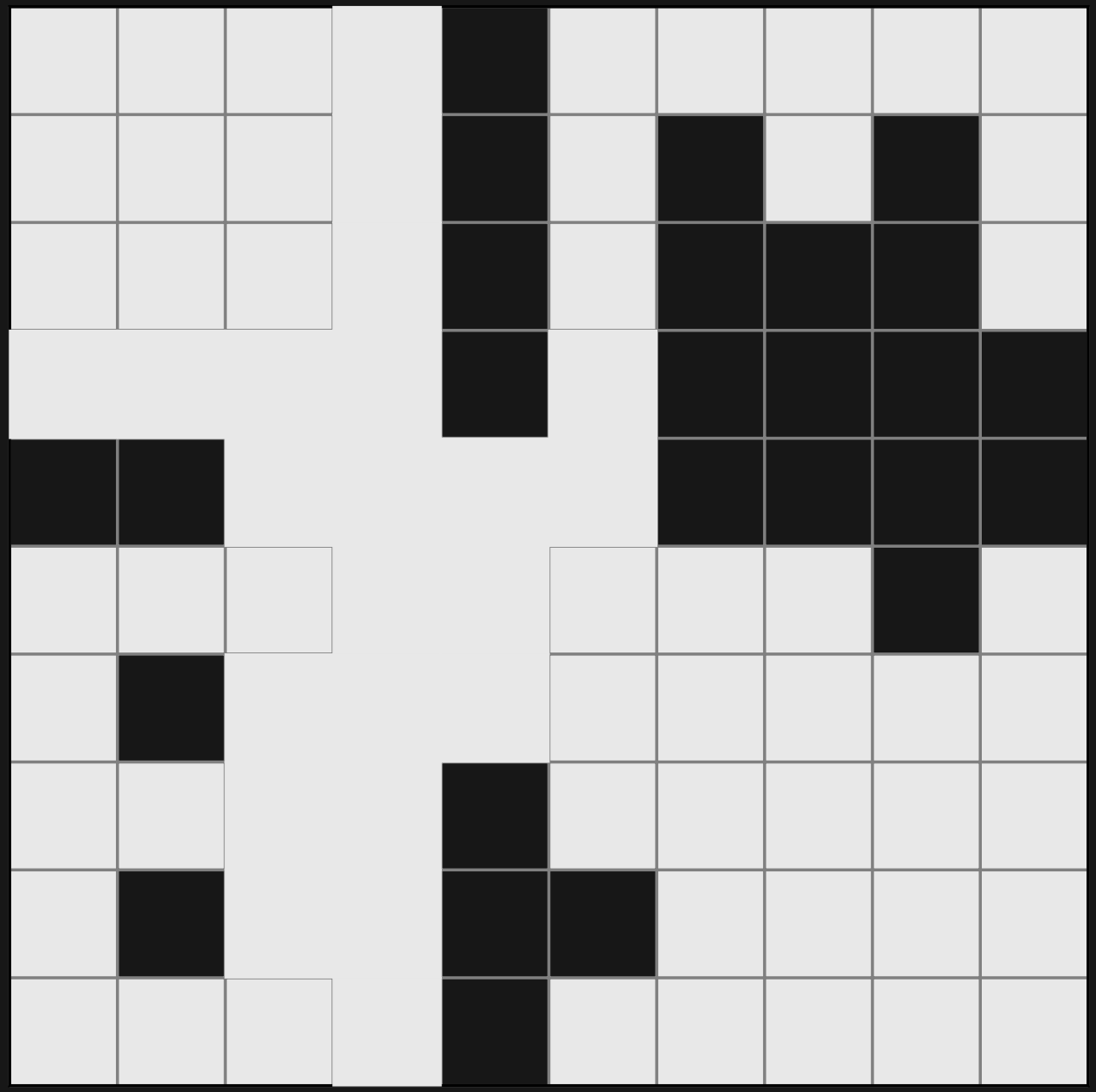
AND gate in f_2



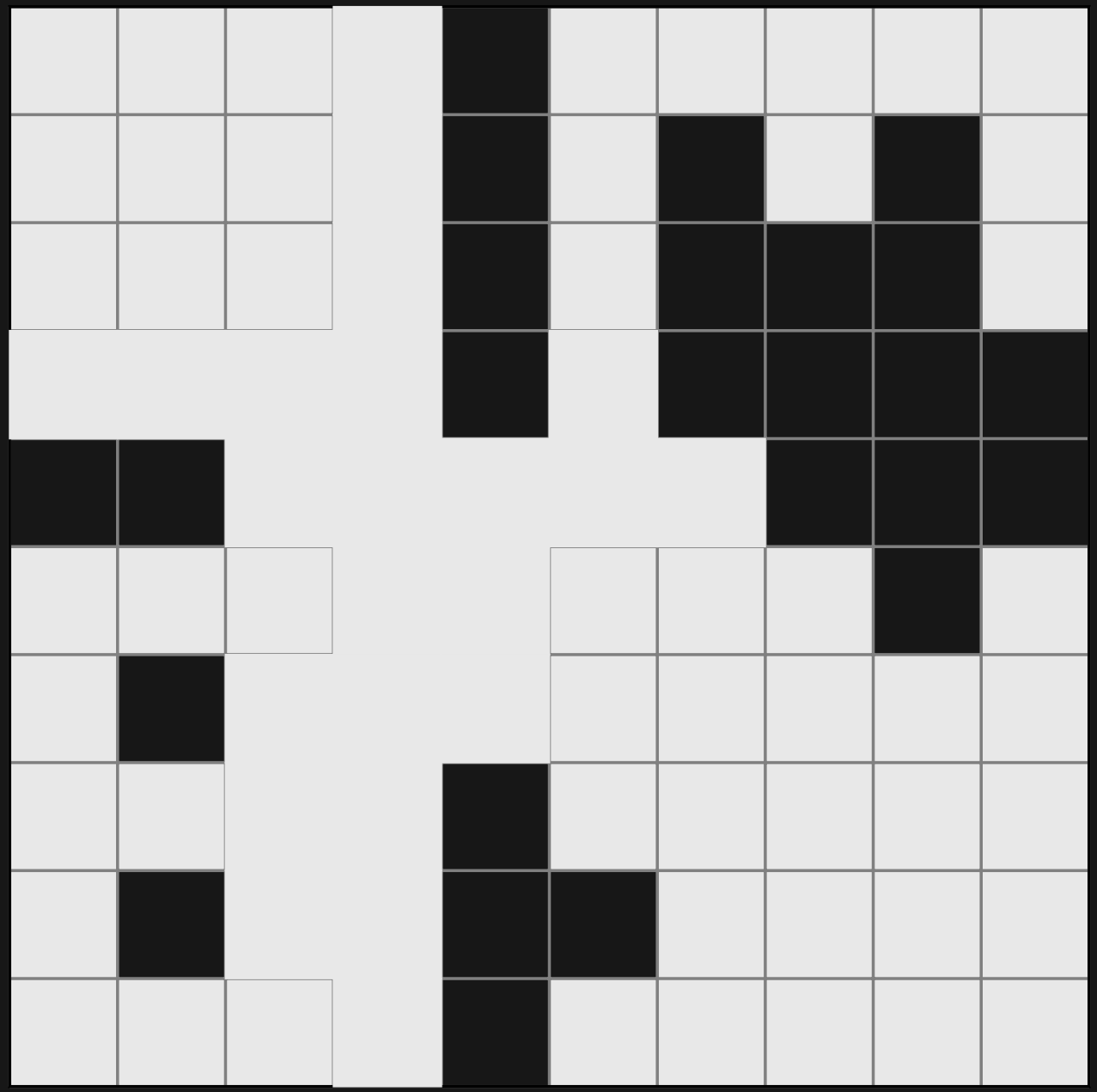
AND gate in f_2



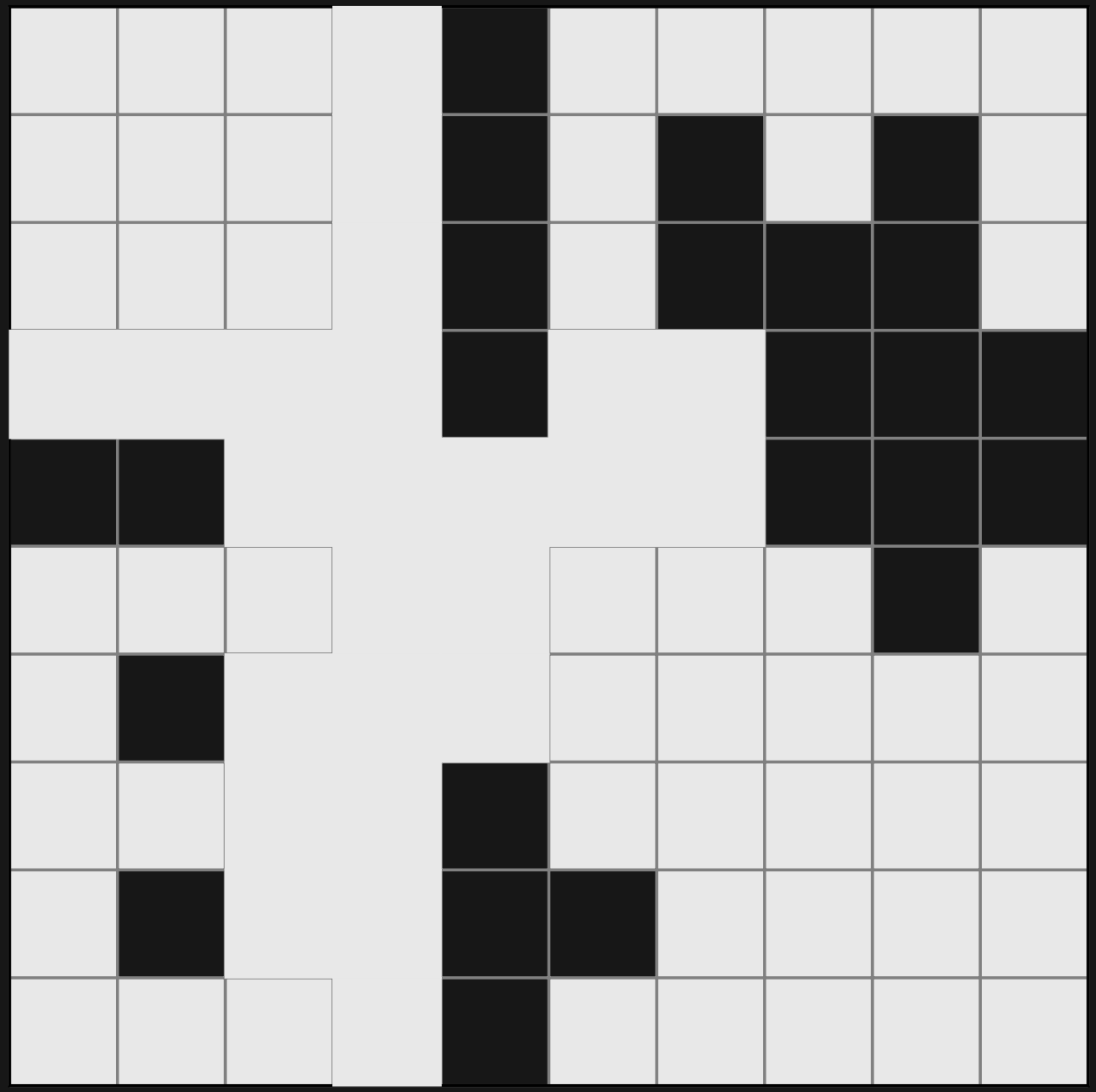
AND gate in f_2



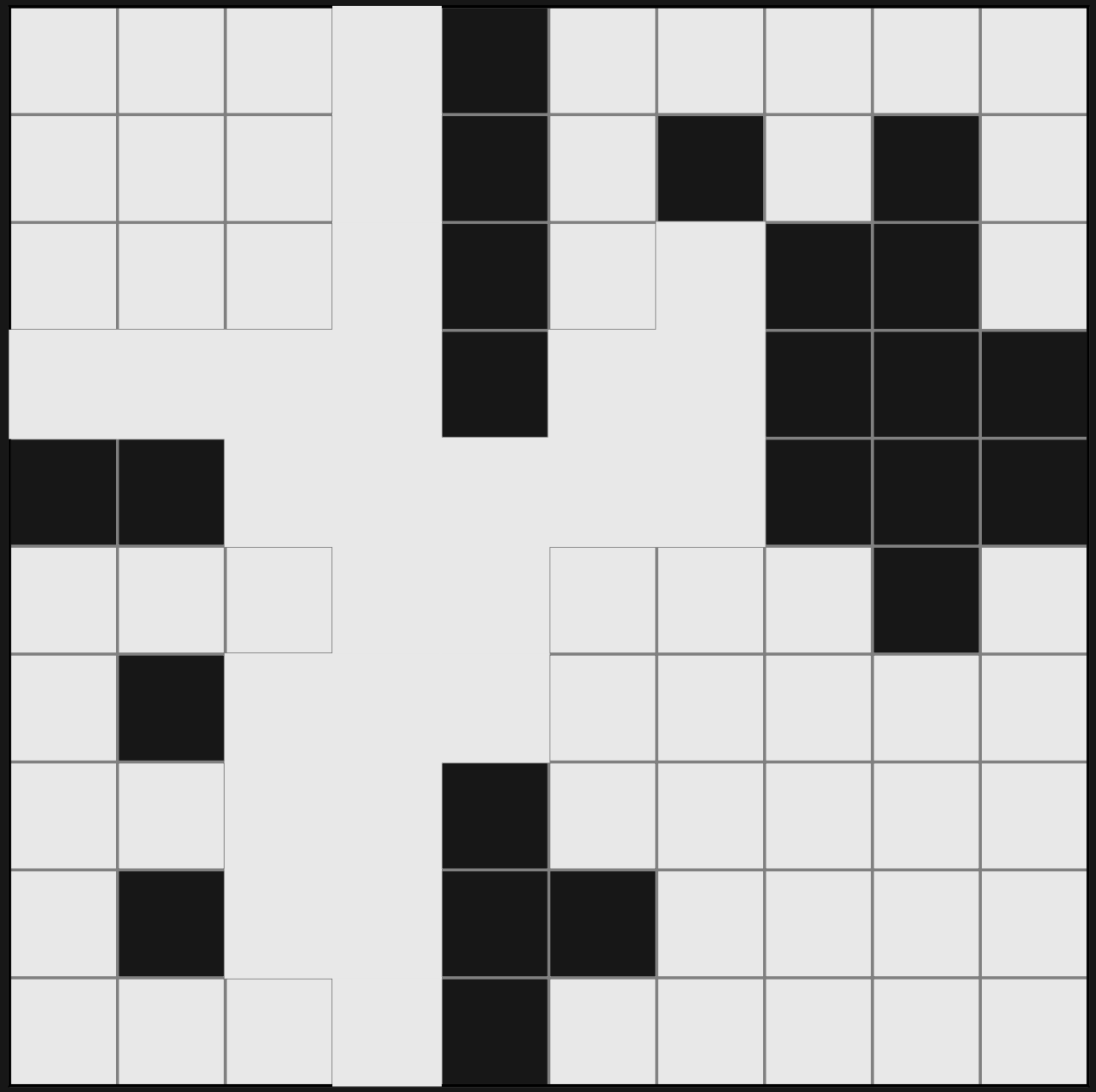
AND gate in f_2



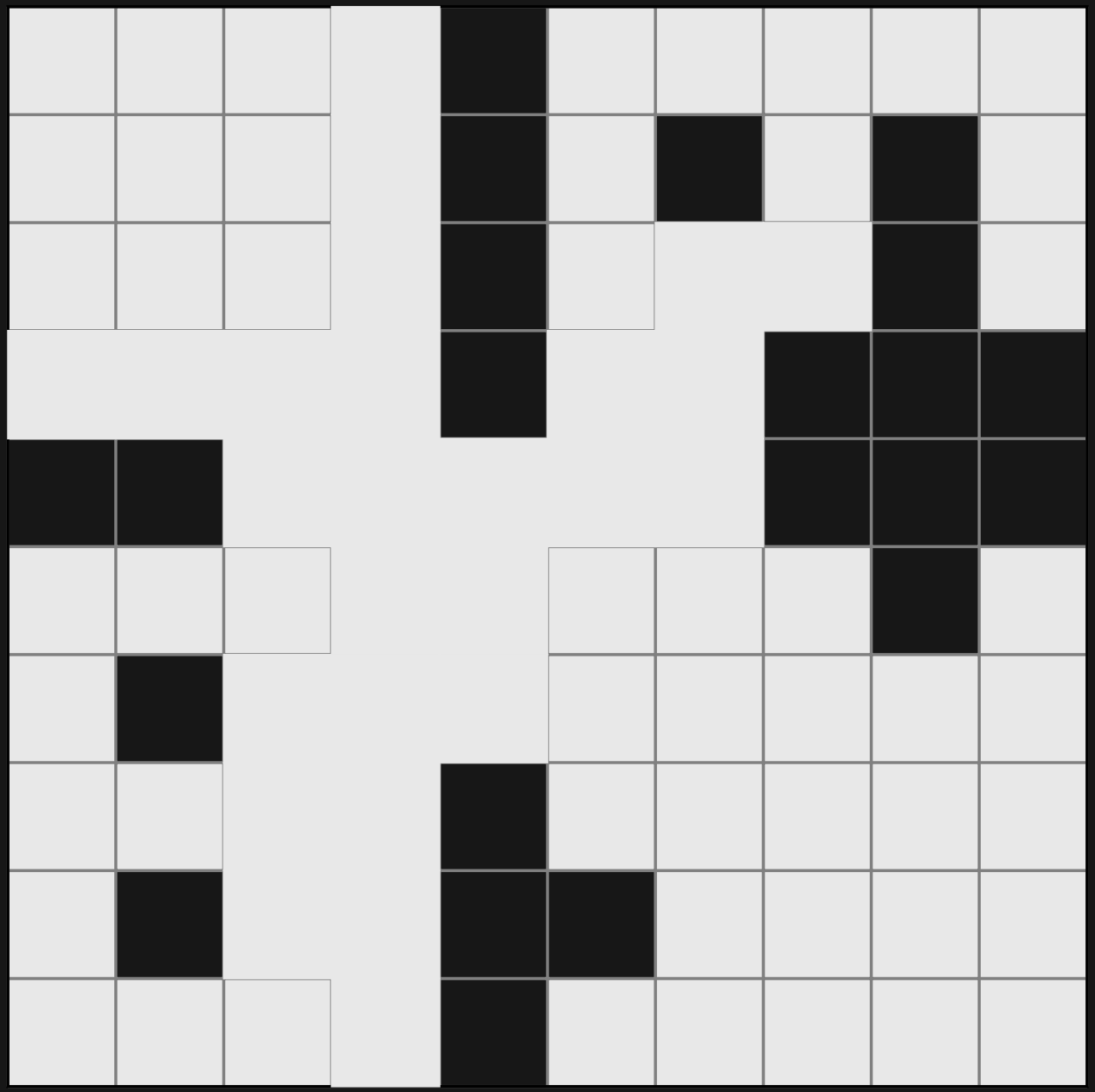
AND gate in f_2



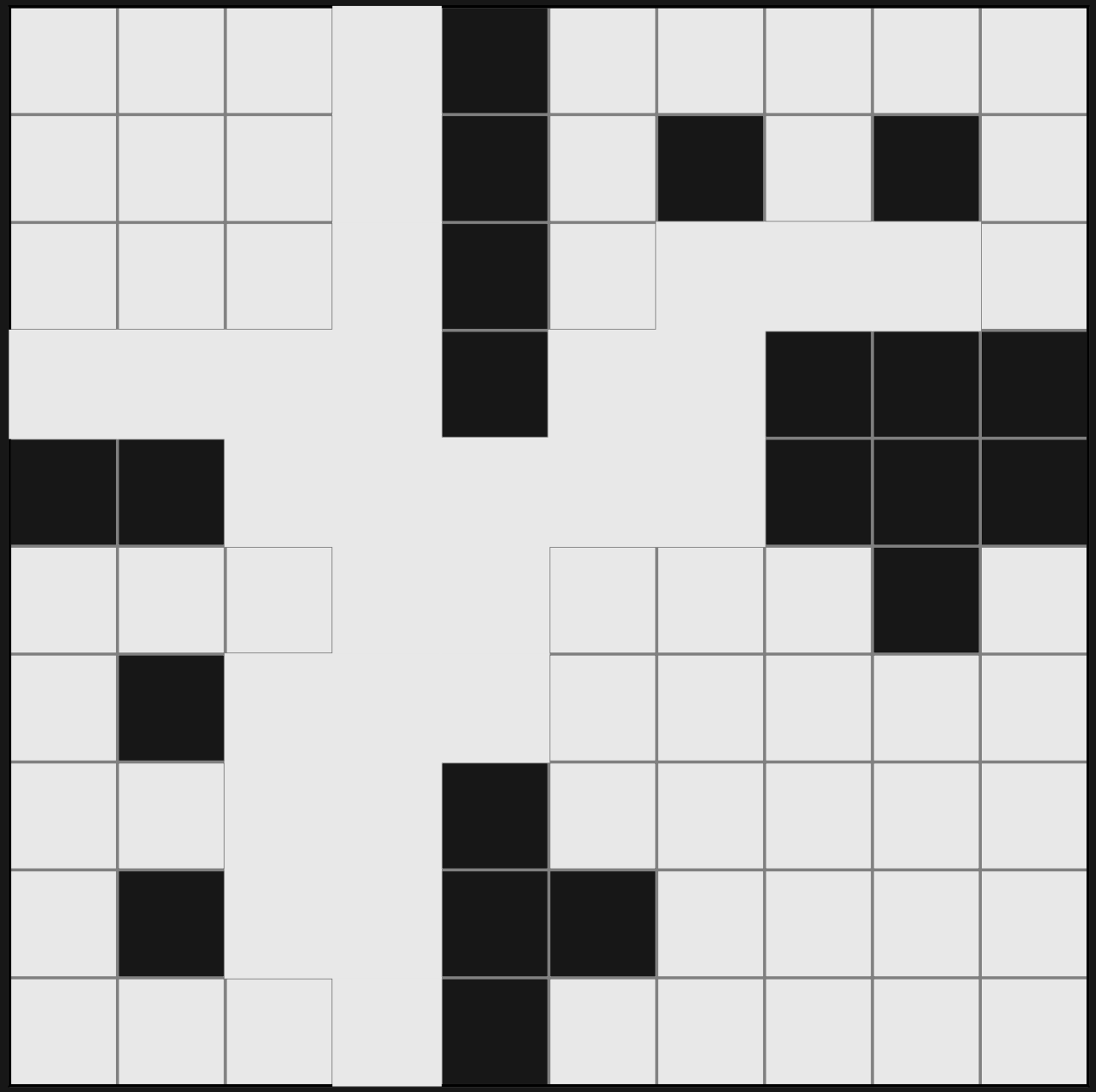
AND gate in f_2



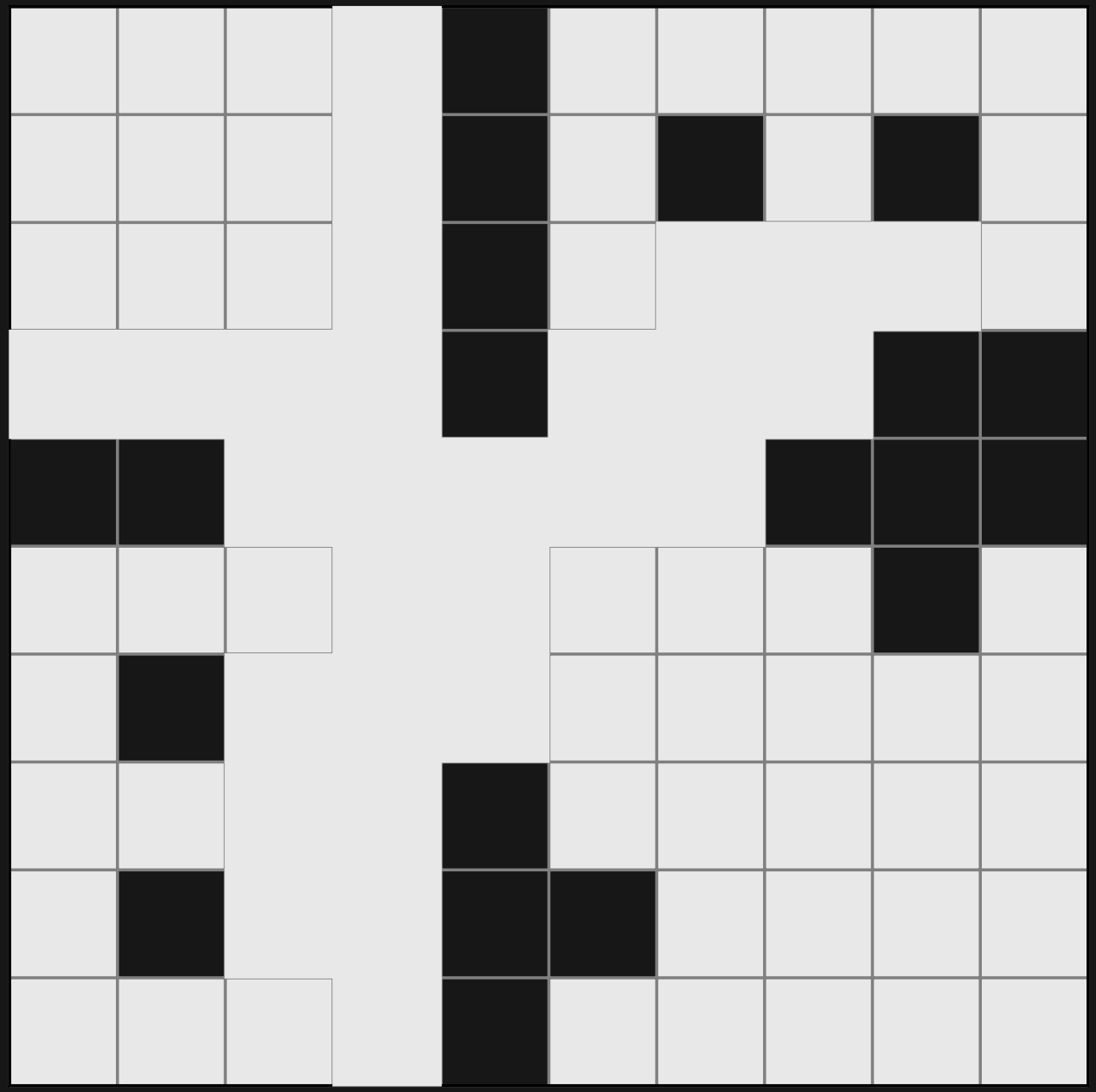
AND gate in f_2



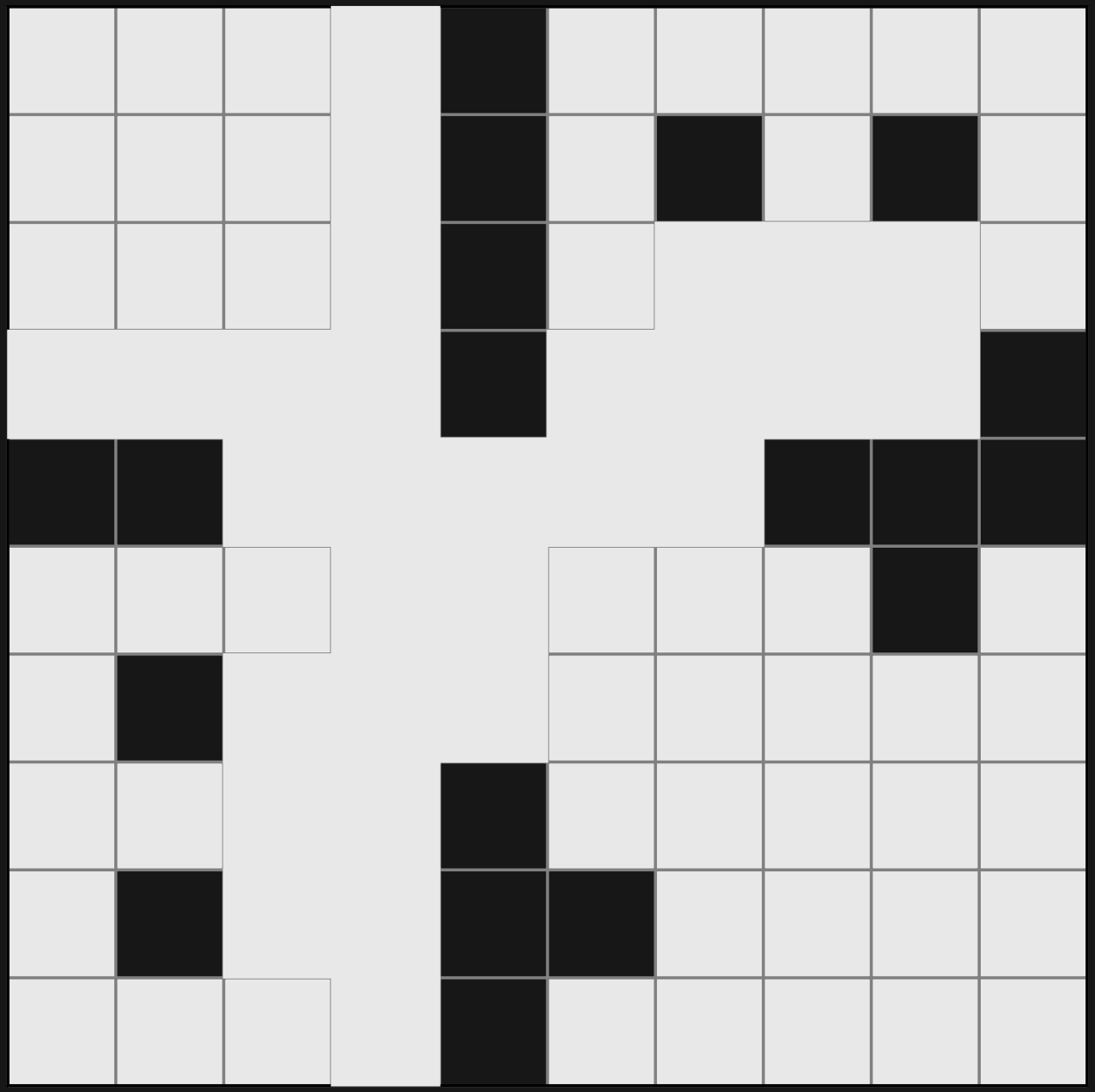
AND gate in f_2



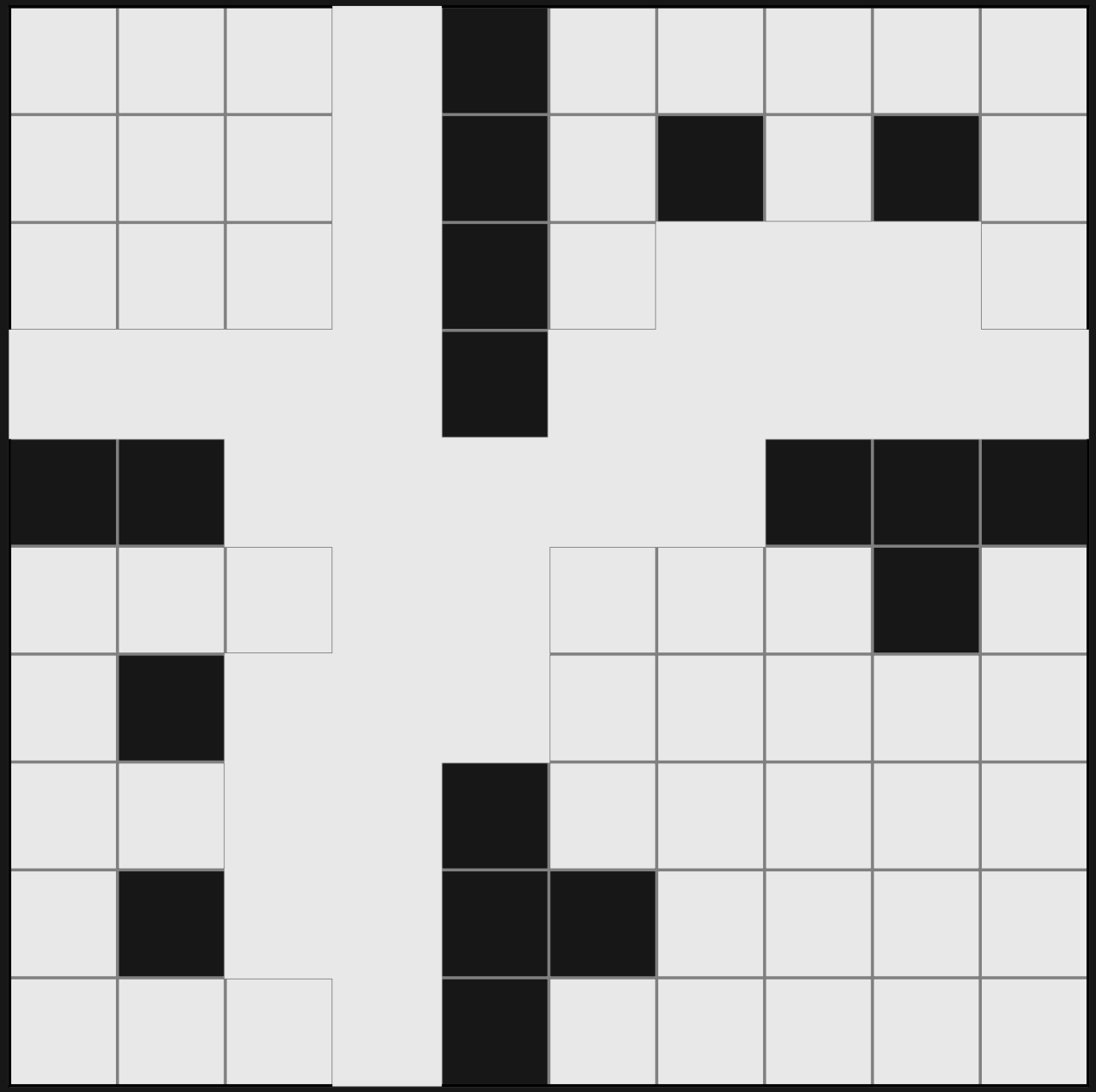
AND gate in f_2



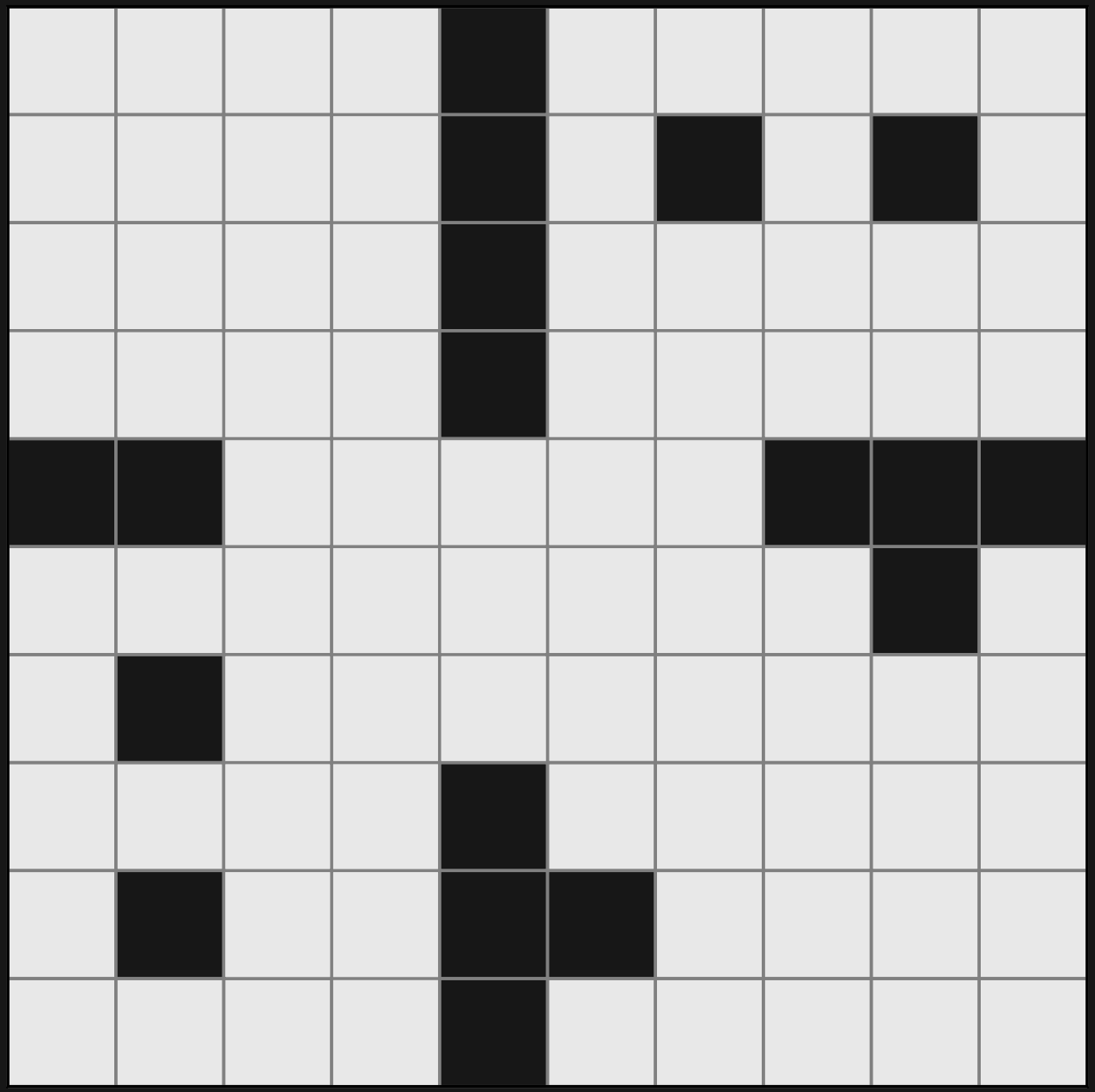
AND gate in f_2



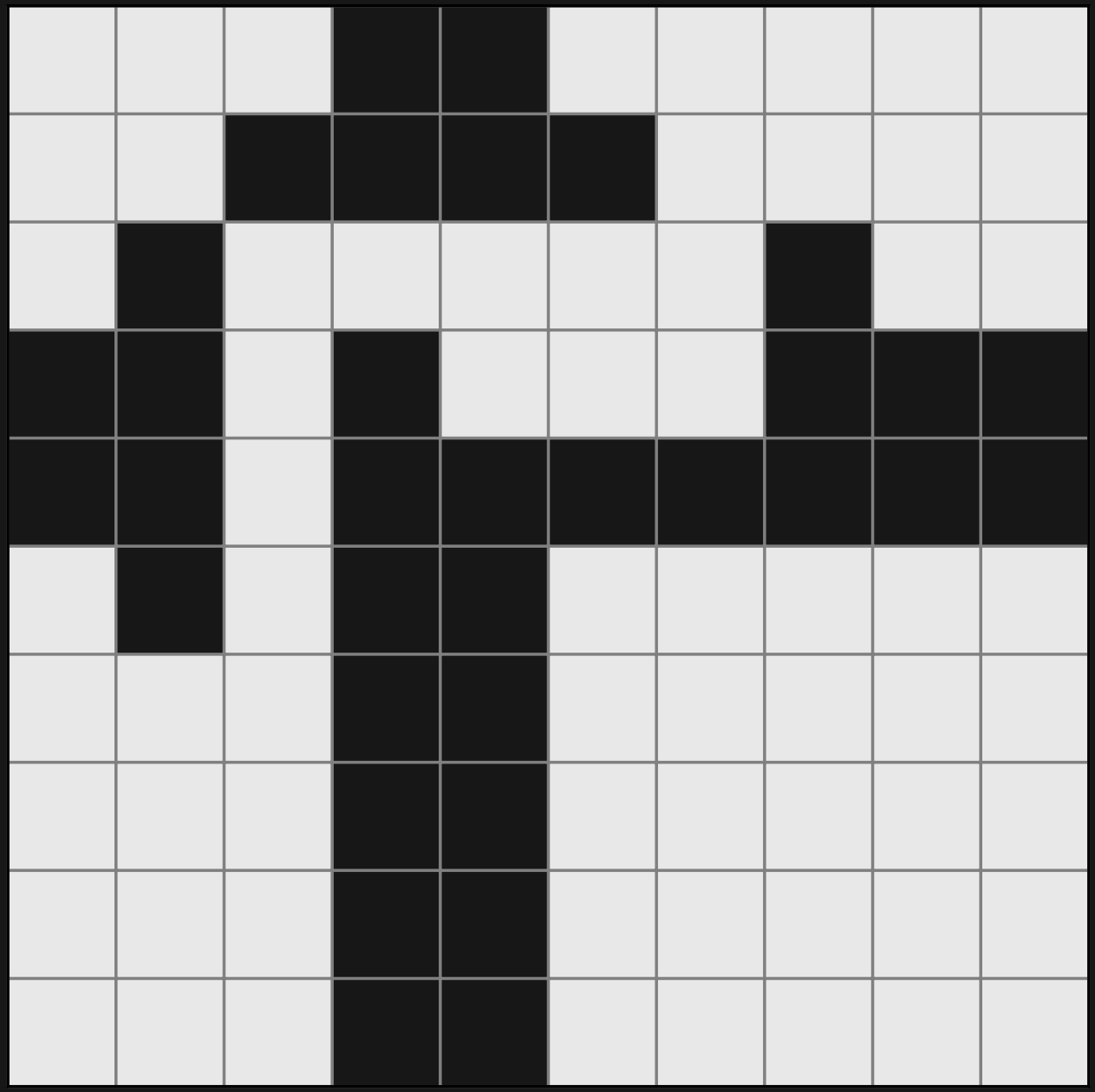
AND gate in f_2



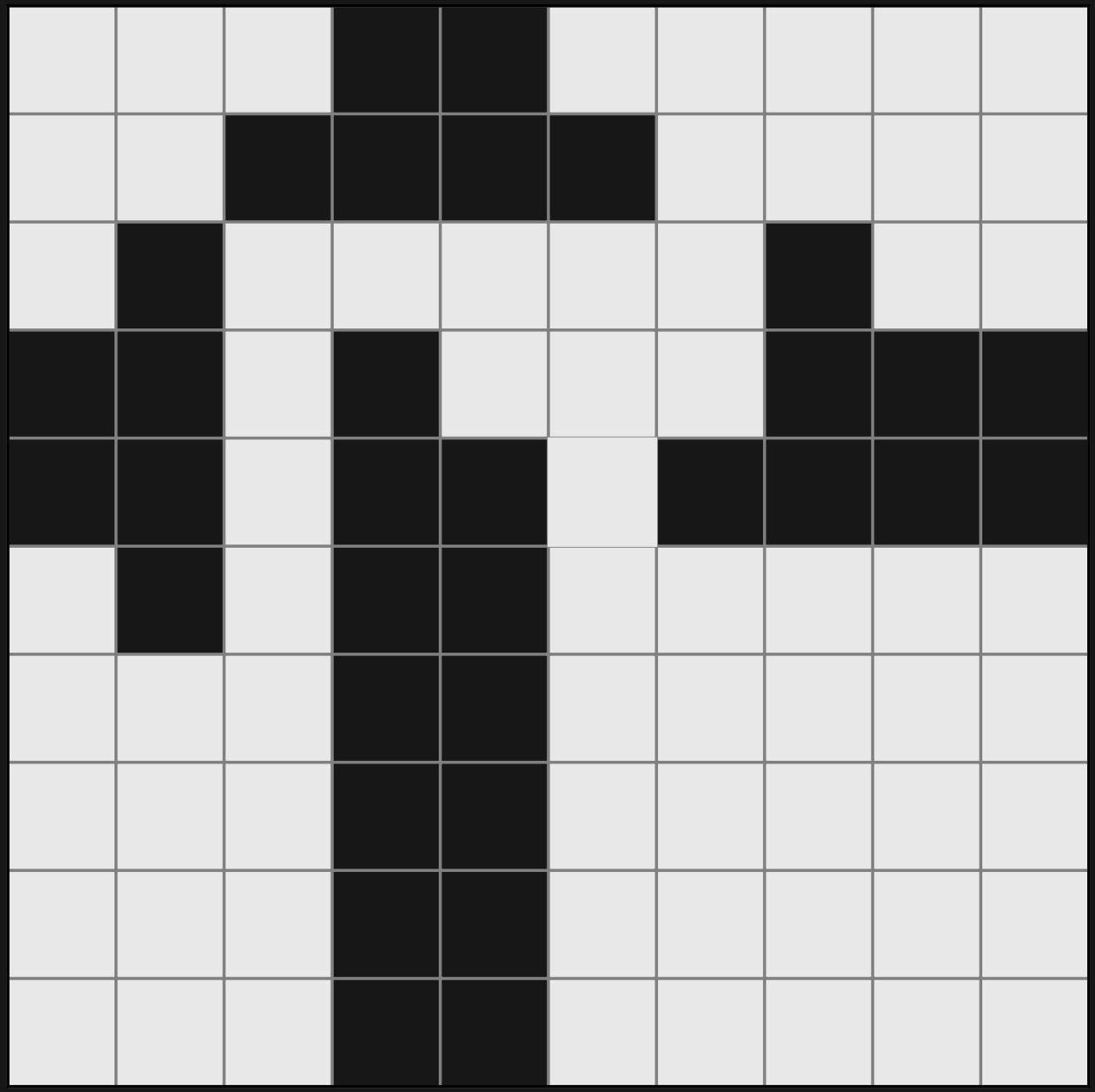
AND gate in f_2



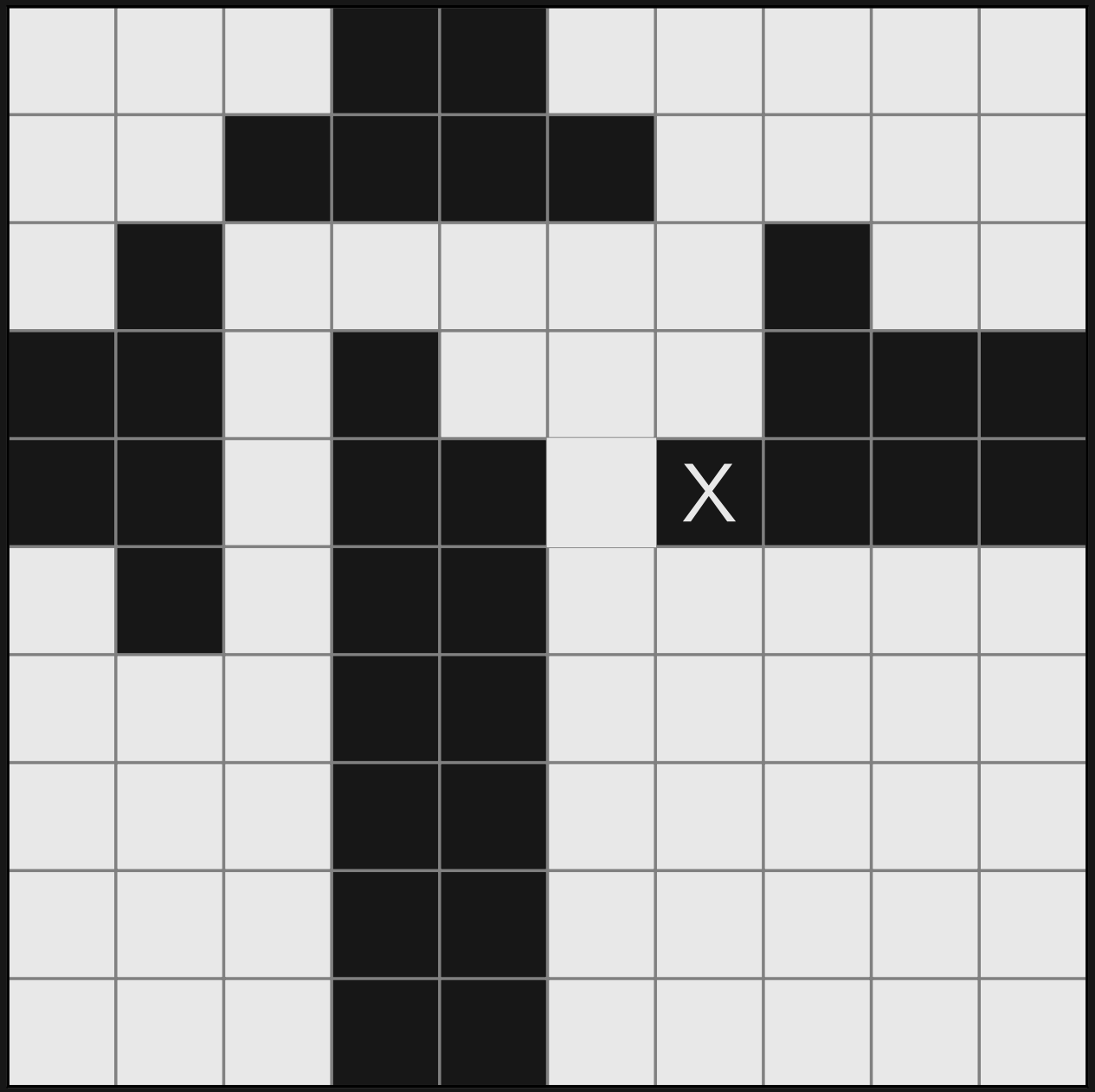
S_i gate in f_2 I



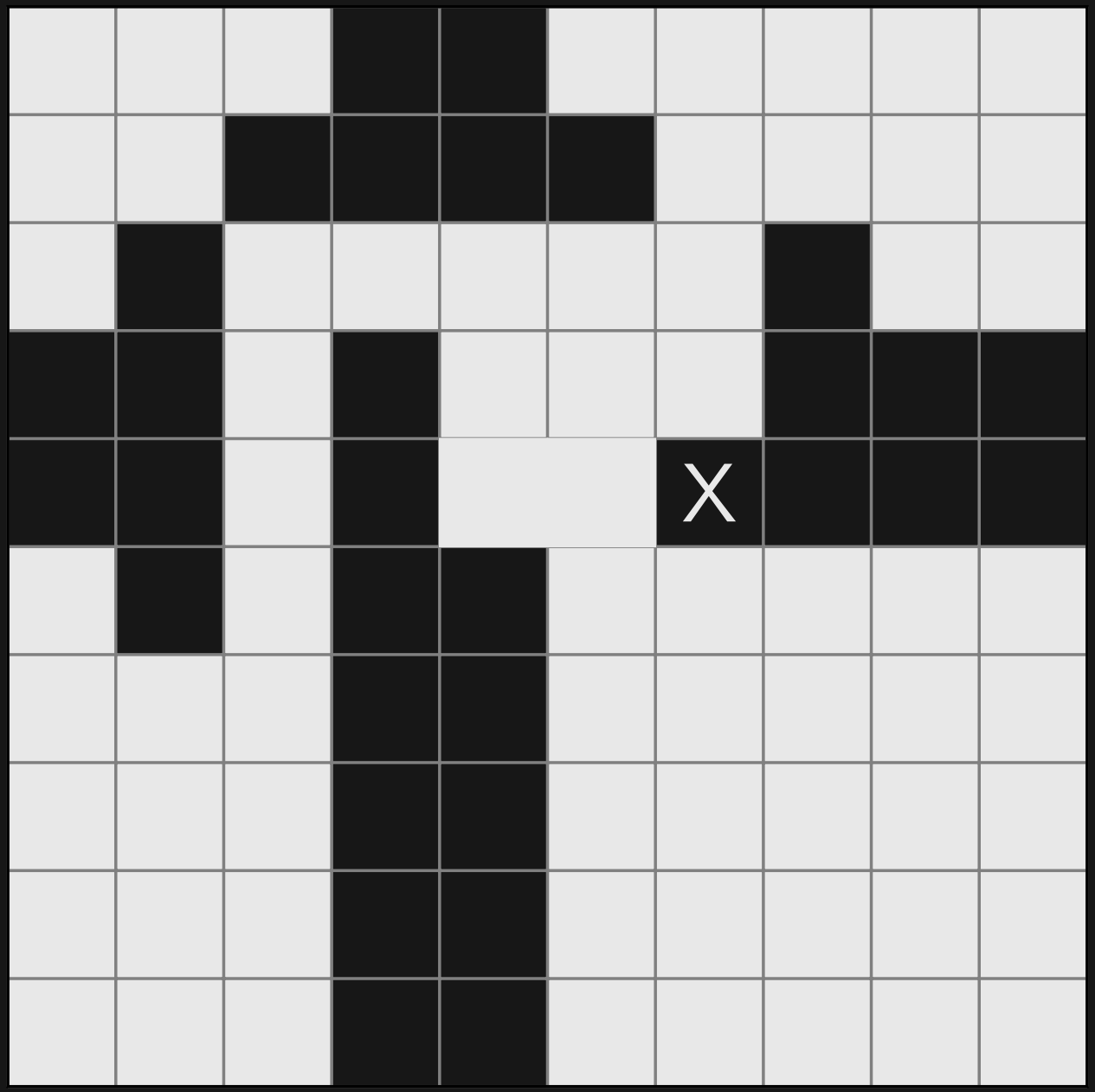
S_i gate in f_2 I



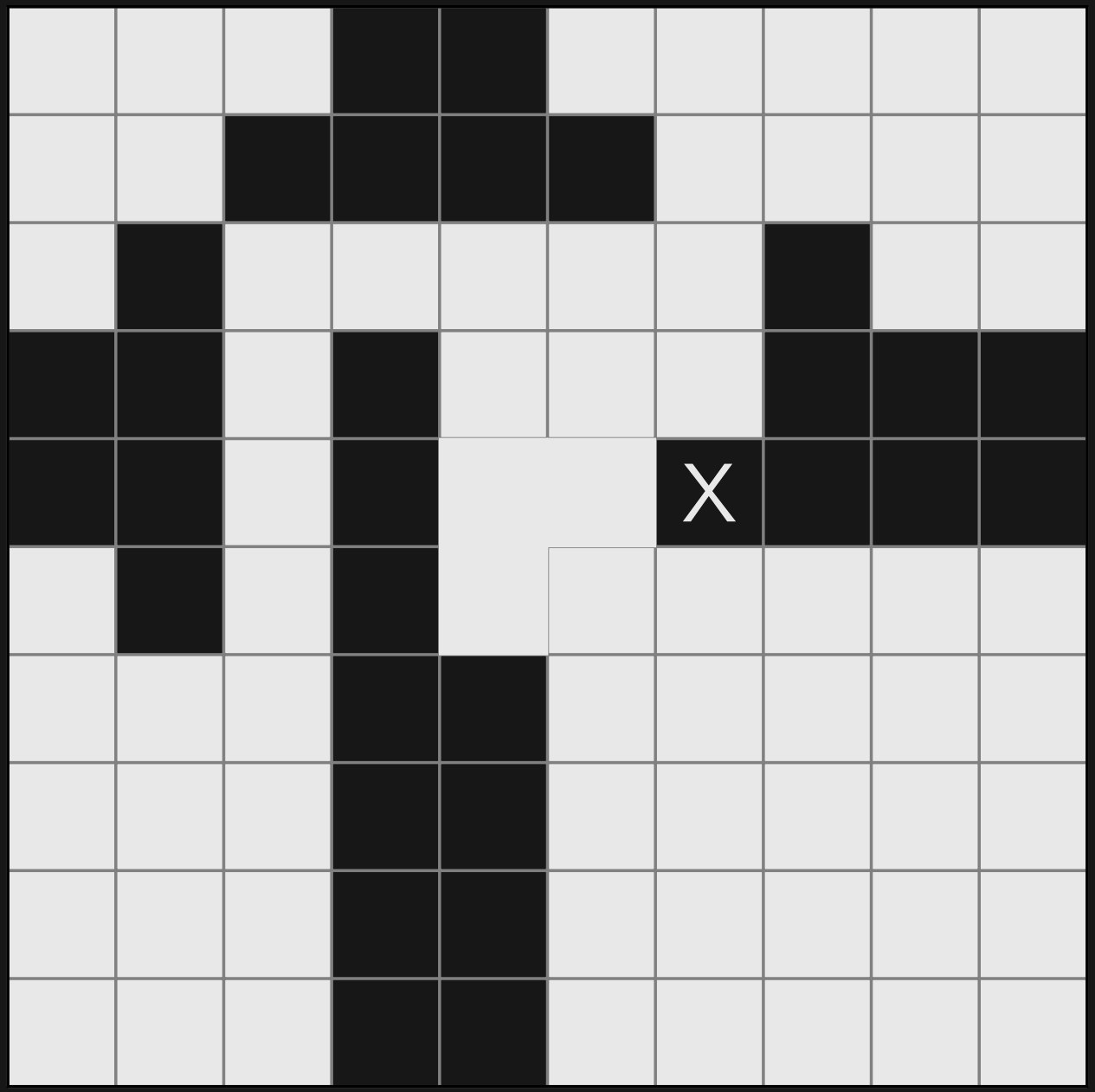
S_i gate in f_2 I



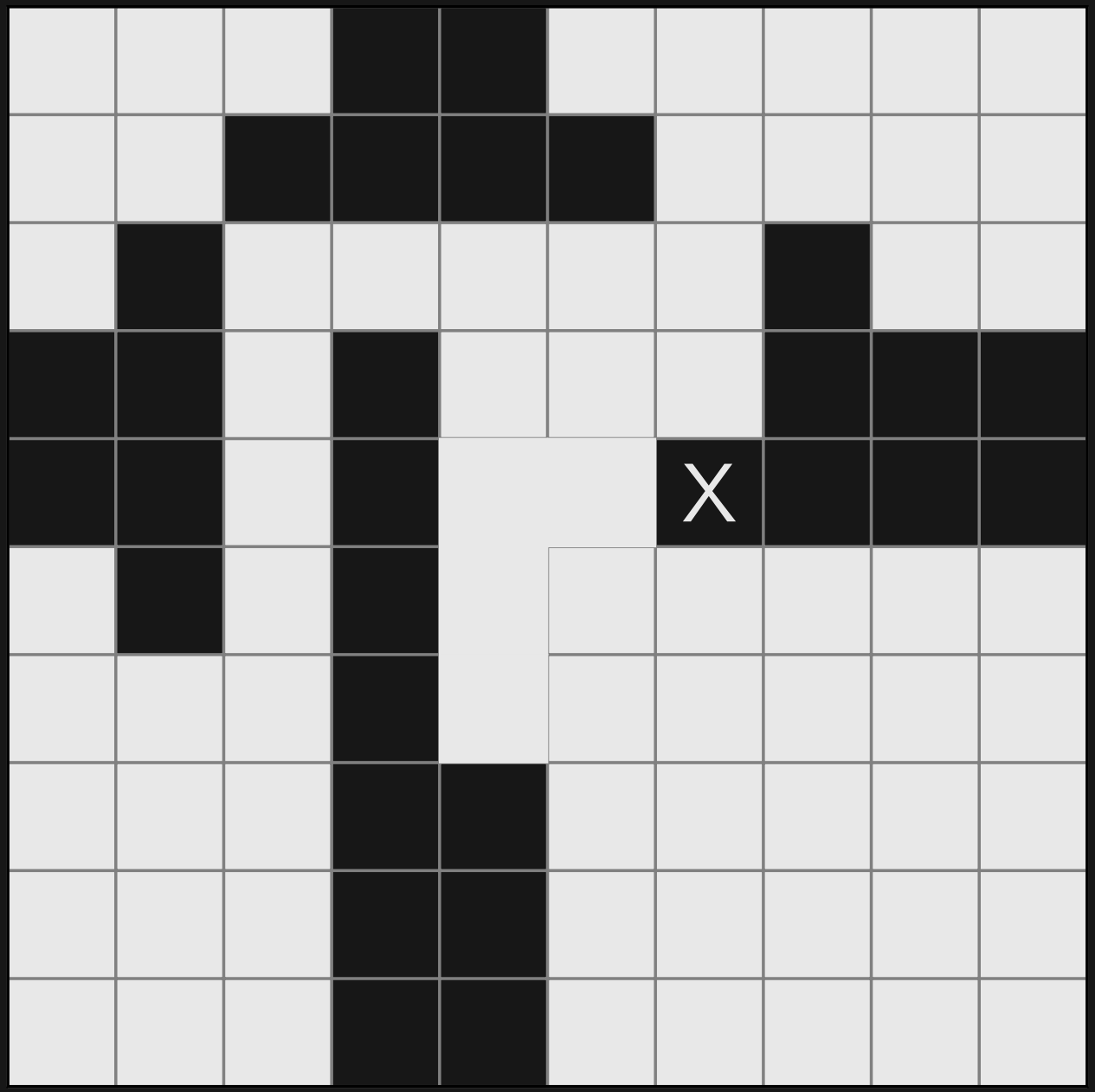
S_i gate in f_2 I



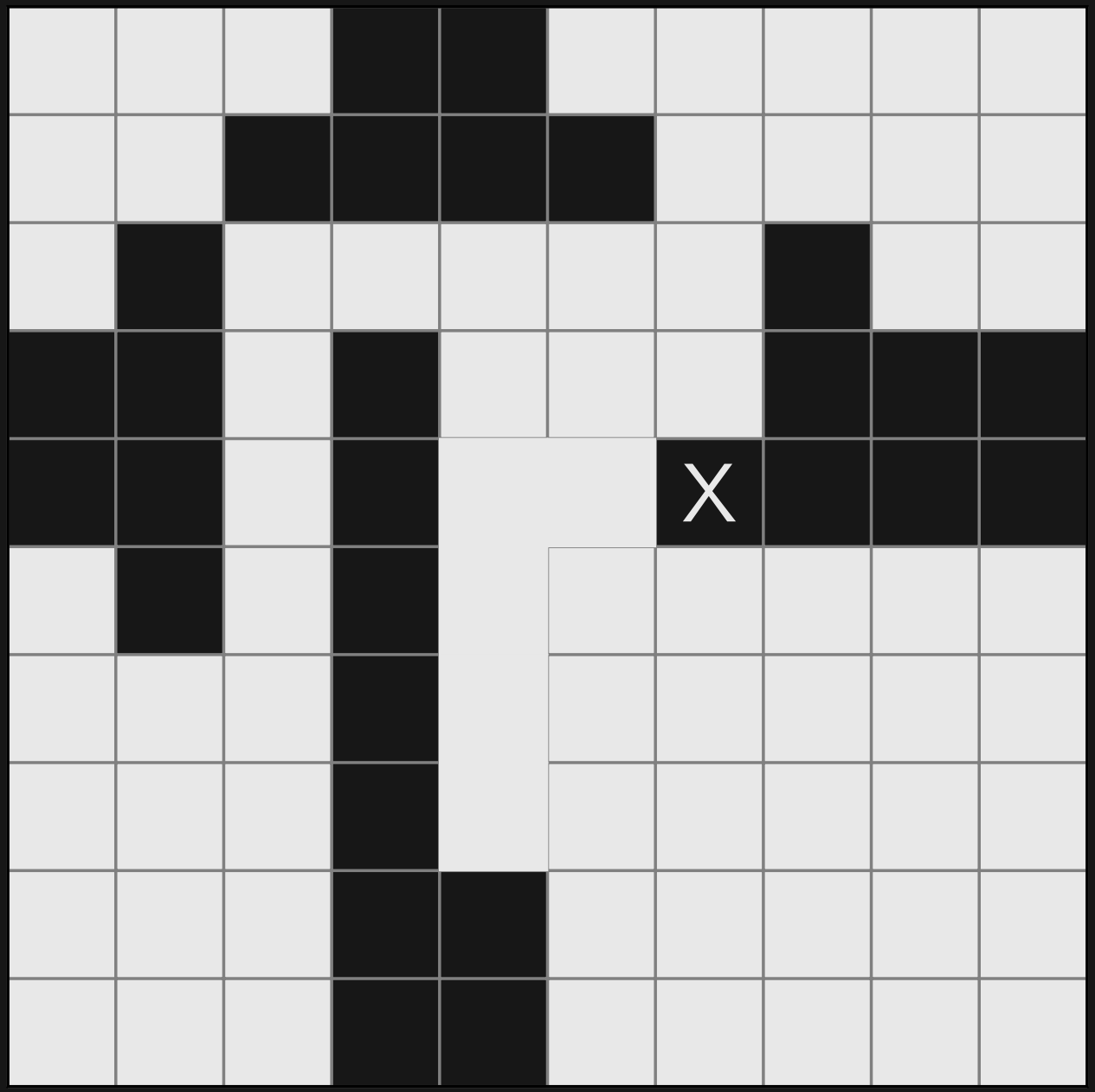
S_i gate in f_2 I



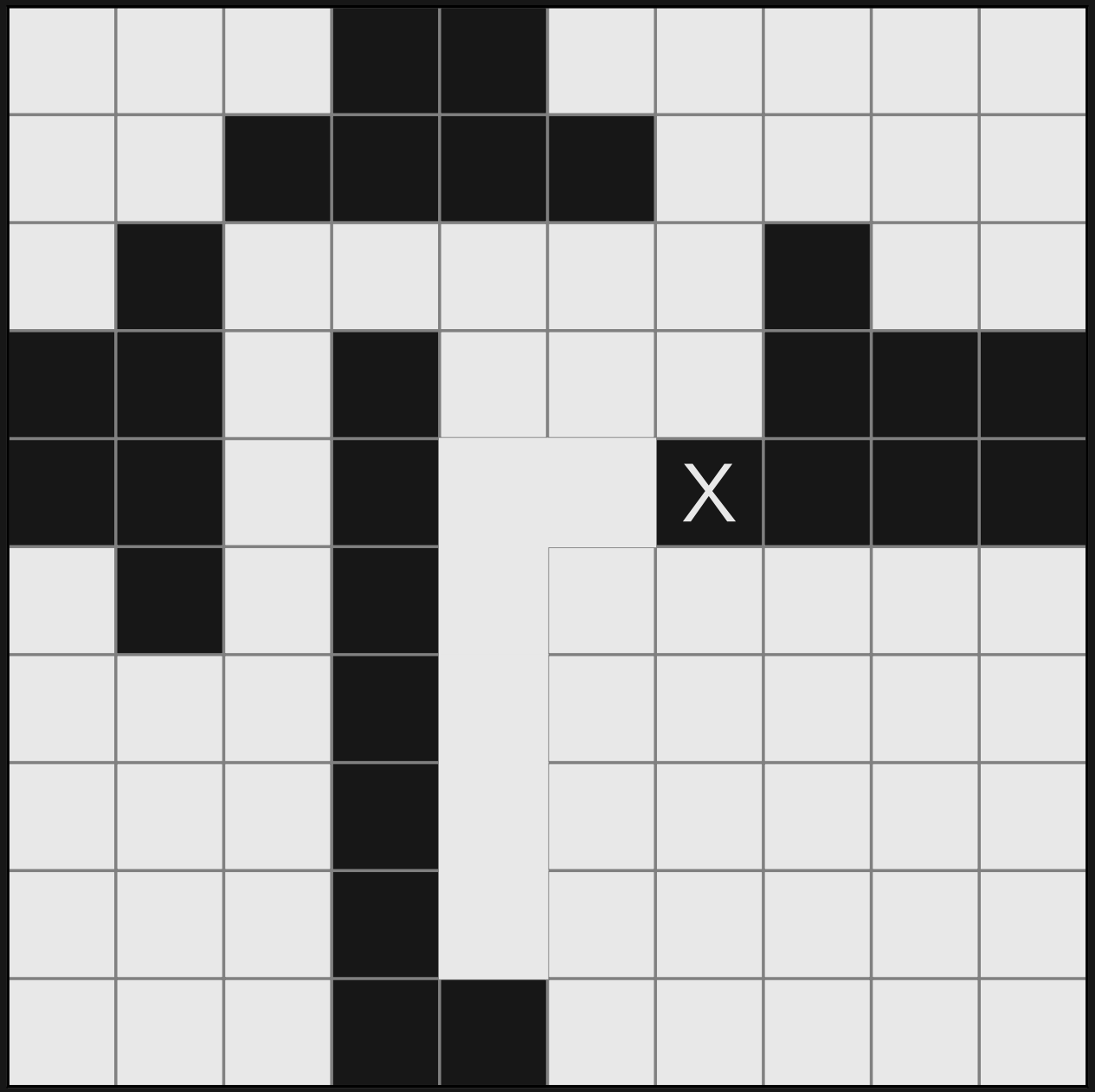
S_i gate in f_2 I



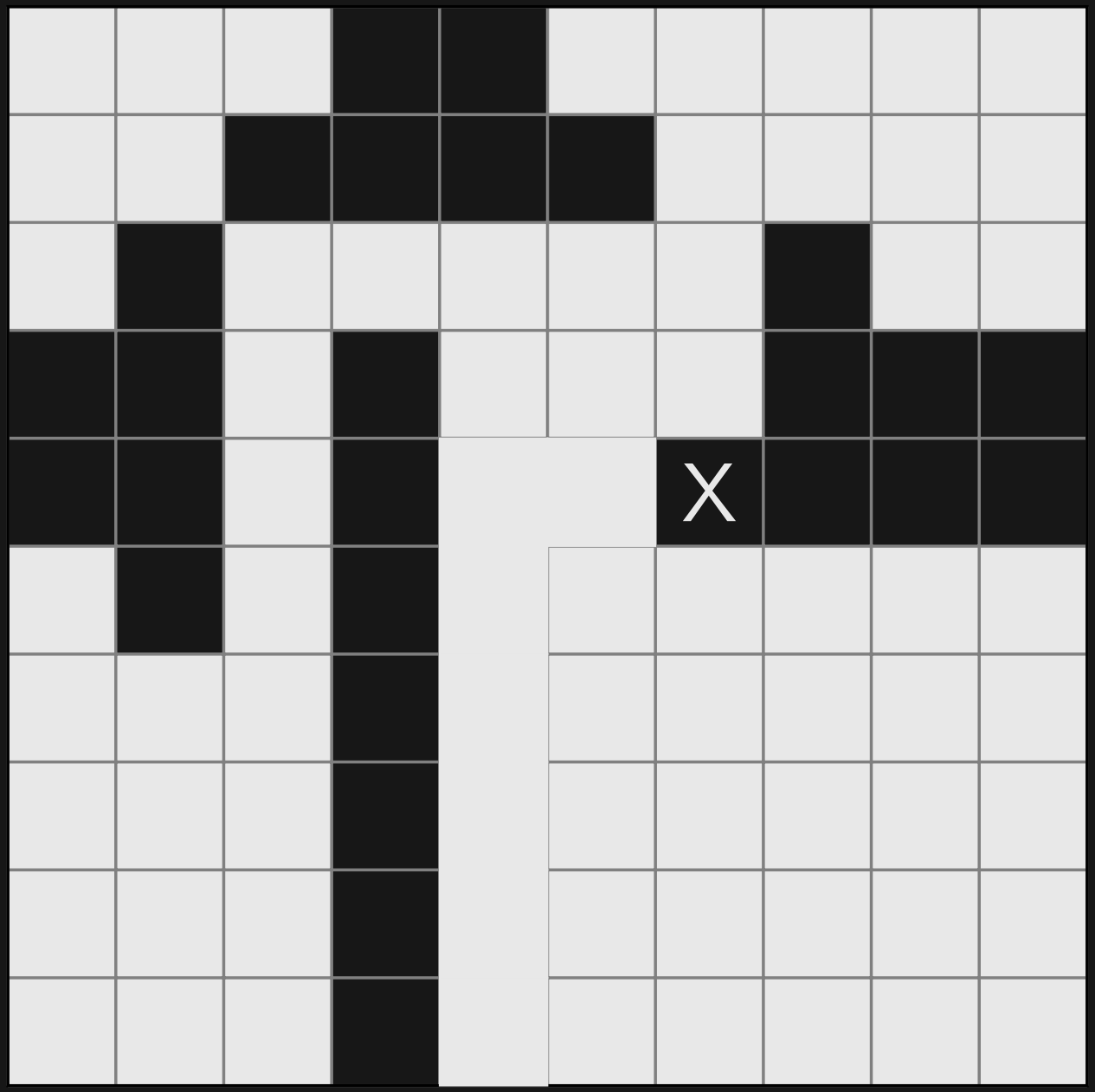
S_i gate in f_2 I



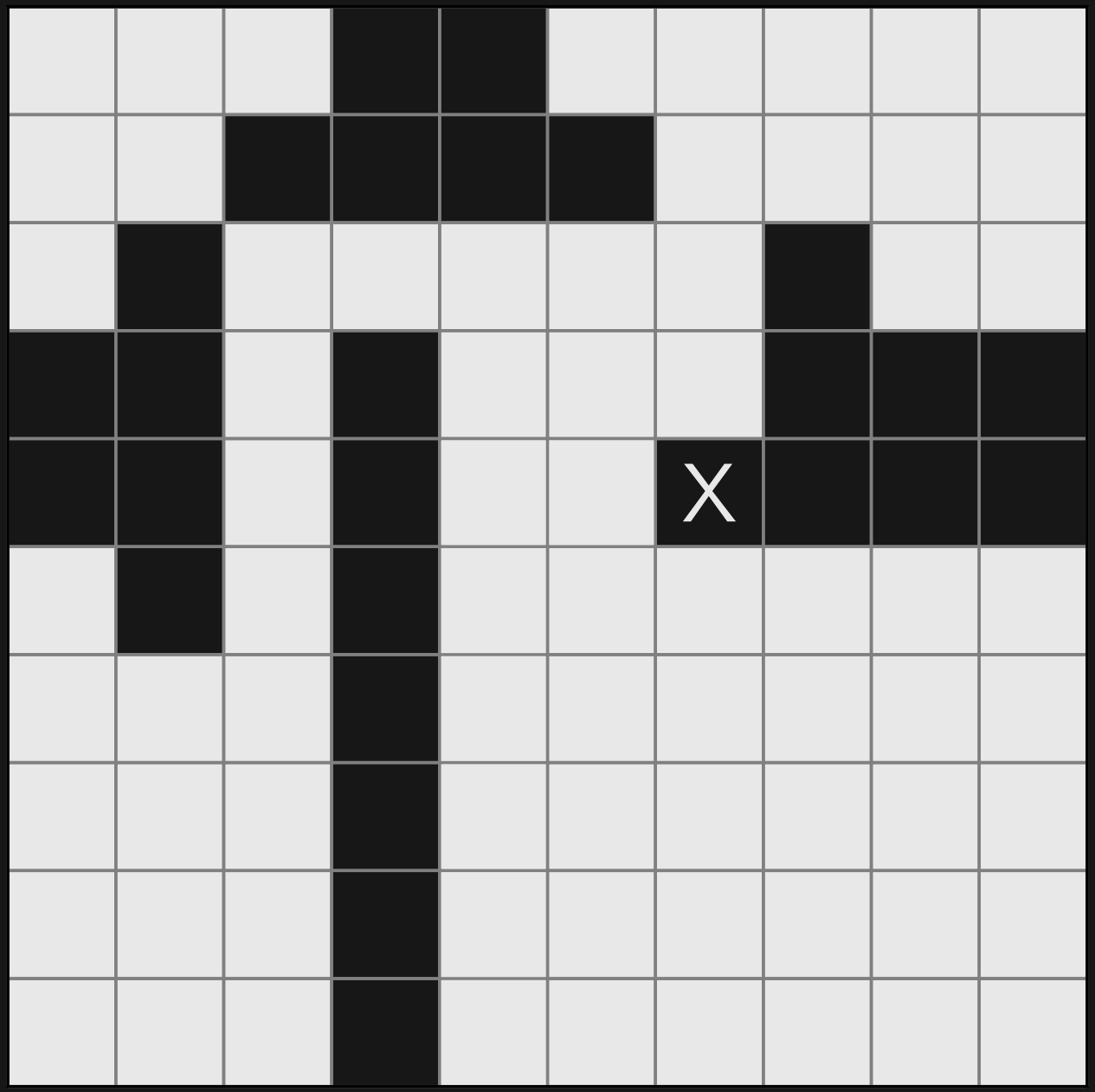
S_i gate in f_2 I



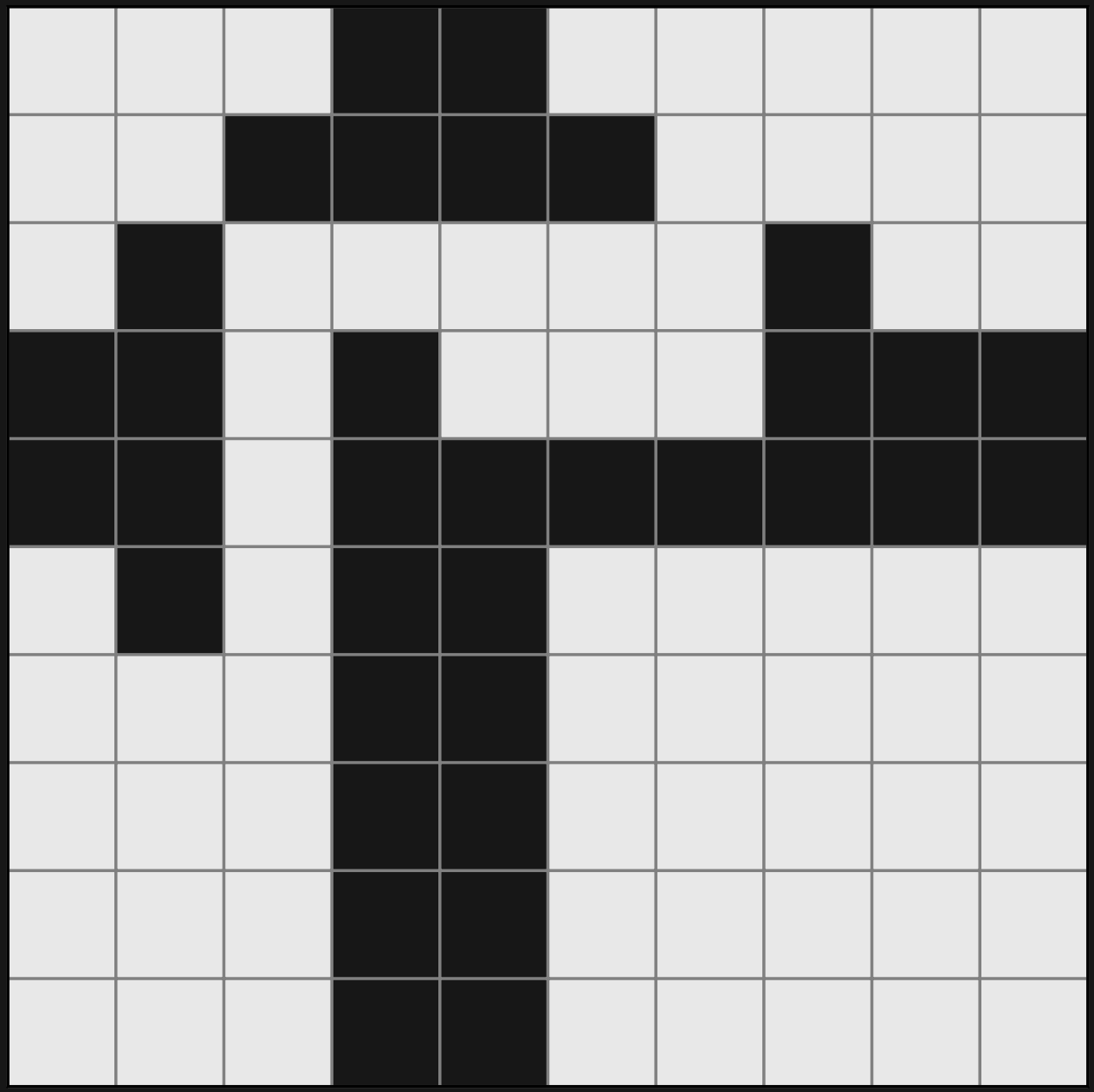
S_i gate in f_2 I



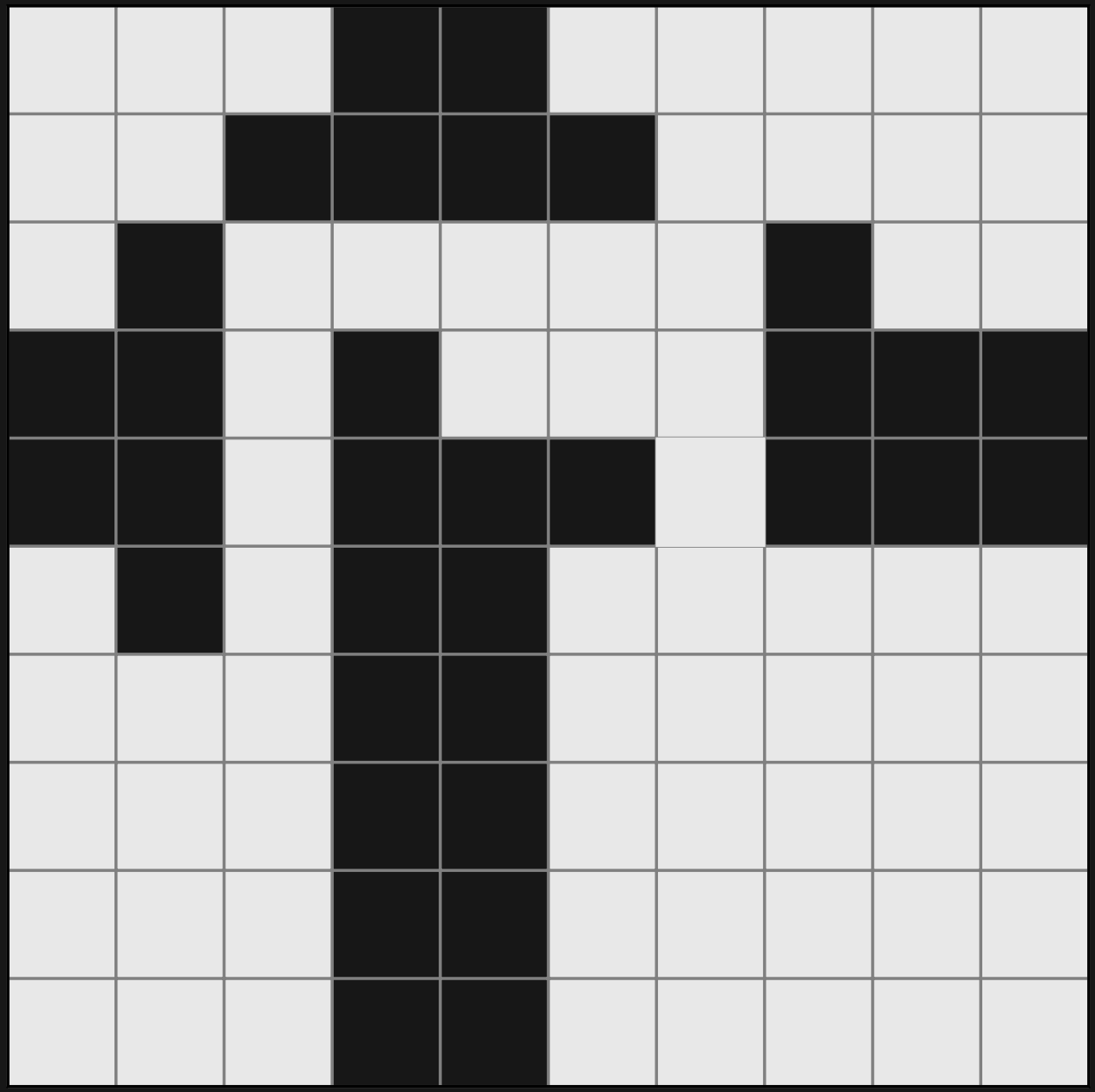
S_i gate in f_2 I



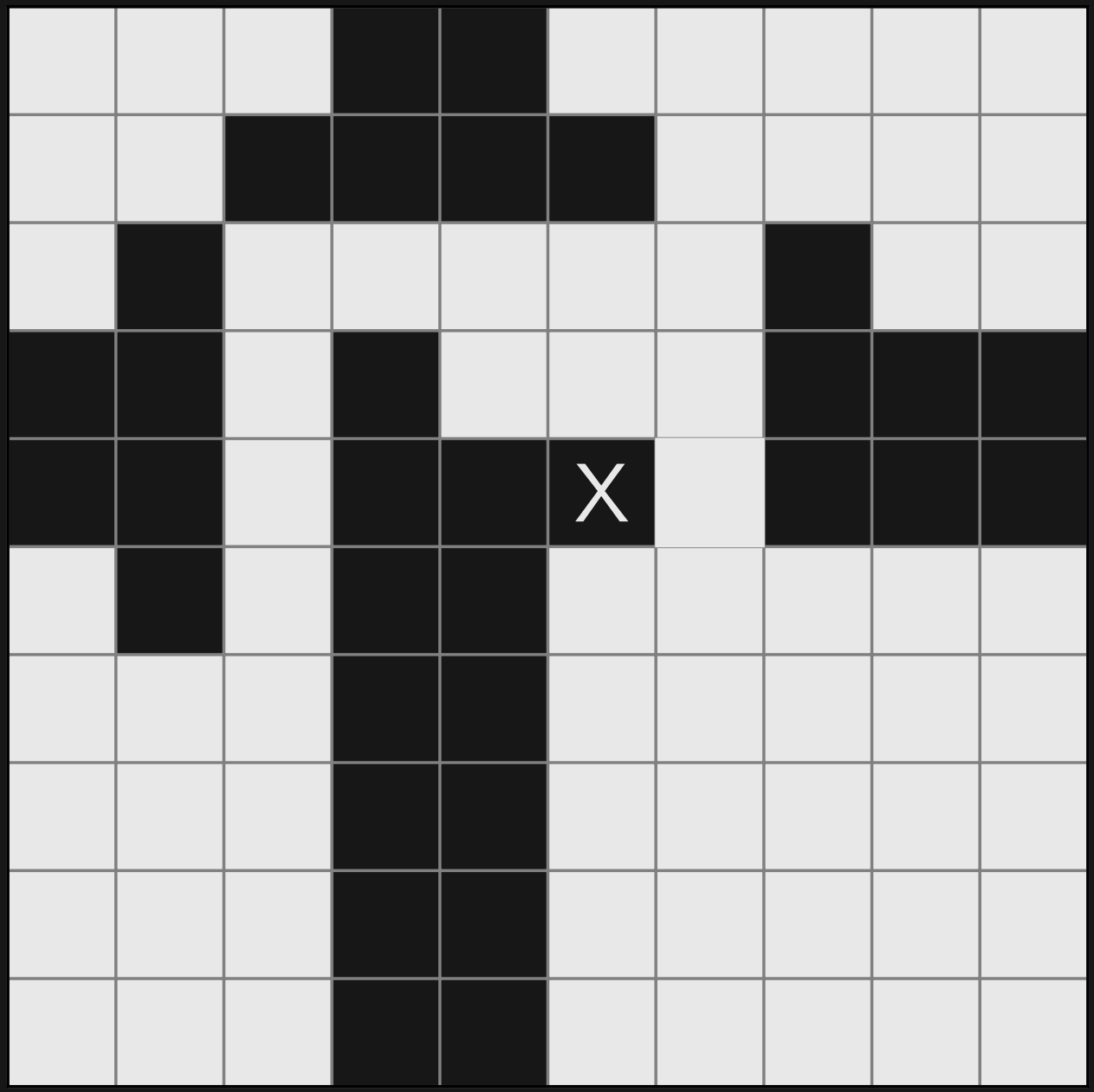
S_i gate in f_2 II



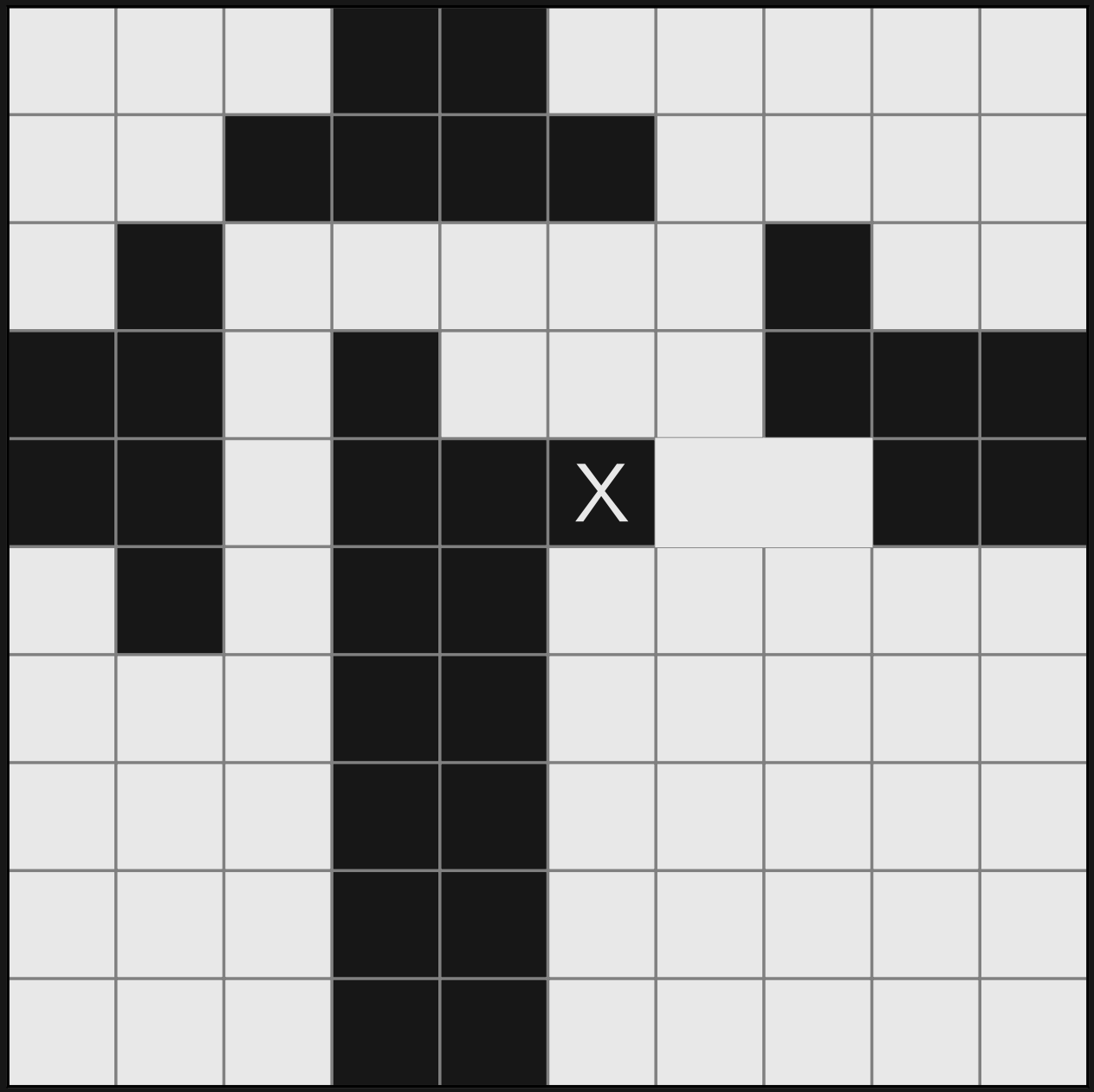
S_i gate in f_2 II



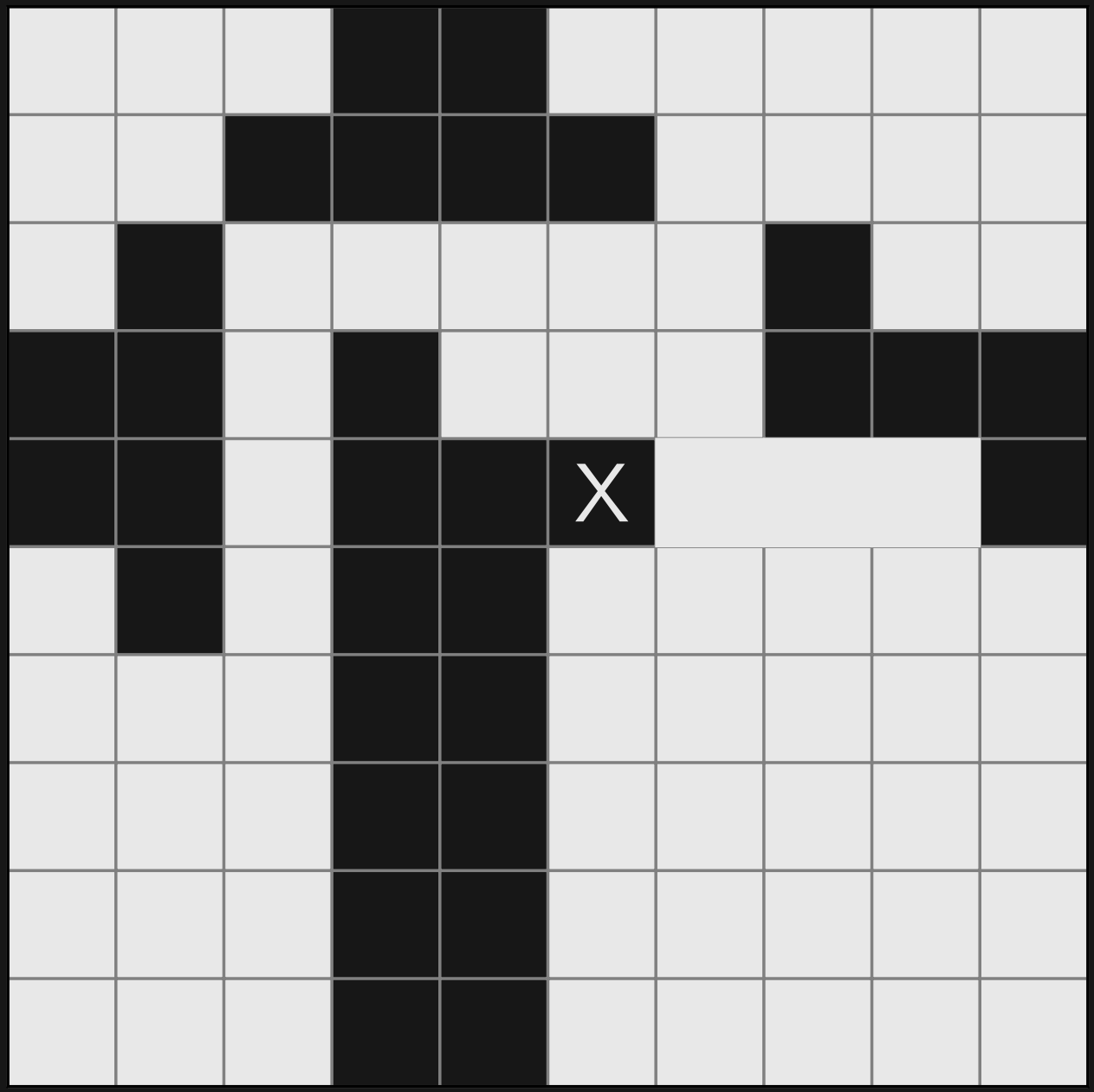
S_i gate in f_2 II



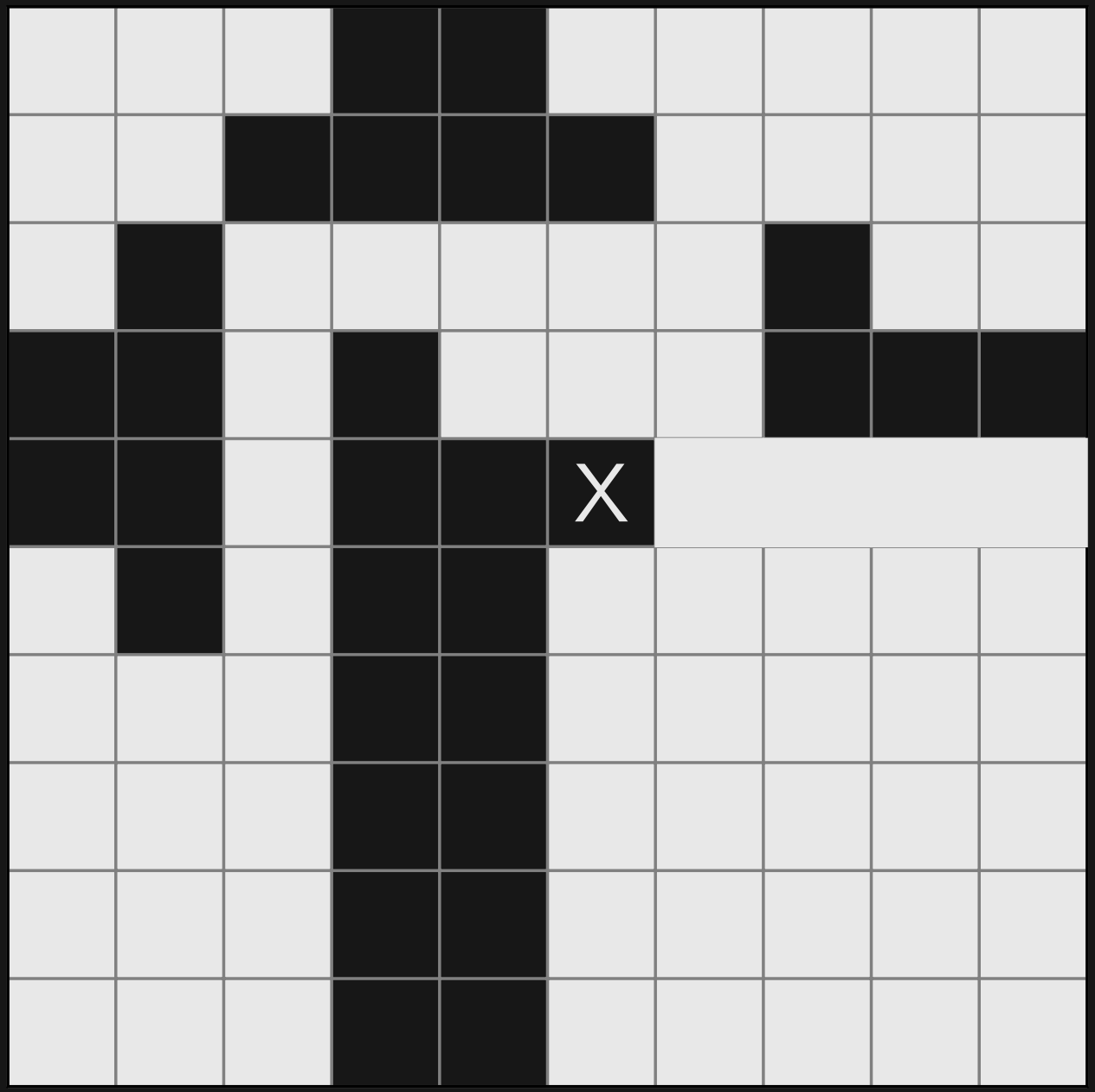
S_i gate in f_2 II



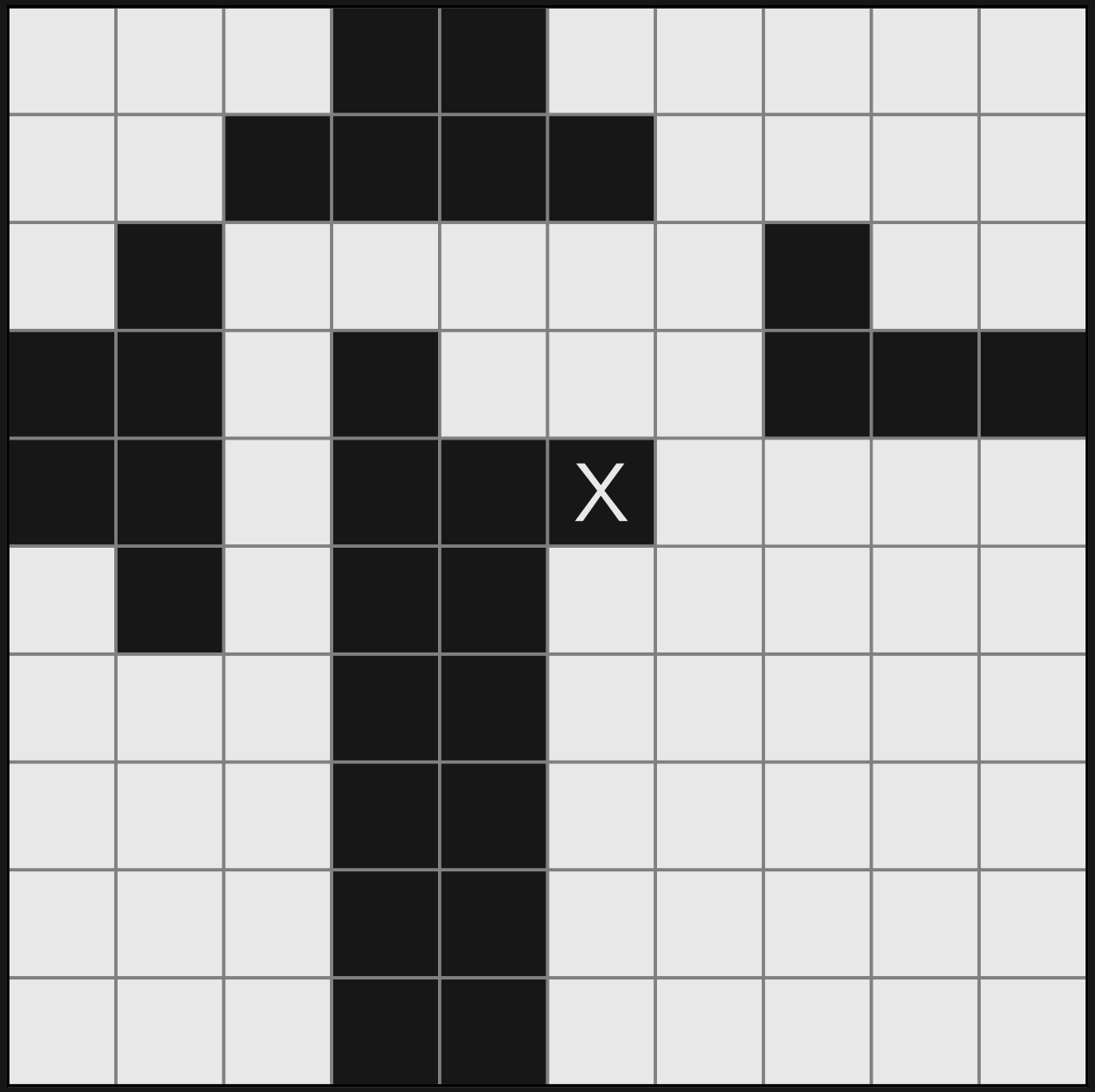
S_i gate in f_2 II



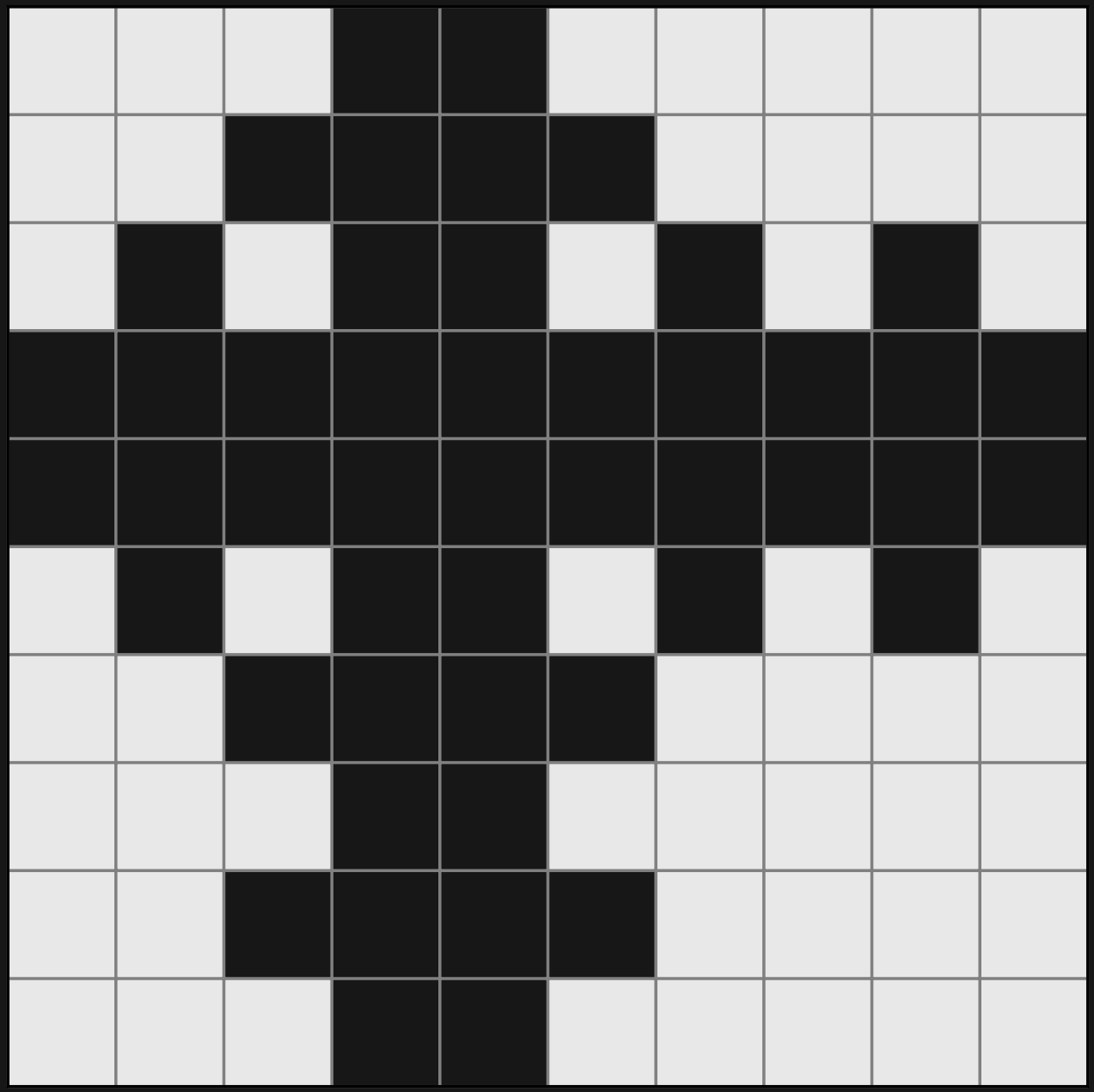
S_i gate in f_2 II



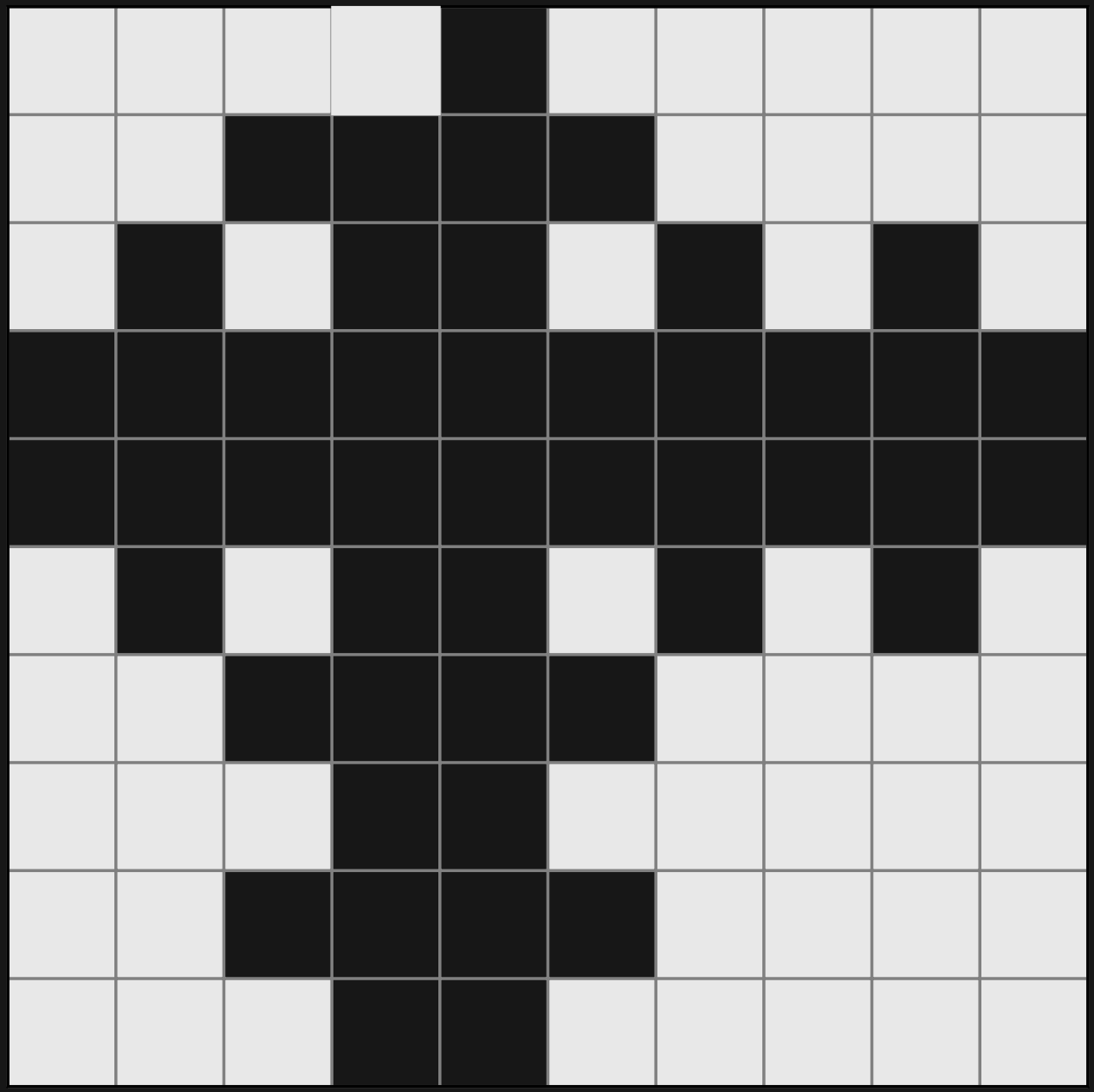
S_i gate in f_2 II



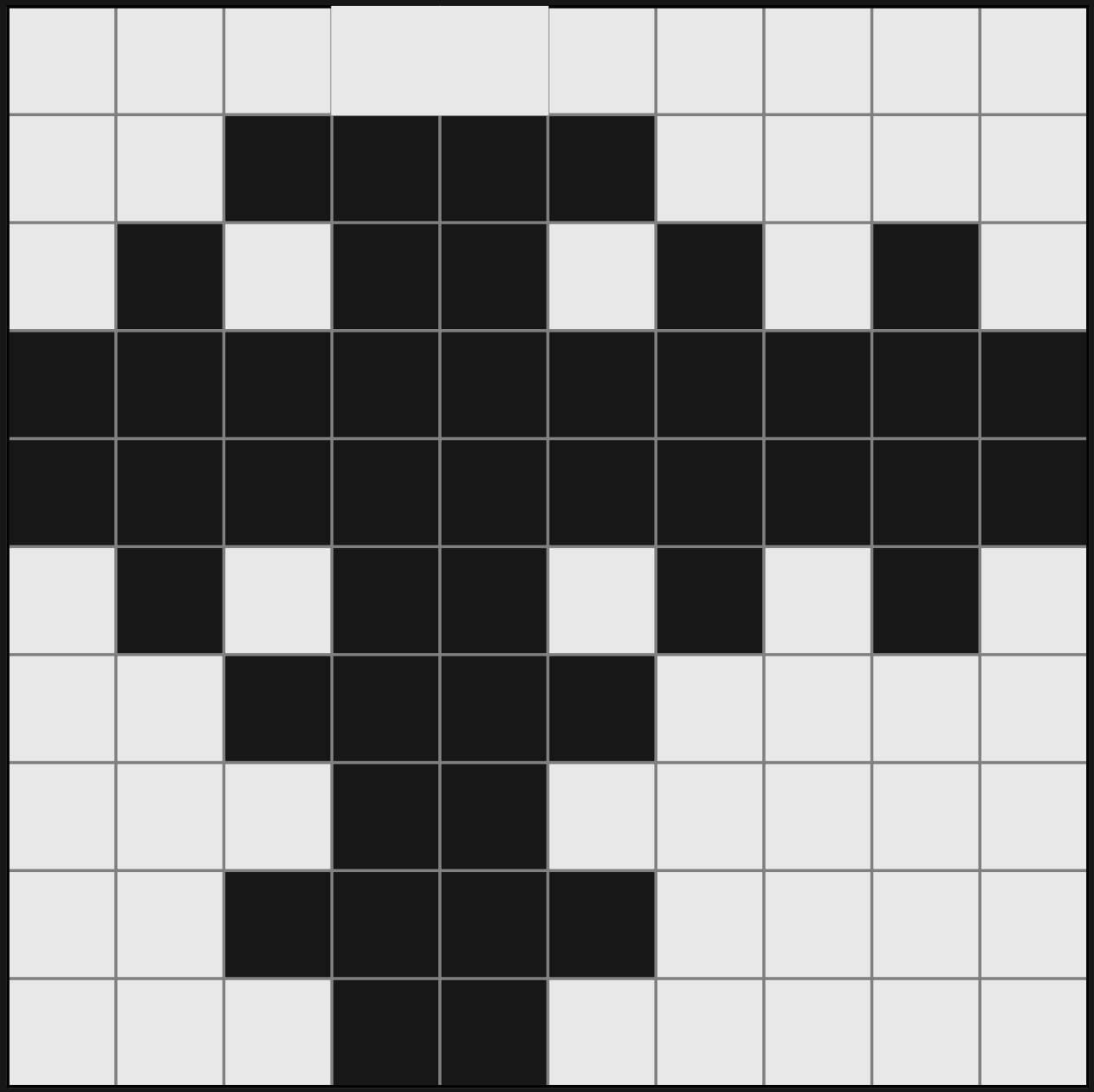
0 gate in f_2 II



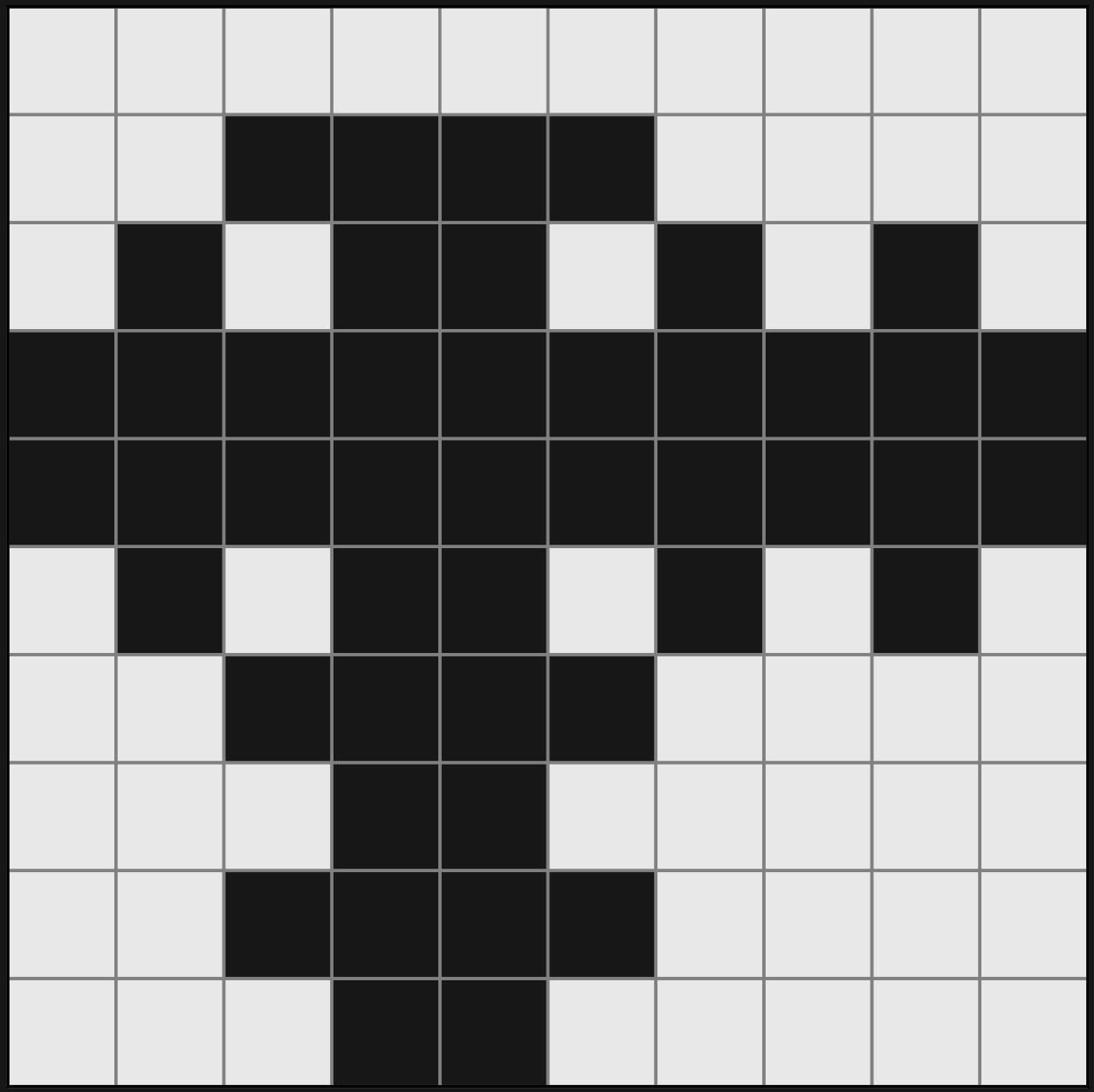
0 gate in f_2 II



0 gate in f_2 II



0 gate in f_2 II



Concluding remarks

- We have a very restrictive NP-complete problem for (A)CA
- We simple local rule with a complex behavior in synchronous and asynchronous update
- To explore the “one way ” (A)CA

Gracias!